BAKERY

fresh & delicious

Cake Bakery Management System

Uqba Gulzar (067)
Zunaira Khatoon (074)

Software Design Description (SDD)

Project Title: *Online Cake Bakery Management System*

Frontispiece

- **Date of Issue and Status**: May 3, 2025 – Final Version
- **Issuing Organization**: FJWU
- **Authorship**: Uqba Gulzar, Zunaira Khatoon
- **Change History**:

Date	Organization	Author
Saturday, May 3, 2025	FJWU	Uqba Gulzar Zunaira Khatoon

Introduction

Purpose

The purpose of the Cake Bakery Management System is to provide a structured and object-oriented software solution for managing customer orders and bakery operations. It allows customers to place, edit, and delete cake orders, while enabling the admin to manage customer details, view orders, and access the menu. The system is developed to automate the traditional manual processes of a bakery, ensuring improved efficiency and better user experience.

Scope

The system includes two primary users: Customers and Admins**.** Customers can register their details and perform operations like placing new orders, editing existing ones, or canceling them. Admins are responsible for managing customer data, viewing all placed orders, and accessing the complete menu. The backend is organized using object-oriented programming principles, with abstract classes, declared classes for customer details, and role-specific functionality separated into different classes for better maintainability and scalability.

Context

This Cake Bakery Management System include object-oriented programming concepts such as abstraction**,** inheritance, and class-based structure to a real-world problem. It simulates the core operations of a cake bakery in a digital form, serving as a model system for academic learning or small business use. The system reflects how real-world entities like customers, admins, and orders can be translated into class structures and functionality using code.

Summary

The Cake Bakery Management System is a class-based software project that encapsulates the key features needed to manage a bakery digitally. It makes use of abstract and concrete classes to define roles and operations, enabling customers to interact with the system and place or manage their orders. The admin has access to customer records, the order list, and the full menu. Overall, the system is structured to be modular, reusable, and aligned with software development best practices using object-oriented principles.

References

- IEEE Std 1016-2009

Glossary

- **DBMS**: Database Management System
- **MVC**: Model View Controller
- **CRUD**: Create, Read, Update, Delete
- **Stakeholder**: Admin, Customer
- **SDD**: Software Design Document

Body

Identified Stakeholders

- Admin
- Customer

Design Viewpoint 1: Use Case View

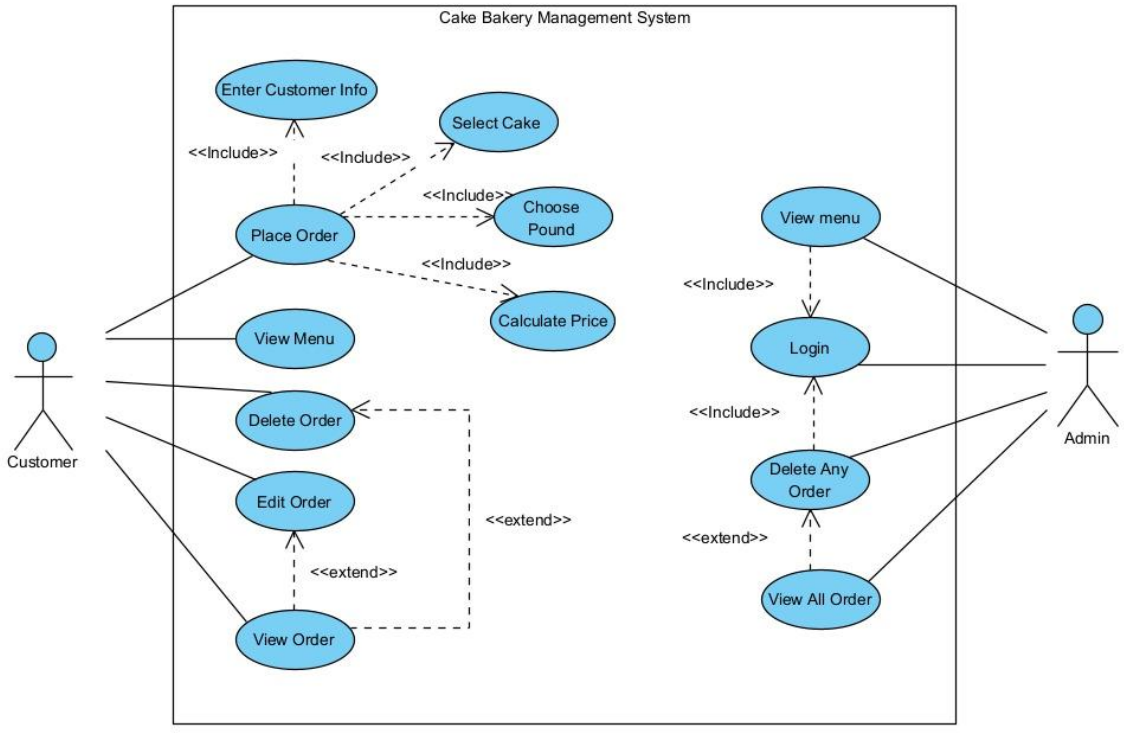
Use Cases:

Customer

- View Menu
- Browse Cakes
- Place Order
- Enter Data
- Edit Order
- Delete Order

Admin

- Login
- View Menu
- View all Order



Use Case: Place Order

Use Case Name: Place Order
Scope: Cake Bakery Management System
Level: User Goal
Primary Actor: Customer
Stakeholders and Interests:

- **Customer:** Wants to successfully place a cake order.
- **Bakery:** Wants accurate order information for timely fulfillment.

Preconditions:

- The system is running and accessible.
- Customer has access to the system interface (website or app).

Postconditions:

- The order is saved in the system with all relevant information.
- The order details are available for viewing or editing.

Main Success Scenario:

1. Customer selects "Place Order."
2. System prompts for customer information.
3. Customer enters personal details (name, contact info, etc.).
4. System prompts customer to select a cake.
5. Customer selects a cake from the menu.
6. System asks for the cake size in pounds.
7. Customer chooses the pound (weight).
8. System calculates the price.
9. System displays the order summary with price.
10. Customer confirms the order.
11. System stores the order and confirms with an order ID.

Extensions:

- **2a. Invalid or incomplete customer info:**
  - 2a1. System prompts for correction.
  - 2a2. Use case resumes at Step 3.
- **5a. Cake is not available:**
  - 5a1. System shows out-of-stock message.
  - 5a2. Customer selects another cake or exits.
- **10a. Customer cancels order before confirming:**
  - 10a1. System discards order data.
  - 10a2. Use case ends.

Special Requirements:

- System must validate all inputs before proceeding.
- Pricing should be calculated based on dynamic rates.

Use Case: View Menu

Scope: Cake Bakery Management System
Primary Actor: Customer / Admin
Preconditions: System is accessible.
Postconditions: Menu is displayed.

Main Success Scenario:

1. Actor selects "View Menu".
2. System fetches available cakes and options.
3. Menu is displayed with cake names, sizes, and prices.

Use Case: Edit Order

Primary Actor: Customer
Preconditions: Customer must have placed an order.
Postconditions: Order details are updated.

Main Success Scenario:

1. Customer selects “Edit Order”.
2. System shows list of previous orders.
3. Customer selects an order to edit.
4. System displays editable fields (cake type, size, etc.).
5. Customer modifies order details.
6. System recalculates the price.
7. Customer confirms changes.
8. System updates the order.

Extensions:

- **3a. No previous orders:** Show “No orders found” message.

Use Case: Delete Order

Primary Actor: Customer
Preconditions: An order must exist.
Postconditions: Order is deleted.

