**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](https://cengproject.cankaya.edu.tr/wp-content/uploads/sites/10/2017/12/SDD-ieee-1016-2009.pdf)

**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**

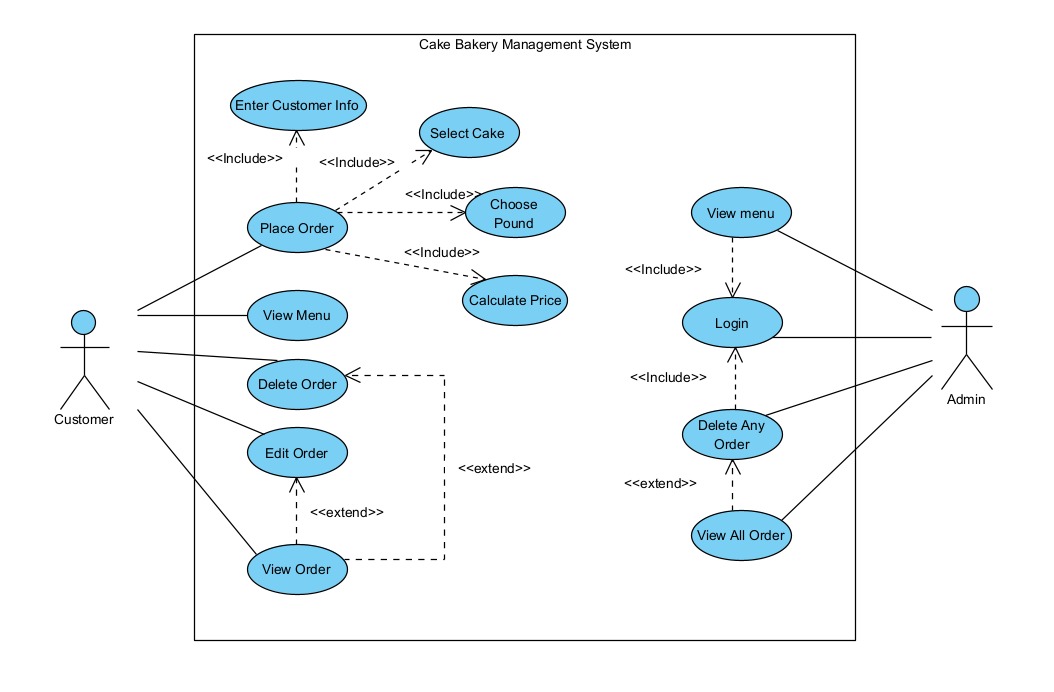
**Use Cases**:

**Customer**

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

**Admin**

* Login
* View Menu
* View all Order



**Design Viewpoint 2: Logical View**

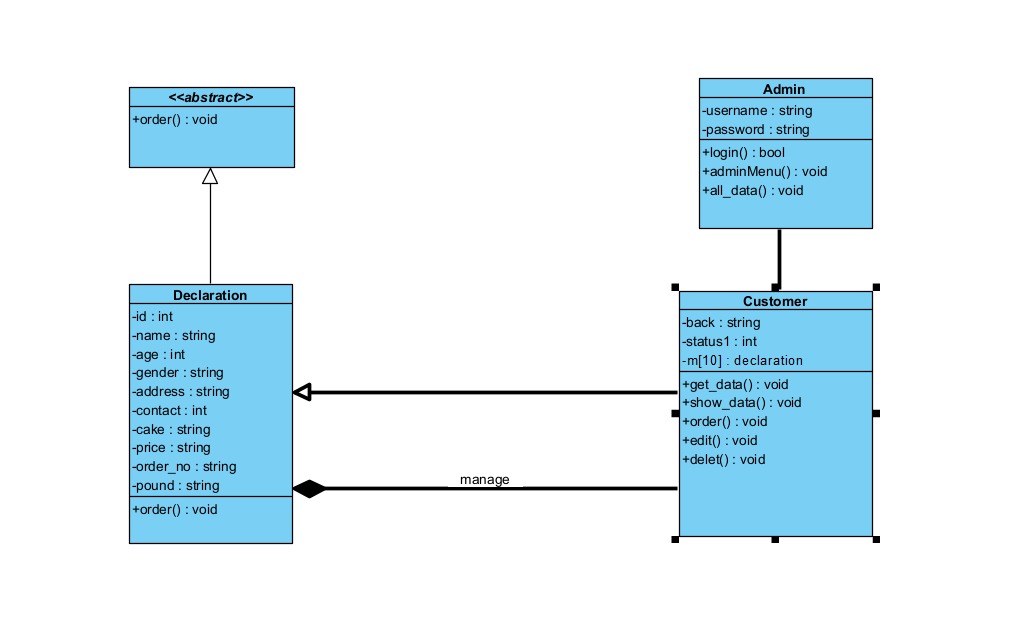
* **Class Diagram**:

**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.**

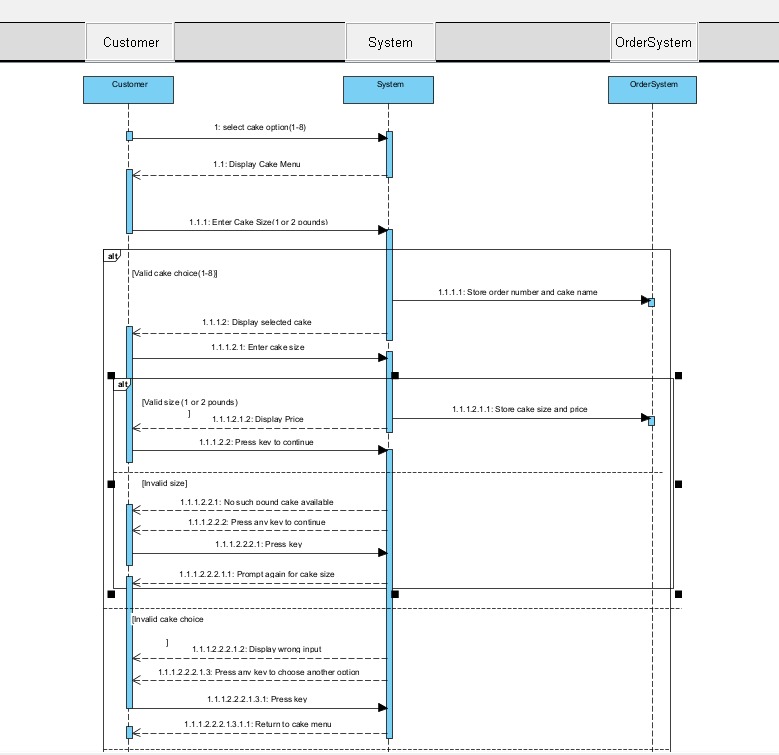
Customer place order

Customer Edit order

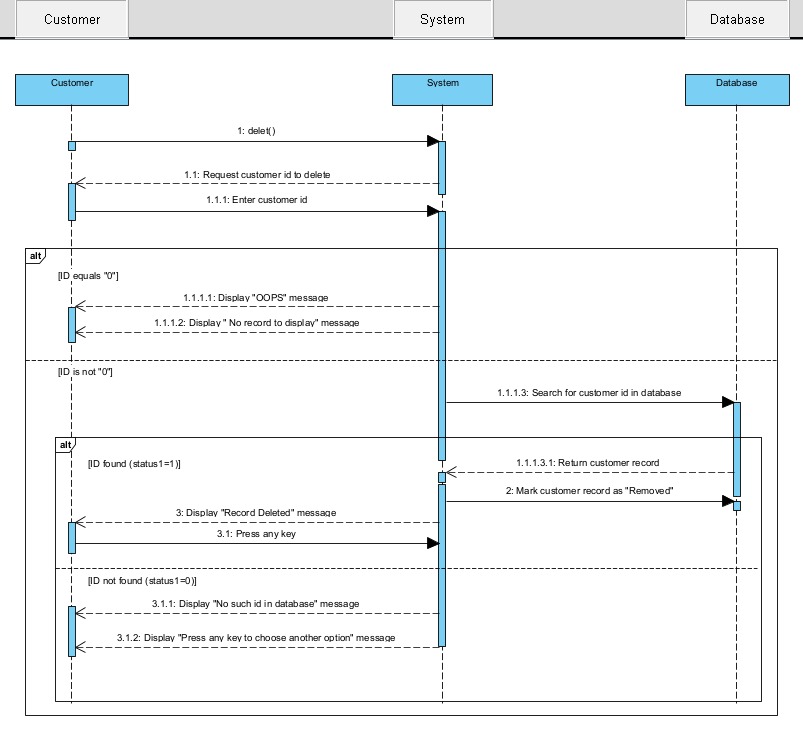
Customer Delete order

**Customer:**

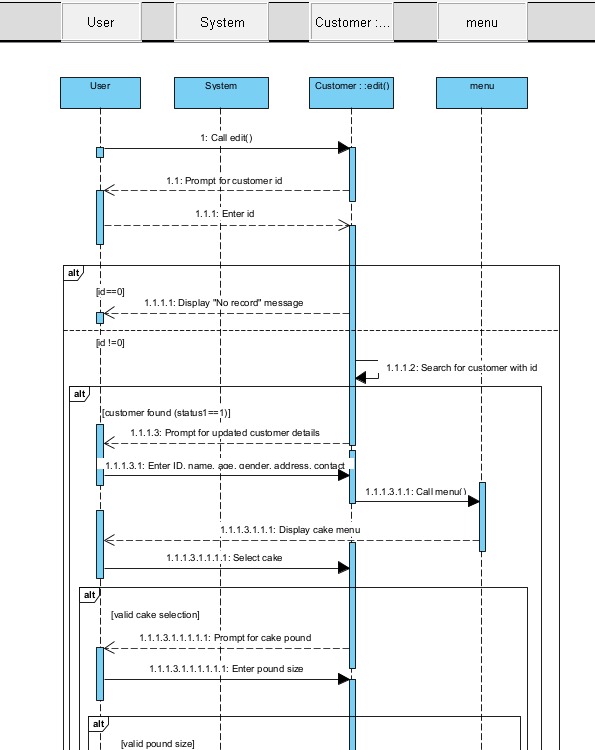
* **Place Order**

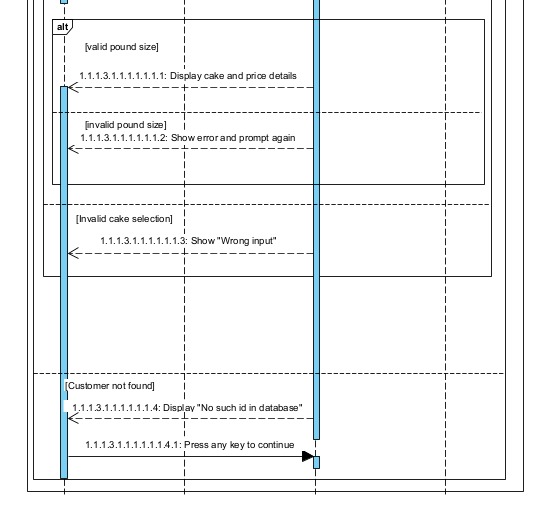


* **Delete Order**

****

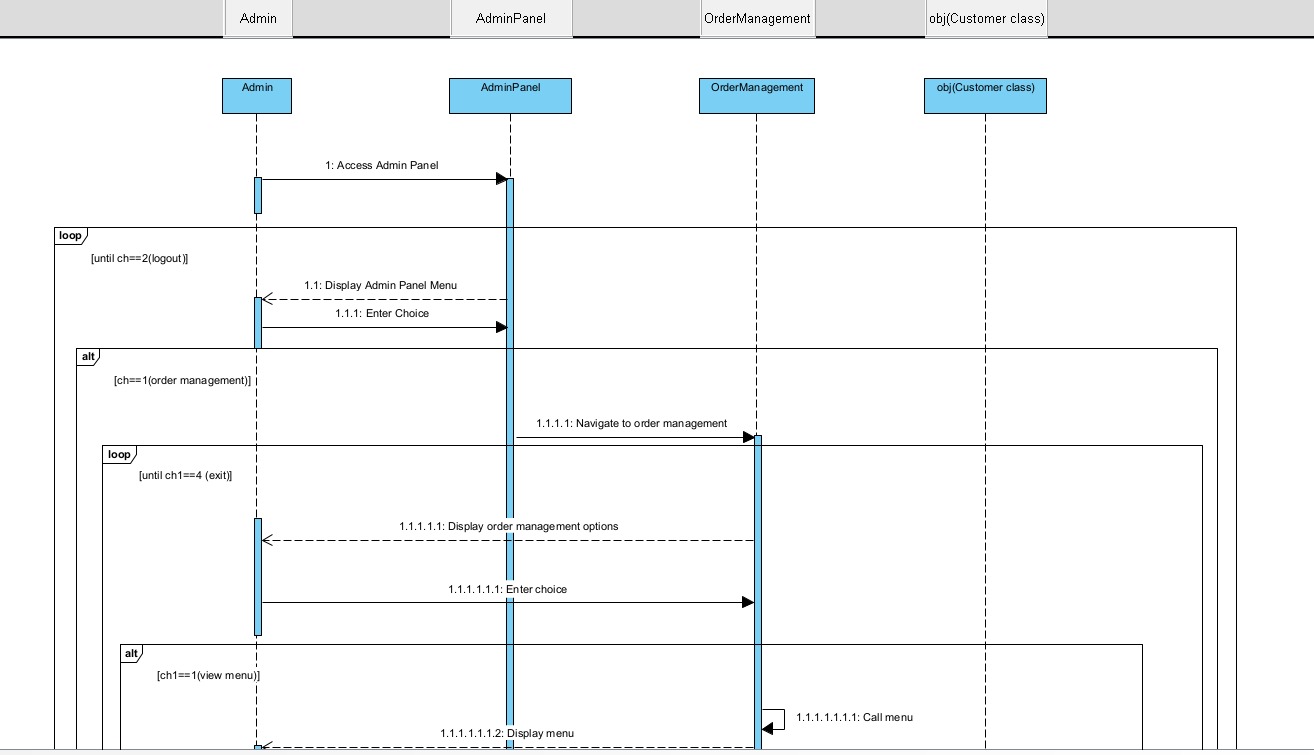
* **Edit Order:**

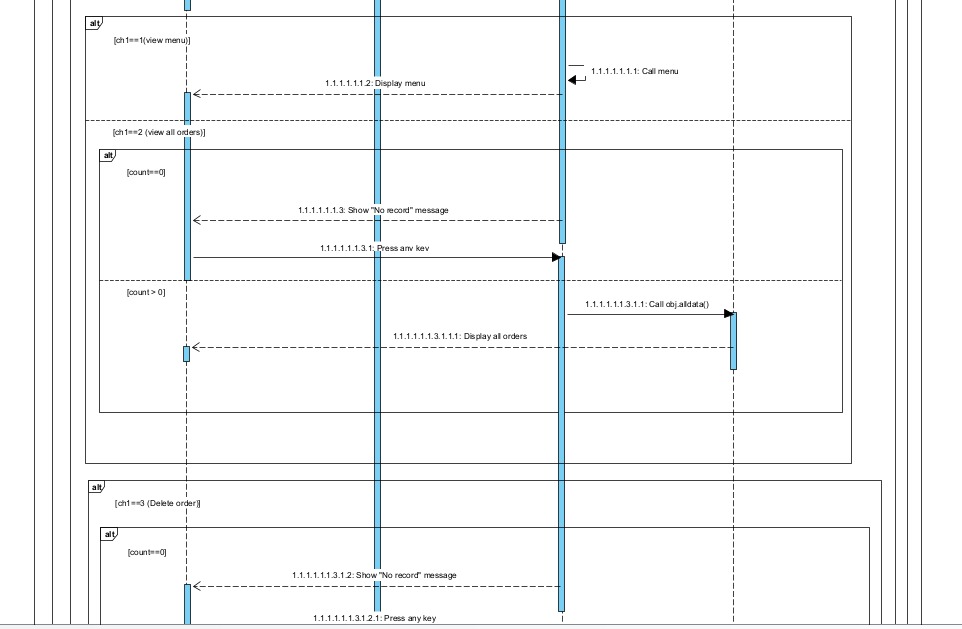
****

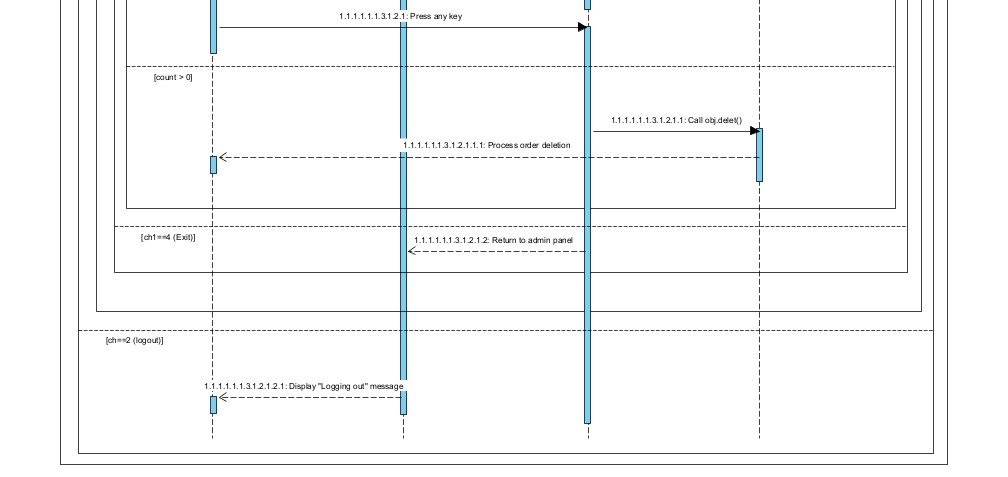
****

**Admin**

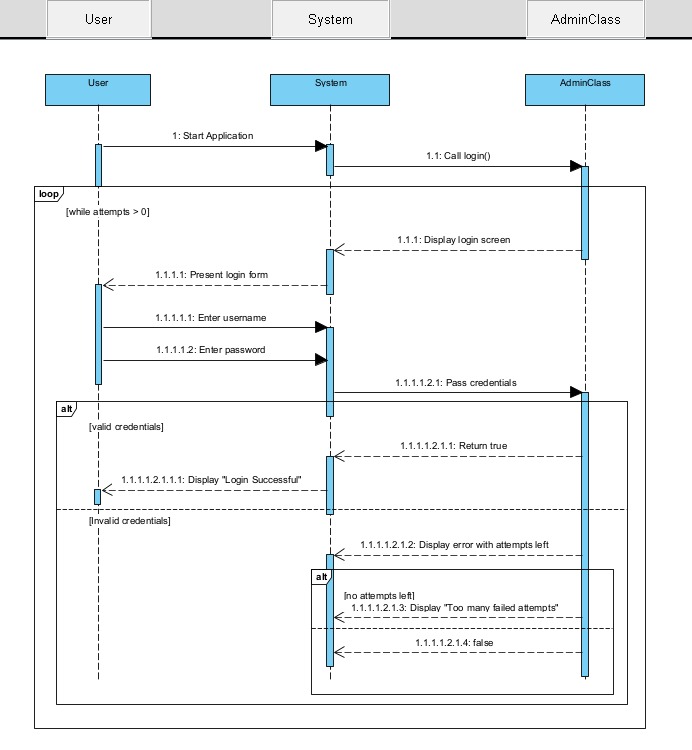
* **Admin View Menu:**