Main Success Scenario:

1. Customer selects “Delete Order”.
2. System displays existing orders.
3. Customer selects an order to delete.
4. System asks for confirmation.
5. Customer confirms.
6. System deletes the order.

Use Case: View Order

Primary Actor: Customer
Preconditions: Order must exist.
Postconditions: Order details are displayed.

Main Success Scenario:

1. Customer selects “View Order”.
2. System lists all placed orders.
3. Customer clicks on one to view full details.
4. System displays order details (cake type, size, price, status).

Use Case: Login

Primary Actor: Admin
Preconditions: Admin must have credentials.
Postconditions: Admin is logged in.

Main Success Scenario:

1. Admin selects “Login”.
2. System prompts for username and password.
3. Admin enters credentials.
4. System validates and logs in the xfadmin.

Extensions:

- **3a. Incorrect credentials:**

- 3a1. System displays error.
- 3a2. Retry login.

Use Case: Delete Any Order

Primary Actor: Admin
Preconditions: Admin must be logged in.
Postconditions: Selected order is deleted.

Main Success Scenario:

1. Admin logs in.
2. Selects “Delete Any Order”.
3. System displays all customer orders.
4. Admin selects order to delete.
5. System asks for confirmation.
6. Admin confirms.
7. System deletes the order.

Use Case: View All Orders

Primary Actor: Admin
Preconditions: Admin must be logged in.
Postconditions: All orders are displayed.

Main Success Scenario:

1. Admin logs in.
2. Selects “View All Orders”.
3. System fetches and displays list of all orders with customer info.

Design Viewpoint 2: Logical View

- **Class Diagram**:

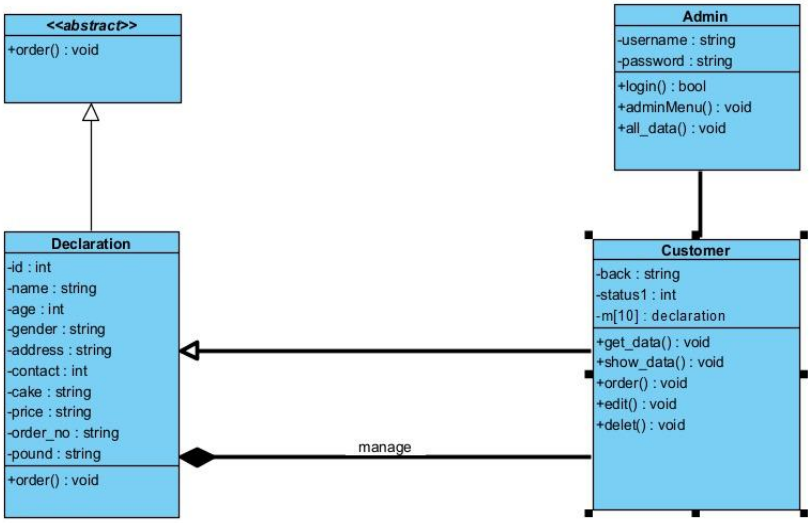
Classes:

- Empty
- Declaration

- Customer
- Admin

Relationships:

- Inheritance
- Abstraction
- Composition
- Association



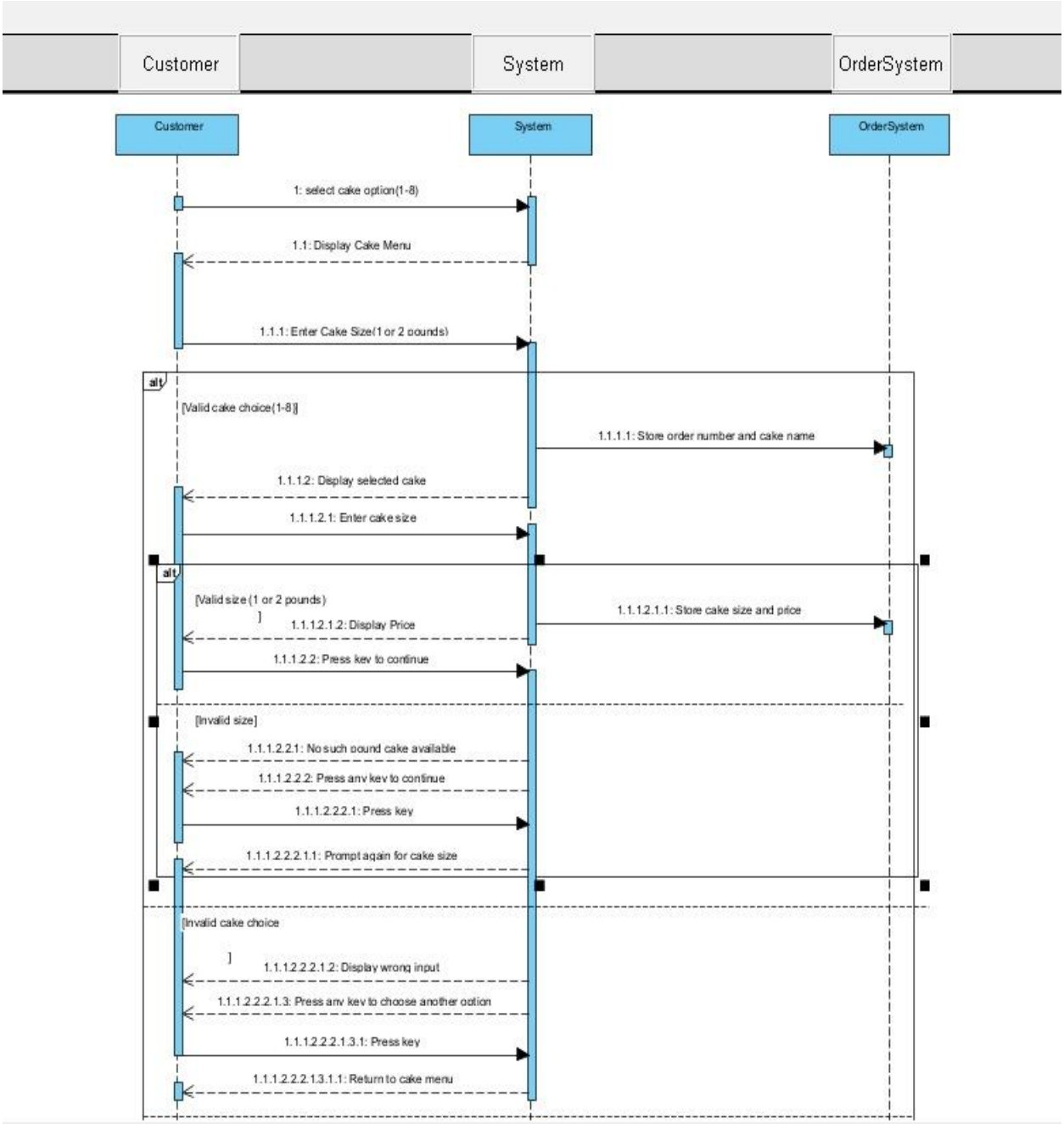
Design Viewpoint 3: Interaction View

Sequence Diagram

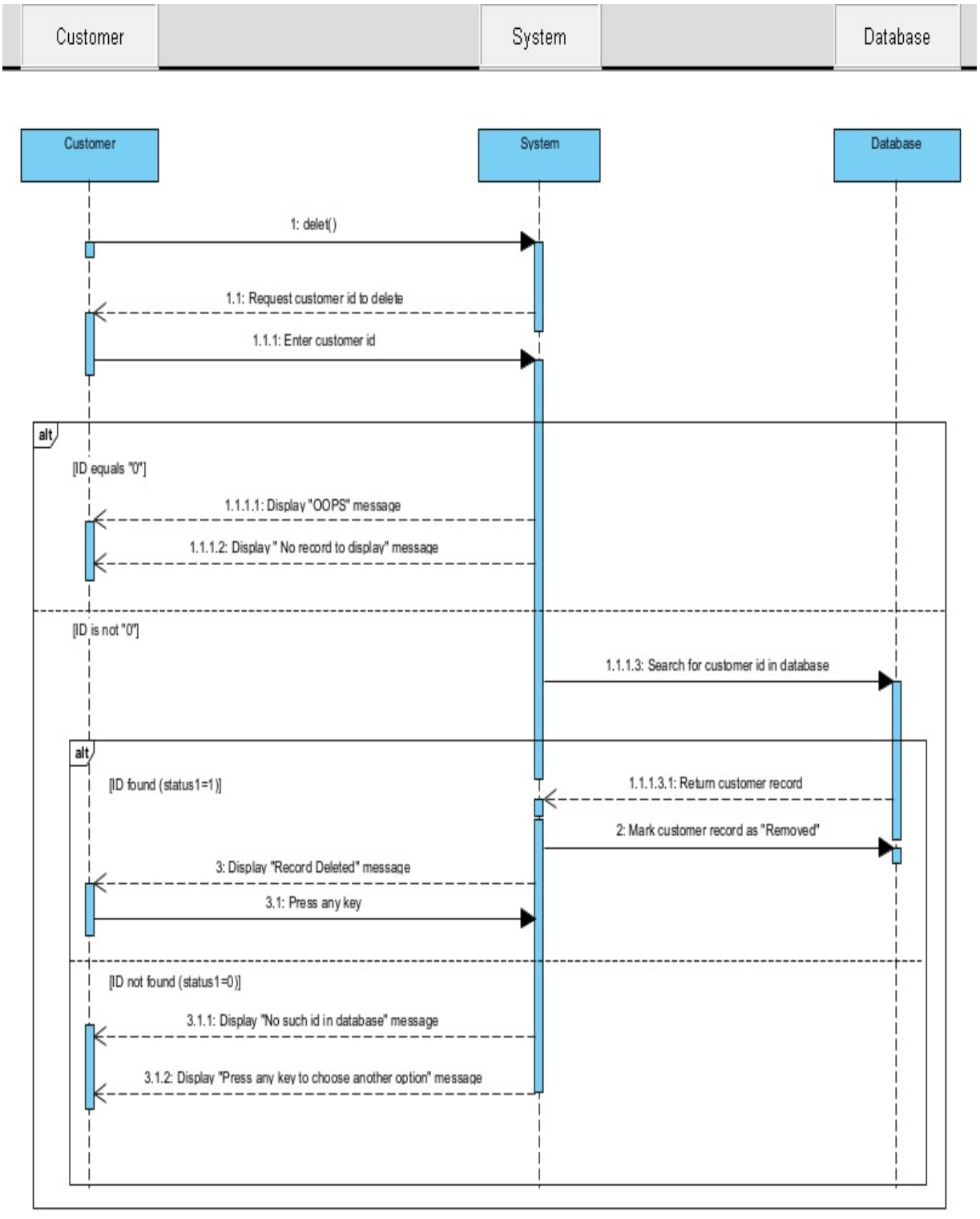
- **Purpose:**
  A Sequence Diagram is a type of UML (Unified Modeling Language) diagram that models the flow of interactions between system components or objects over time**.**
  Customer place order
  Customer Edit order
  Customer Delete order

Customer:

- **Place Order**

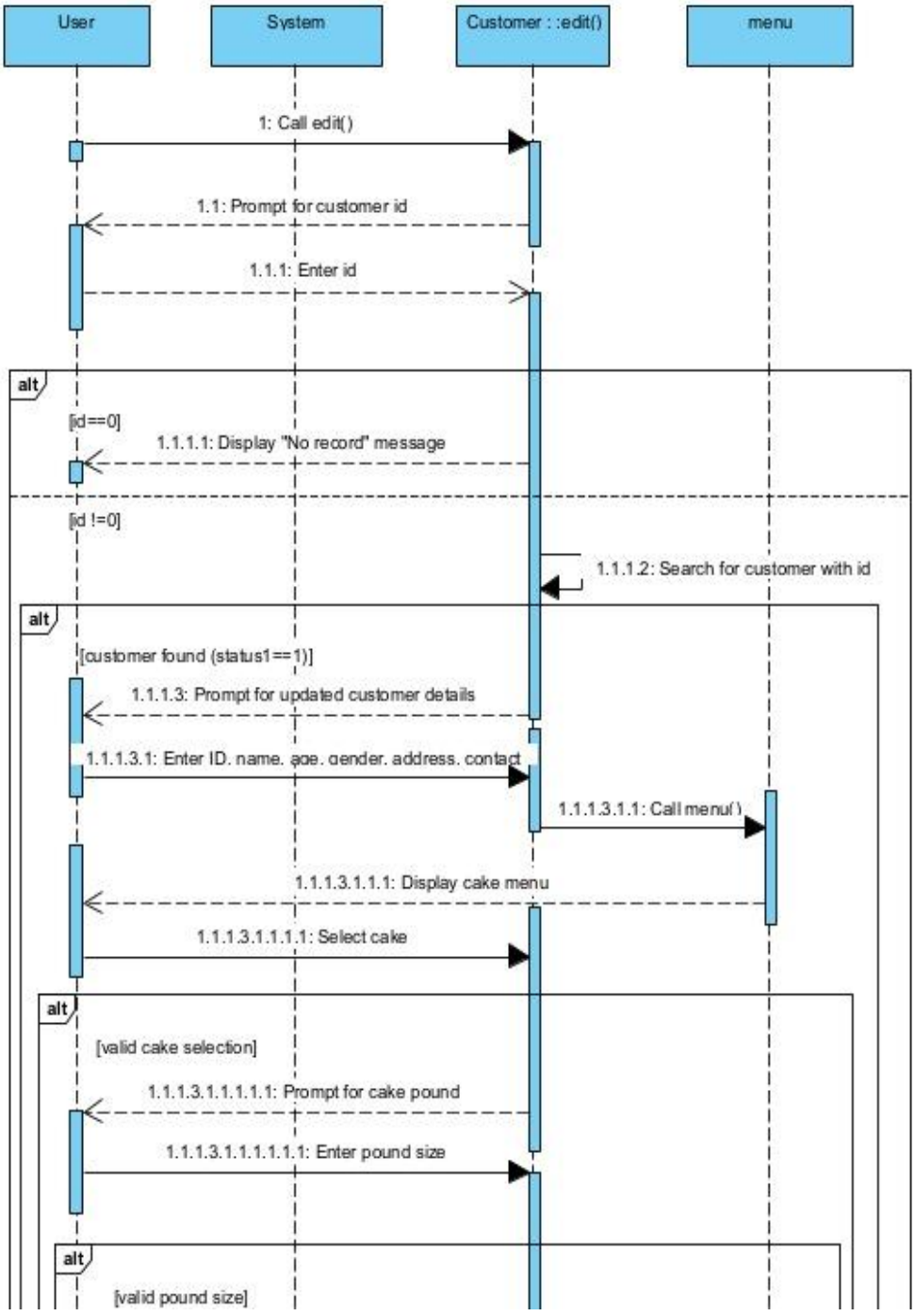


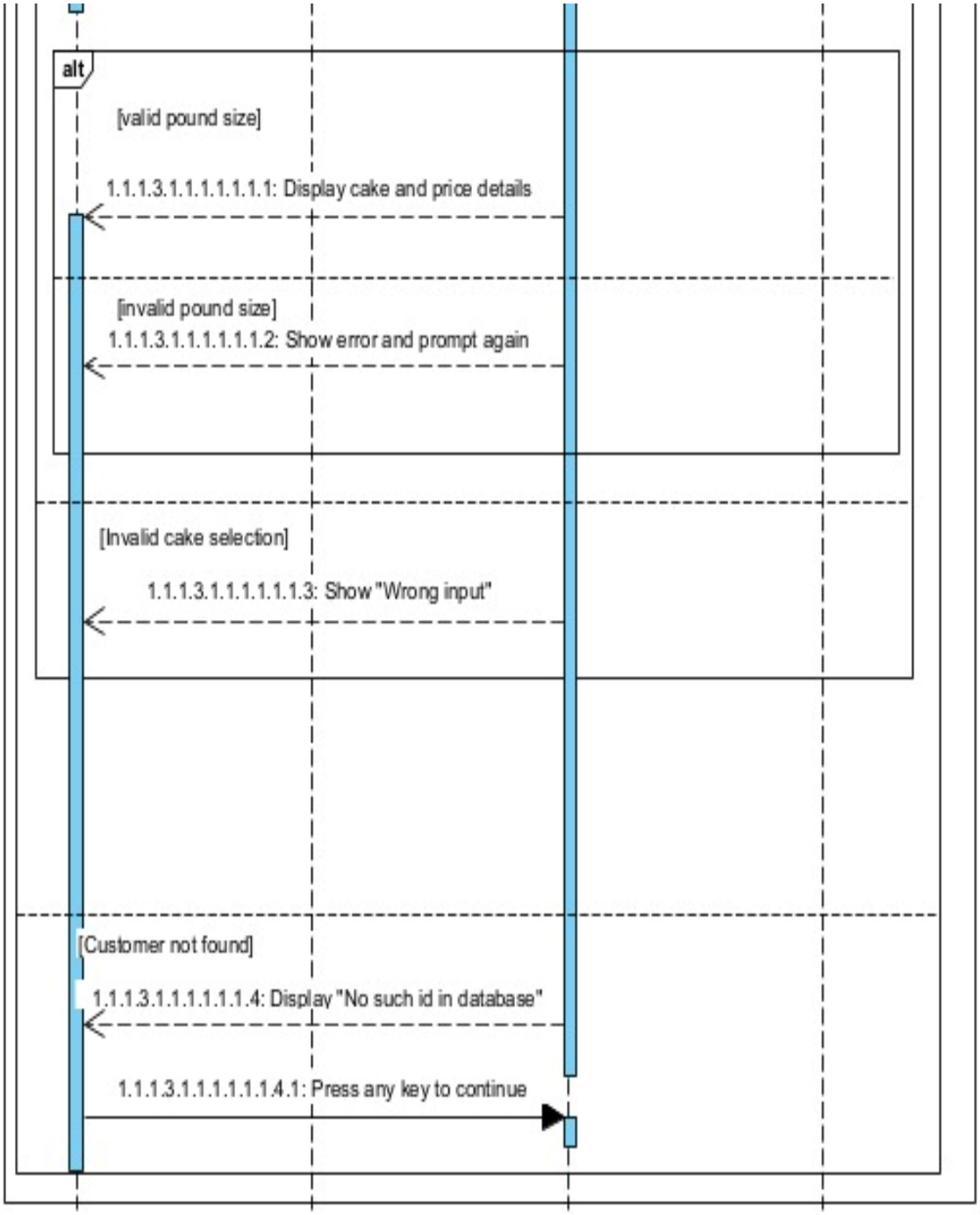
- **Delete Order**



- **Edit Order:**

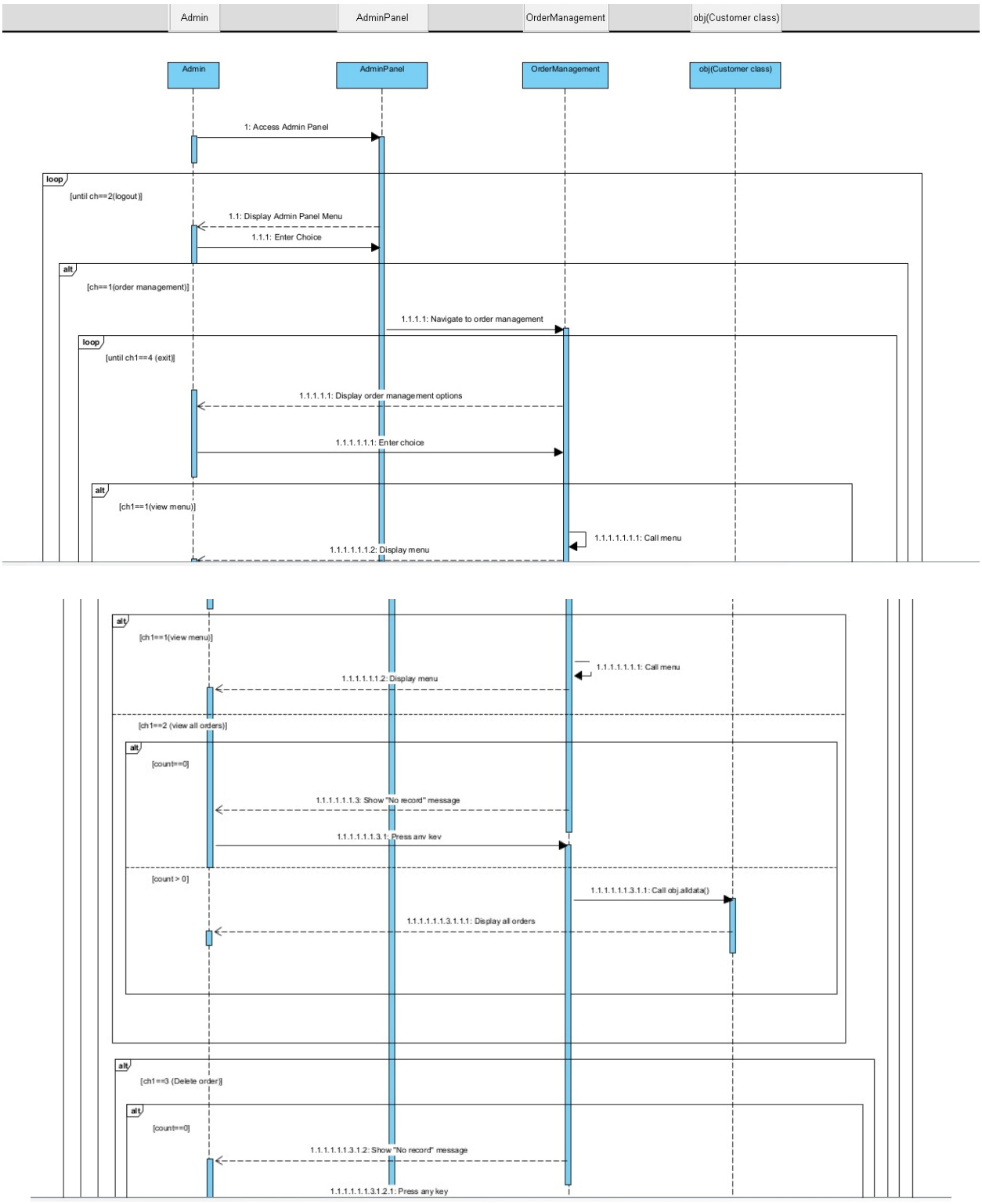

User
System
Customer :...
menu

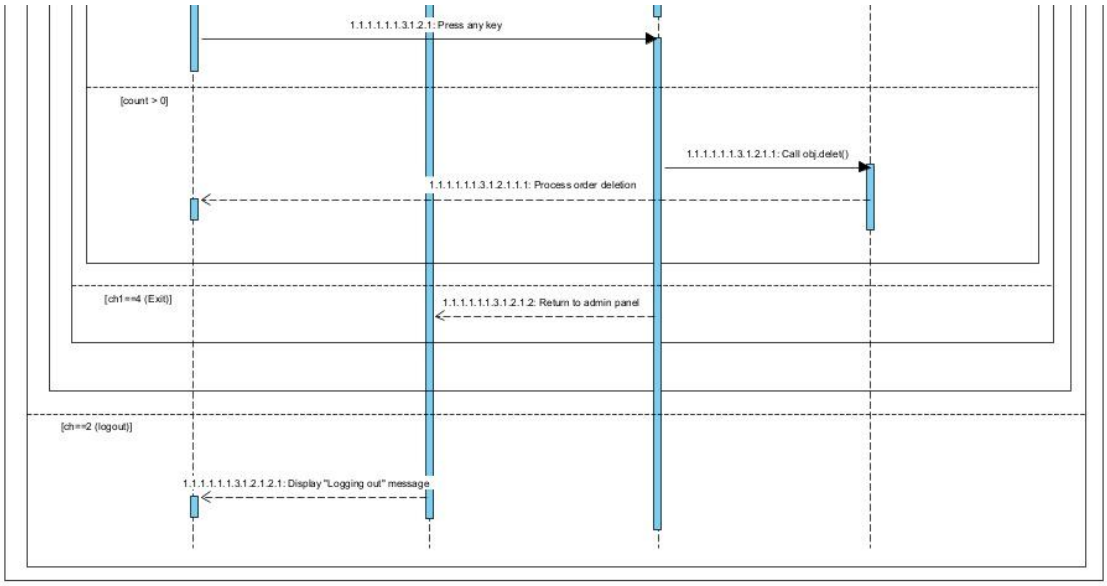

User
System
Customer : :edit()
menu
1: Call edit()
1.1: Prompt for customer id
1.1.1: Enter id
alt
[id==0]
1.1.1.1: Display "No record" message
[id !=0]
1.1.1.2: Search for customer with id
alt
[customer found (status1==1)]
1.1.1.3: Prompt for updated customer details
1.1.1.3.1: Enter ID. name. age. gender. address. contact
1.1.1.3.1.1: Call menu( )
1.1.1.3.1.1.1: Display cake menu
1.1.1.3.1.1.1.1: Select cake
alt
[valid cake selection]
1.1.1.3.1.1.1.1.1: Prompt for cake pound
1.1.1.3.1.1.1.1.1.1: Enter pound size
alt
[valid pound size]


alt
[valid pound size]
1.1.1.3.1.1.1.1.1.1.1: Display cake and price details
[invalid pound size]
1.1.1.3.1.1.1.1.1.1.2: Show error and prompt again
[Invalid cake selection]
1.1.1.3.1.1.1.1.1.1.3: Show "Wrong input"
[Customer not found]
1.1.1.3.1.1.1.1.1.1.4: Display "No such id in database"
1.1.1.3.1.1.1.1.1.1.4.1: Press any key to continue