****

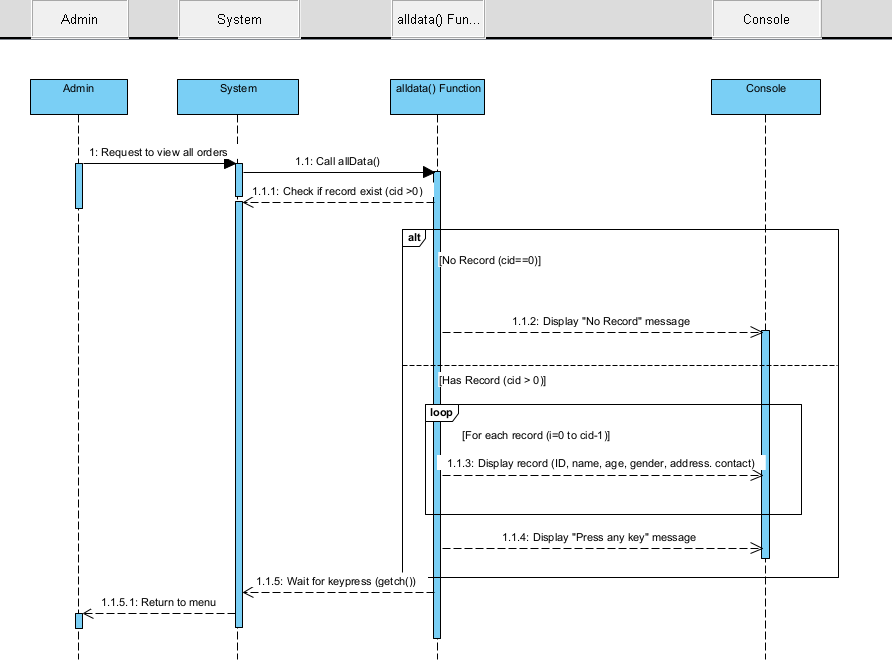
****

****

* **Admin Login:**

****

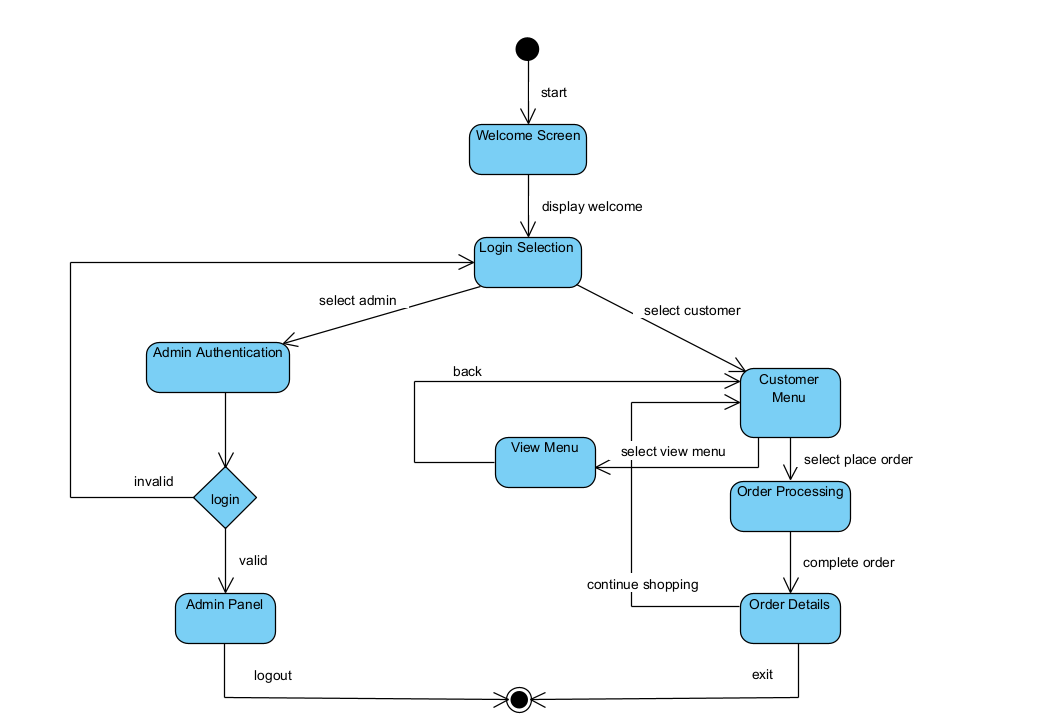
* **Admin View All Order:**

****

**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**

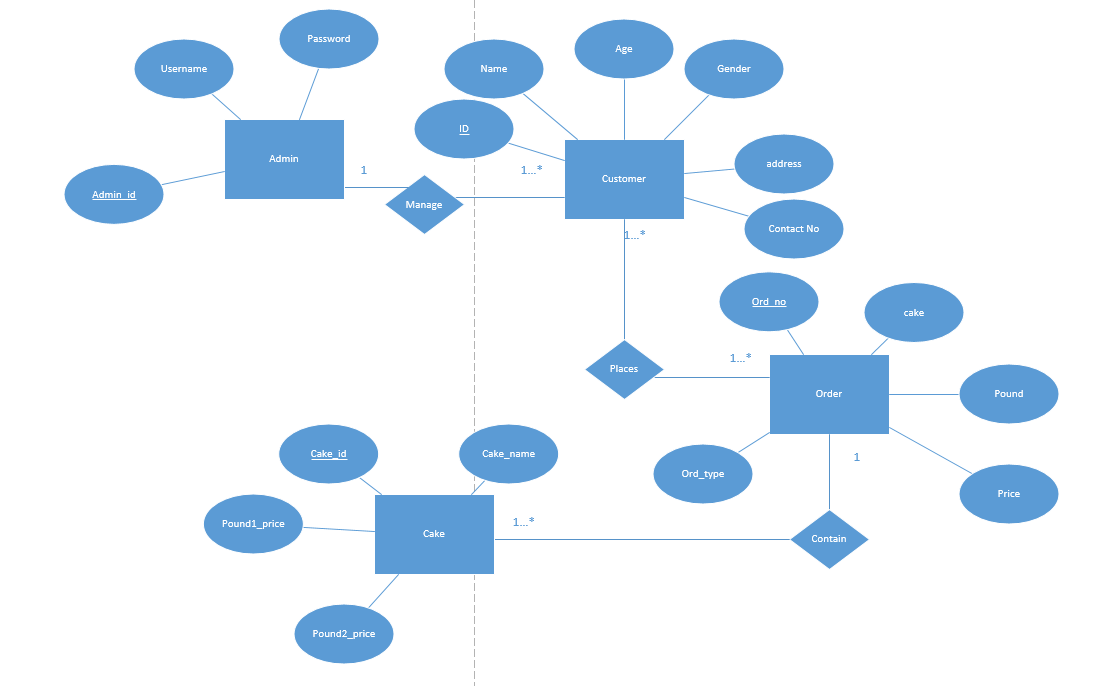
* **Cake**