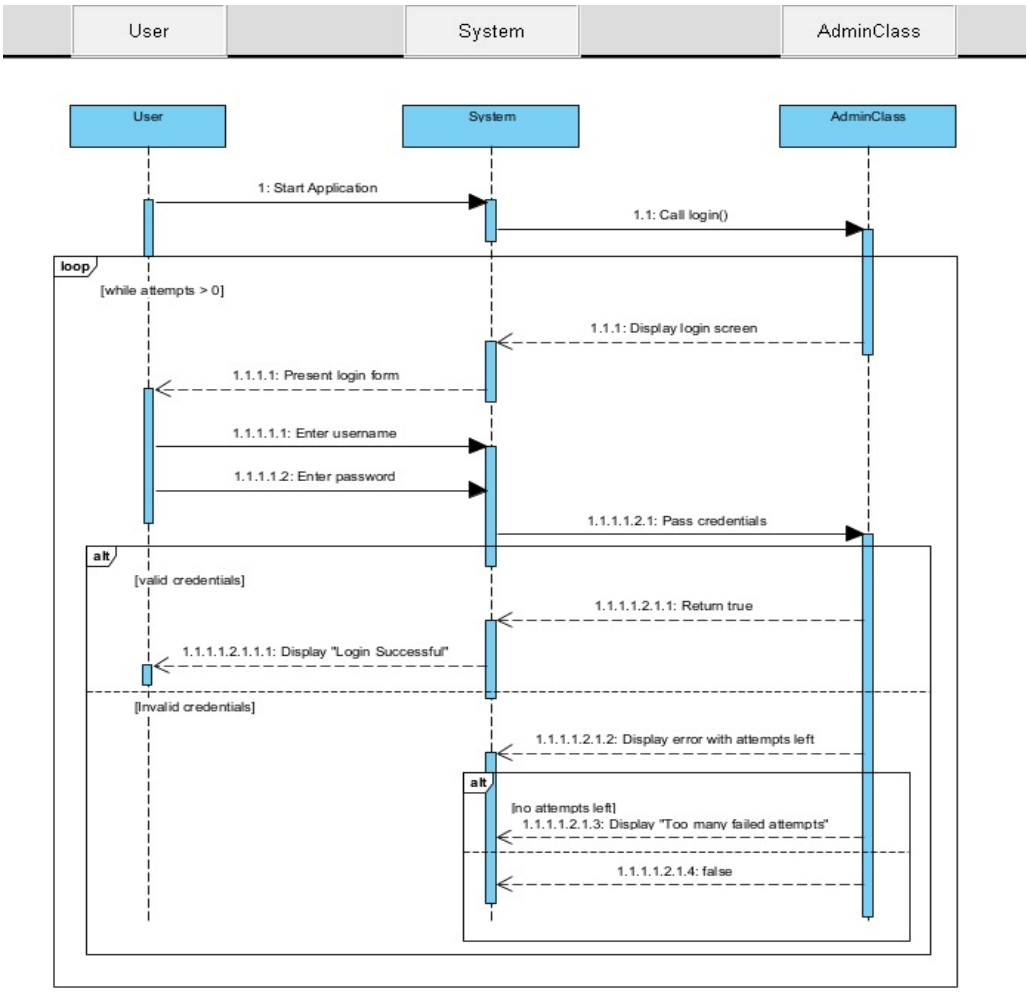
Admin

- **Admin View Menu:**

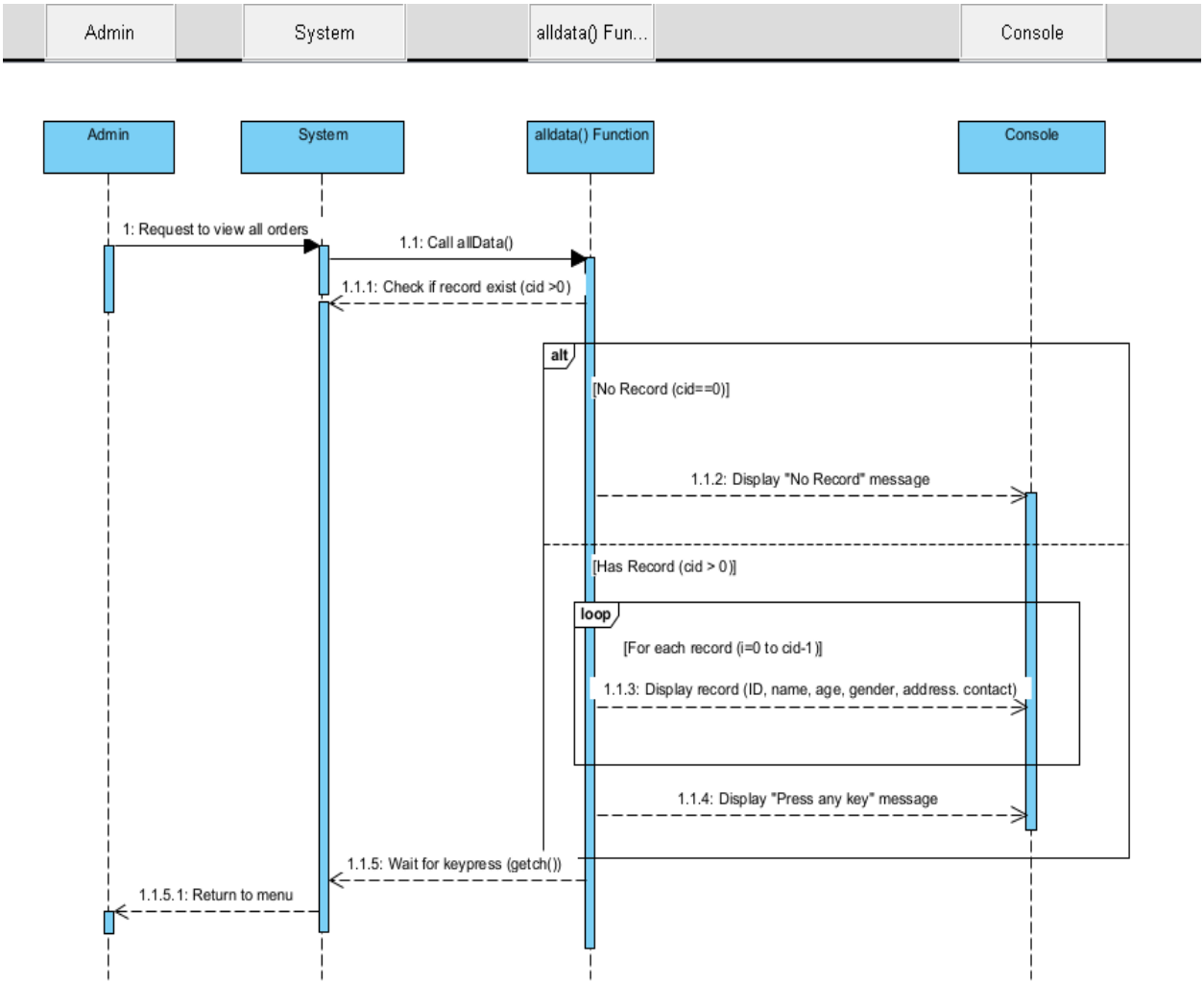



1.1.1.1.1.1.3.1.2.1: Press any key
[count > 0]
1.1.1.1.1.1.3.1.2.1.1: Call obj.delet()
1.1.1.1.1.1.3.1.2.1.1.1: Process order deletion
[ch1==4 (Exit)]
1.1.1.1.1.1.3.1.2.1.2: Return to admin panel
[ch==2 (logout)]
1.1.1.1.1.1.3.1.2.1.2.1: Display "Logging out" message

- **Admin Login:**


User
System
AdminClass
User
System
AdminClass
1: Start Application
1.1: Call login()
loop
[while attempts > 0]
1.1.1: Display login screen
1.1.1.1: Present login form
1.1.1.1.1: Enter username
1.1.1.1.2: Enter password
1.1.1.1.2.1: Pass credentials
alt
[valid credentials]
1.1.1.1.2.1.1: Return true
1.1.1.1.2.1.1.1: Display "Login Successful"
[Invalid credentials]
1.1.1.1.2.1.2: Display error with attempts left
alt
[no attempts left]
1.1.1.1.2.1.3: Display "Too many failed attempts"
1.1.1.1.2.1.4: false

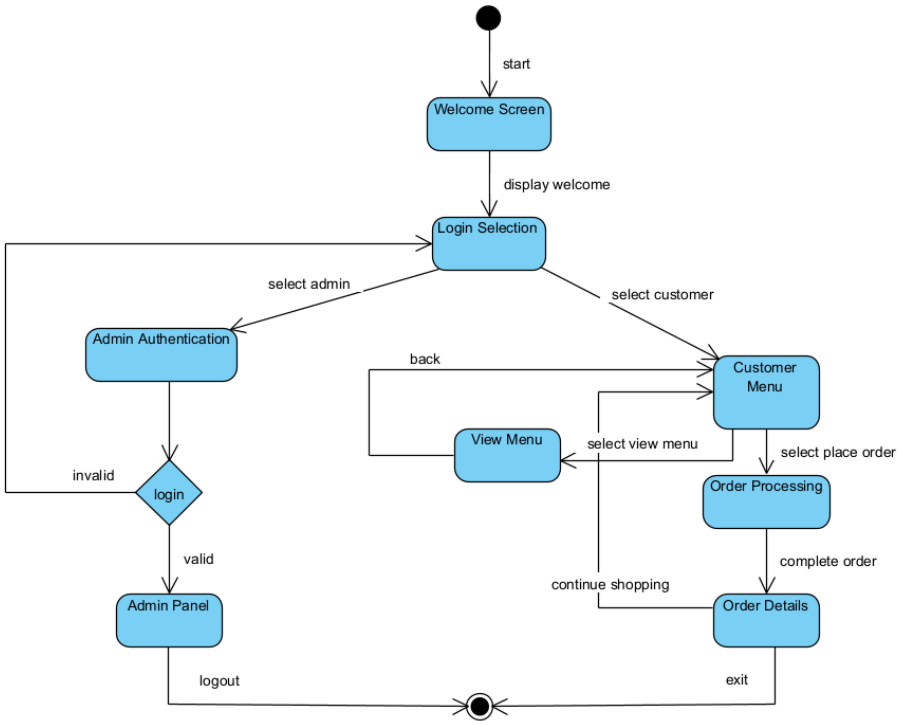
- **Admin View All Order:**



Design Viewpoint 4: State Dynamic View

State Machine Diagram

- **Purpose:**
  To show the different states an object goes through during its lifecycle, and what events trigger state changes.



Design Viewpoint 5: Data View

ER Diagram

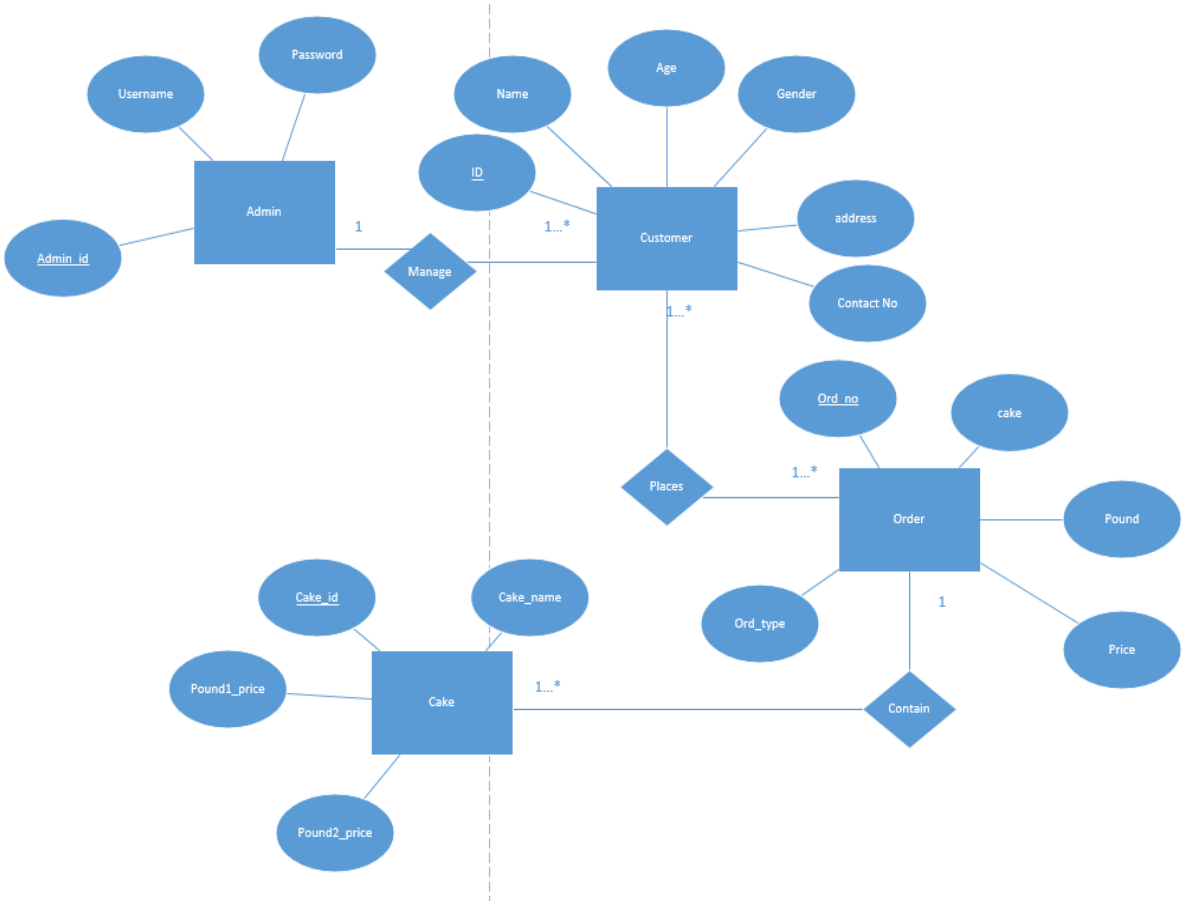
Purpose:
To model the database structure by showing entities (tables), their attributes**,** and relationships between them.

Entities:

- **Admin**
  Admin_id
  Username
  Password
- **Customer**
  Id
  Name
  Age
  Gender
  Address
  Contact_No
- **Order**
  Ord_No

Cake
Pound
Price
Ord_Type

- **Cake**
  Cake_id
  Cake_Name
  Pound1_Price
  Pound2_Price

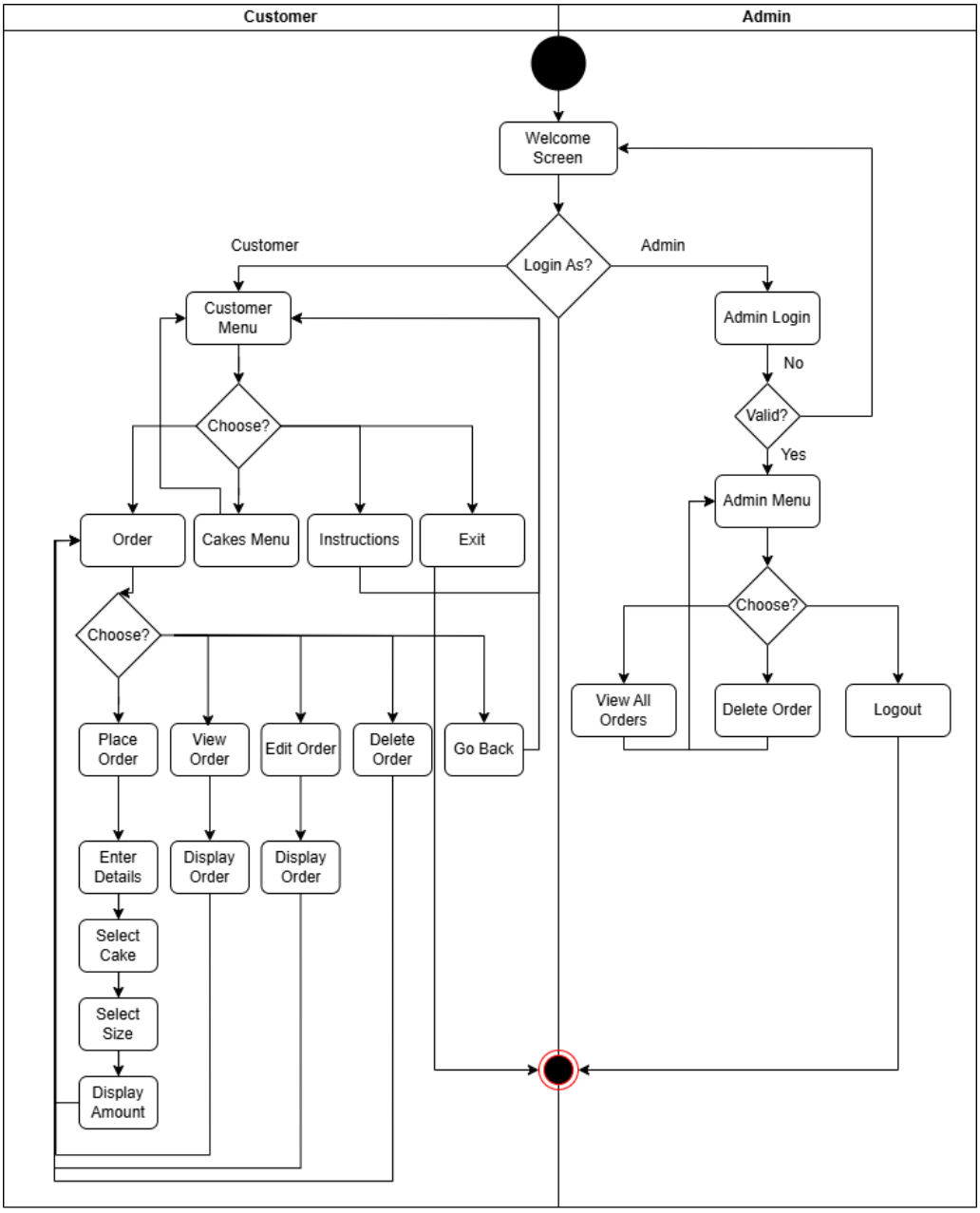


Design Viewpoint 6: Process View

Activity Diagram

Purpose:

Activity Diagrams show how a system behaves dynamically over time

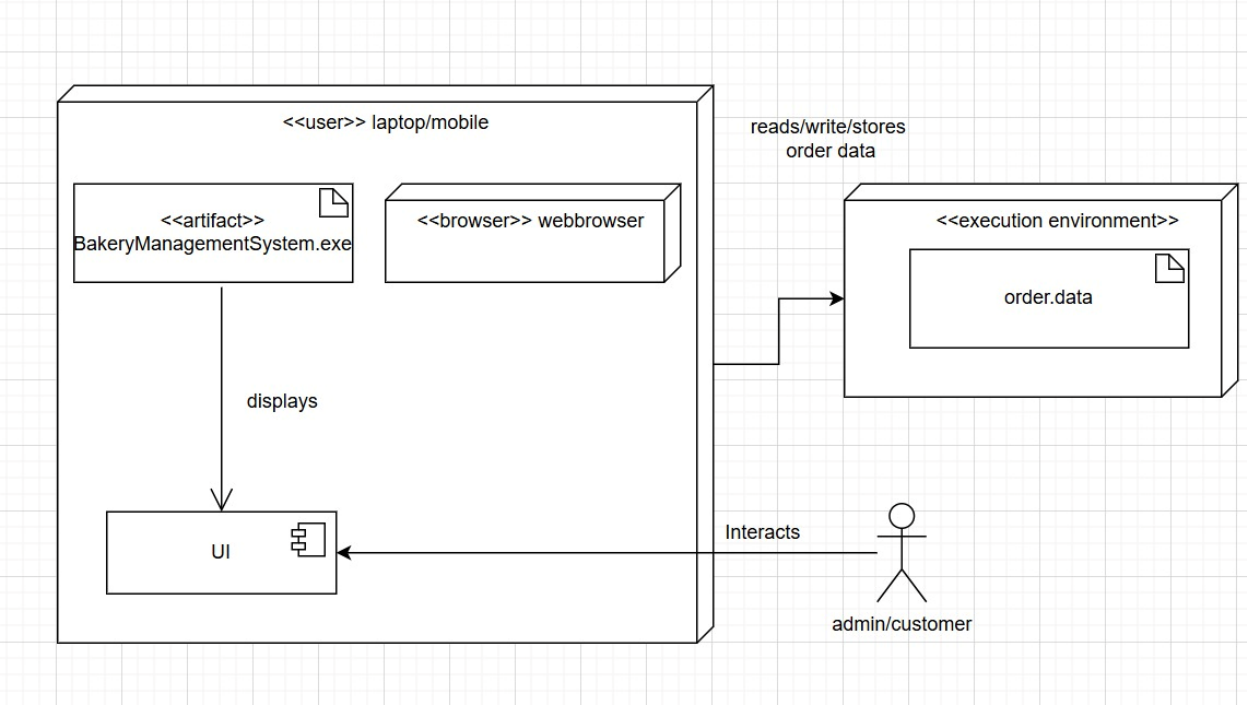


Design Viewpoint 7: Physical View

Deployment Diagram

Purpose:

Shows the physical arrangement of hardware (nodes) and the software components deployed on them, illustrating how a system is distributed across infrastructure.

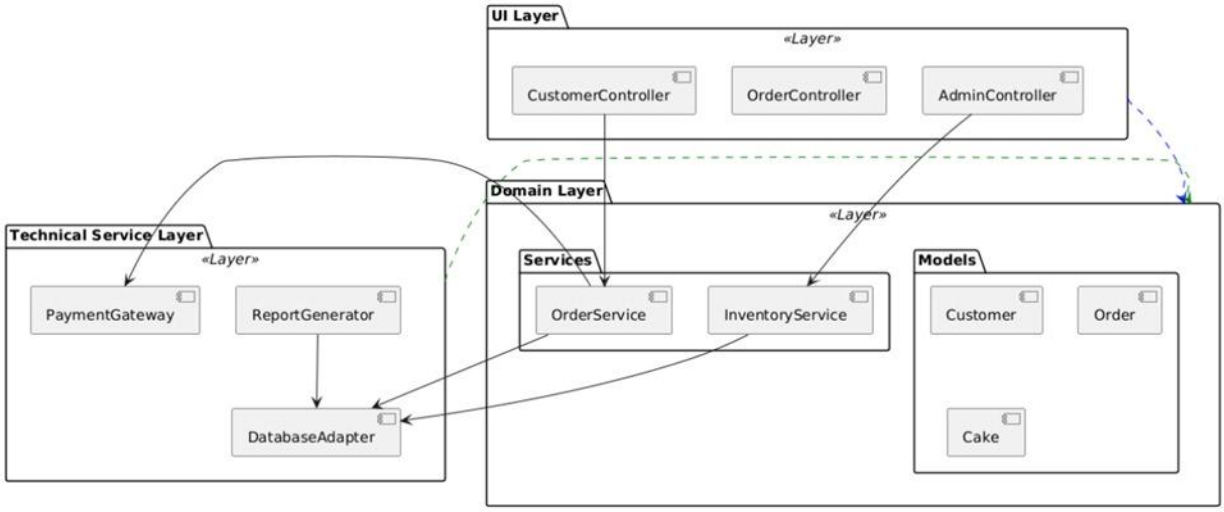


Design Viewpoint 8: Dependency View

Package Diagram

Purpose:

Package diagram illustrates the organization and dependencies of various packages or modules in a system, highlighting their relationships and structure.

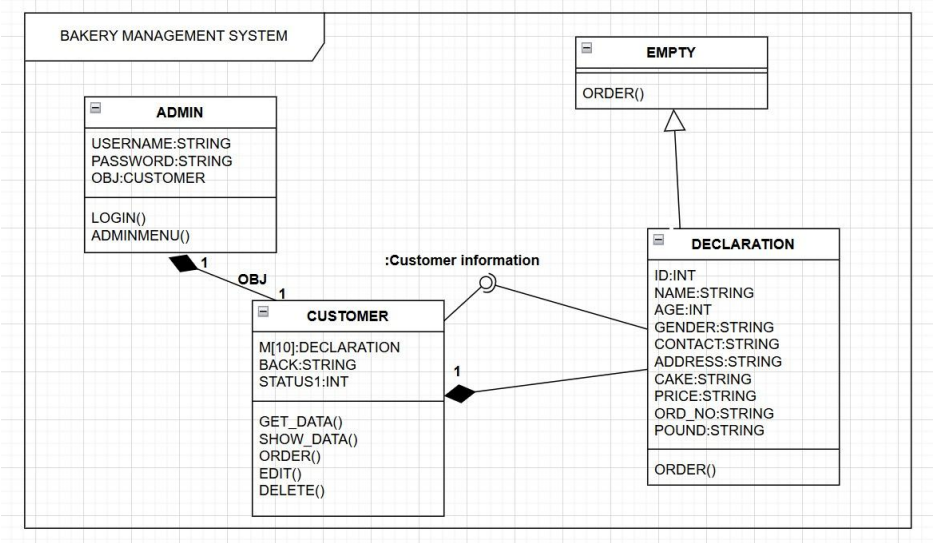


Design Viewpoint 9: Structure View

Composite Structure Diagram

Purpose:

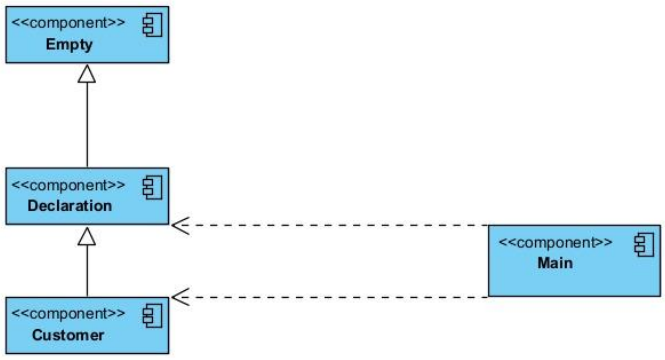
Depicts the internal structure of a class or component, showing its parts, their relationships, and how they collaborate to achieve the component's functionality.



Design Viewpoint 10: Composition

Component Diagram

- **Purpose:**
  To show the high-level structure of the software system as a set of components and how they interact



Design Viewpoints

Summary Of Design Viewpoints

Diagram	Viewpoint	Focus
Usecase Diagram	Use Case View	Shows interaction between system and users/services.
Class Diagram	Logical View	Describes static structure of classes and object organization.
Sequence Diagram	Interaction View	Models object communication over time.
State Diagram	State Dynamic View	Represents dynamic changes in object states.
ER Diagram	Data View	Models persistent data, entities, and their relationships.
Activity Diagram	Process View	Workflow, business logic, control flow
Deployment Diagram	Physical View	Infrastructure, hardware mapping
Package Diagram	Dependency View	Grouping of model elements and dependencies
Composite Structure Diagram	Structure View	Internal structure of classes/components
Component Diagram	Composition View	Represents modular parts of the system and their interfaces.