**Cake\_id**

**Cake\_Name**

**Pound1\_Price**

**Pound2\_Price**

****

**Design Viewpoint 6: Physical Composition View**

**Deployment Diagram**

**Design Viewpoint 7: Dependency View**

**Package Diagram**

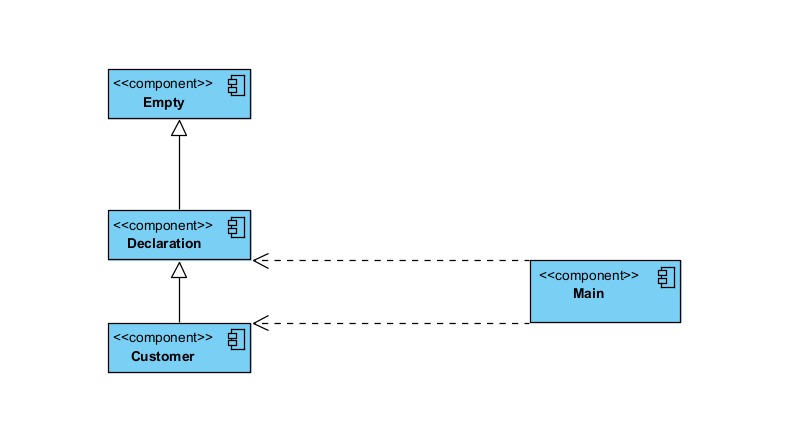
**Design Viewpoint 8: Structure View**

**Composite Structure Diagram**

**Design Viewpoint 9: Composition**

**Component Diagram**

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

****

**Design Viewpoint 10: Security View**

**Design Viewpoint 11: Performance View**

**Design Viewpoint 12: Scalability View**

**Design Viewpoints**

**Summary Of Design Viewpoints**

|  |  |  |
| --- | --- | --- |
| **Diagram** | **Viewpoint** | **Reason** |
| 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. |
| Component Diagram | Composition View | Represents modular parts of the system and their interfaces. |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |