****

**Ain Shams University**

**Faculty of Engineering**

**Computer and Systems Engineering Department**

**CSE 321: Software Engineering – 3rd Year CSE – 1st Semester 2016/2017**

**Submitted by:**

|  |  |  |
| --- | --- | --- |
| Name | Section | Student ID |
| Ahmed Mohammed Hesham El-Moselhy | 1 | 1300208 |
| Ahmed Sameh Mohammed Soliman | 1 | 1300114 |
| Ahmed Mohsen Abd El-Ba’eth | 1 | 1300177 |
| Ahmed Nasser Fathy Mosalam | 1 | 1300230 |
| Hassan Saeed Hassan Ahmed | 1 | 1300466 |
| Mohammed Mansour Abo-Bakr | 3 | 14x0008 |

Table of Content

1. Abstract …………………………………………… 3
2. Introduction …………………………………………… 4
   1. Purpose …………………………………………… 4
   2. List of definitions …………………………………………… 4
   3. Scope …………………………………………… 4
   4. Overview …………………………………………… 4
3. General Description …………………………………………… 5 - 6
   1. Product perspective …………………………………………… 5
   2. General Capabilities …………………………………………… 5
   3. General Constrains …………………………………………… 5
   4. User Characteristics …………………………………………… 5
   5. Environment description …………………………………………… 6
   6. Assumptions and dependencies ………………………………… 6
   7. Other Resources needed ..…..…………………………… 6
4. System Requirements …………………………………………… 7
   1. Functional Requirements …………………………………… 7
   2. Non-functional Requirements .. ………………………………… 7
5. Use-Case Diagram ……………………………… 8
6. Narrative Description/Swim lane Diagram of Use Cases ……………… 9 - 10
7. Data Model …………………………………………… 11
8. Requirements Validation …………………………………………… 12
9. Class Model …………………………………………… 13
10. State Diagram …………………………………………… 14
11. Interaction Diagram …………………………………………… 15 - 16
12. Detailed Class Diagram …………………………………………… 17
13. Data Model Design ………………………………………… 18
14. User Interface Design .…………………………………………… 19
15. Client-Object Relation Diagram ..…………………………………………… 20
16. Detailed Design ……………………………………………… 21 - 22
17. Testing ……………………………………………… 23
18. Estimated Project Cost ……………………………………………… 24

**0.0 Abstract:**

Developing a Software system for a car trading company working on .NET framework using C# programming language.

The application is aiming to organize the whole work of the company such as:

* Facilitate and make the responsibility of the various departments obvious.
* Controlling the company’s Database of employees.
* Facilitate the process of employees’ evaluation and salaries.
* Stock Management.
* Organizing selling and purchase operation.

1. **Introduction**
   1. **Purpose:**

Giving a detailed description about the software system requirements, product description, software system implementation and system components, design and database.

* 1. **List of definitions:**
* UML: Unified modeling language
* EER: Enhanced Entity Relationship model
* KPI: Keep Performance Indication
  1. **Scope**

A car trading company system that works on .NET framework.

It is used for:

* Facilitate and make the responsibility of the various departments obvious.
* Controlling the company’s Database of employees.
* Facilitate the process of employees’ evaluation and salaries.
* Stock Management.
* Organizing selling and purchase operation.
  1. **Over View**

The system makes every department manager and its employees’ able to fulfil their job description and targets easily by an integrated software system that connect all the entities of the company.

In the following topics, the capabilities of the system, constrains, user characteristics and system requirements will be mentioned in details.

1. **General Description**
   1. **Product Perspective**

The system helps the departments of the company to fulfil their jobs, as every employee in each department has an account in the system which makes him able to access an available services related to his job description.

* 1. **General Capabilities**

The system provide many services for the employees depending on their department and job description, such as:

* HR Departments:
  + Determine the attendance of the employees.
  + Facilitate providing an evaluation of the employees.
  + Facilitate sending warnings or promotions.
  + Monitoring KPIs.
* Finance Department:
  + Calculating of net income.
  + Calculating of net outcome.
  + Facilitate the process of calculating and paying the employees’ salaries.
* Sales Department:
  + Organizing the Selling process for clients and agents.
* Stock Management Department:
  + Providing a deep stock management.
  + Control delivery of the sold cars from the stock to the client.
  1. **General Constrains**
* Stock Capacity
* Employees’ working time is 8 hours/day
* Employees working for 5 days a week, as Friday and Saturday is official vacations.
* Available Budget.
  1. **User Characteristics**

Every member in power of the company such as General Manager, Department Managers and employees is able to use the system with different authorization and services allowance as mentioned in section 2.3.

* 1. **Environment Description**

The software system is working on any .NET framework.

* 1. **Assumptions and Dependencies**
* The company is constructed of four department: Sales, Finance, HR and Stock Management.
* The company has 4 stocks, and every stock has a capacity of 100 car.
* Our clients are individuals and distributers.
* Stock Management Department is responsible for delivering cars to clients.
* Finance Department depends on the HR department reports on employees in the process of purchase salaries and bonuses.
* Stock Management Department depends on Sales Department in the process of delivering the cars to clients.
  1. **Other Resources needed**

No extra resources are needed, just a .NET framework is enough to work on the software.

1. **System Requirements**
   1. **Functional Requirements**

1.1 Customer should be able to place his order.

1.2 The system should record logins of employees

1.3 The system should check the availability of the order in the

stock.

1.4 The system should notify the employee in case order is not

available.

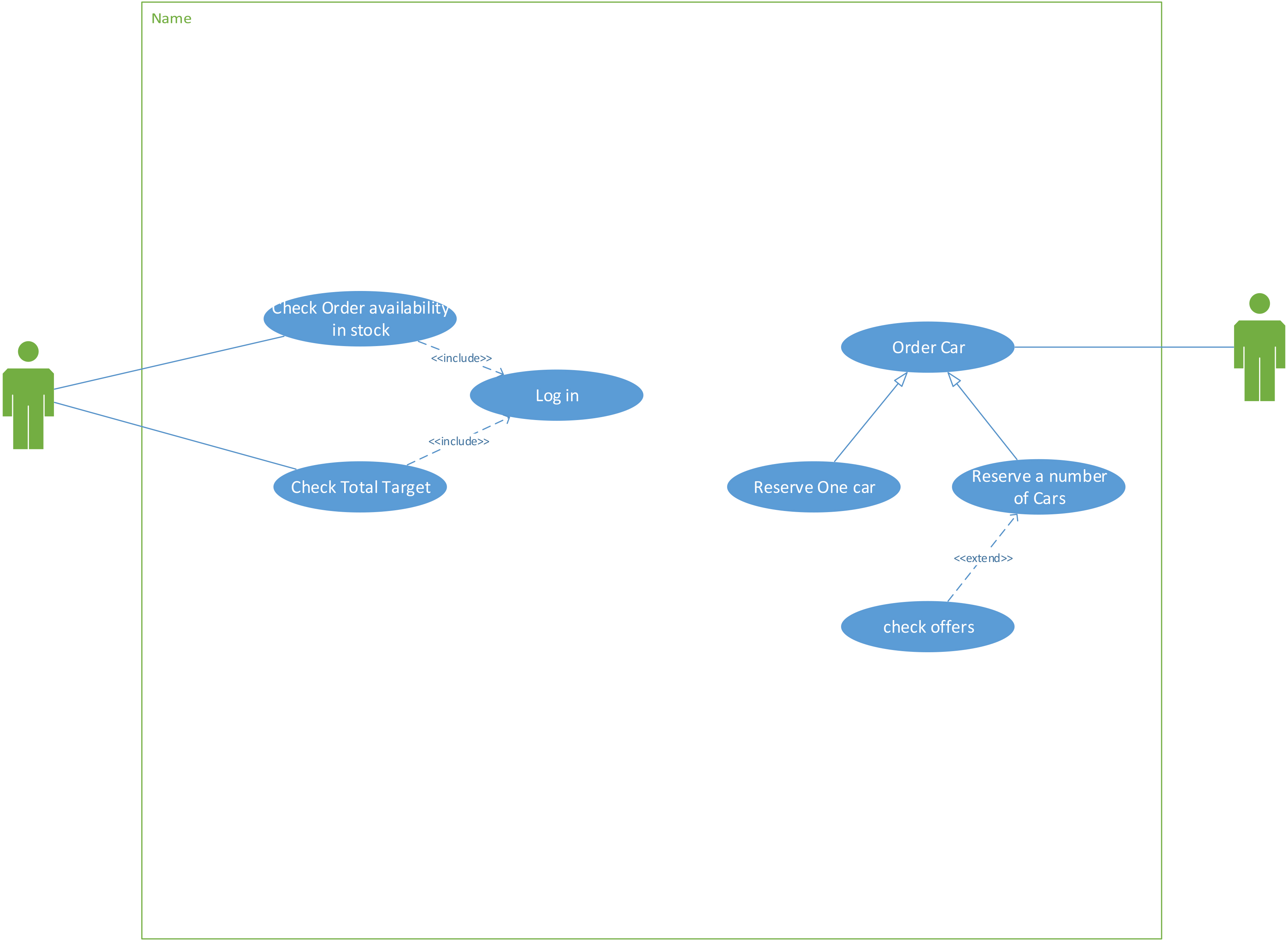
1.5 The system should check sales count of the employee regularly.

1.6 The system should compare number of sales to the target.

1.7 The system should assign Bonuses to employees if they satisfied their target.

* 1. **Non-Functional Requirements**
* The system should be reliable.
* The system should recover from breakdown in less than 2 minutes.
* The code should be written in C#
* The delivery date should not be assigned for more than 1 week away from order.
* The system obeys the laws and regulations of car trading.

1. **Use Case Diagram**

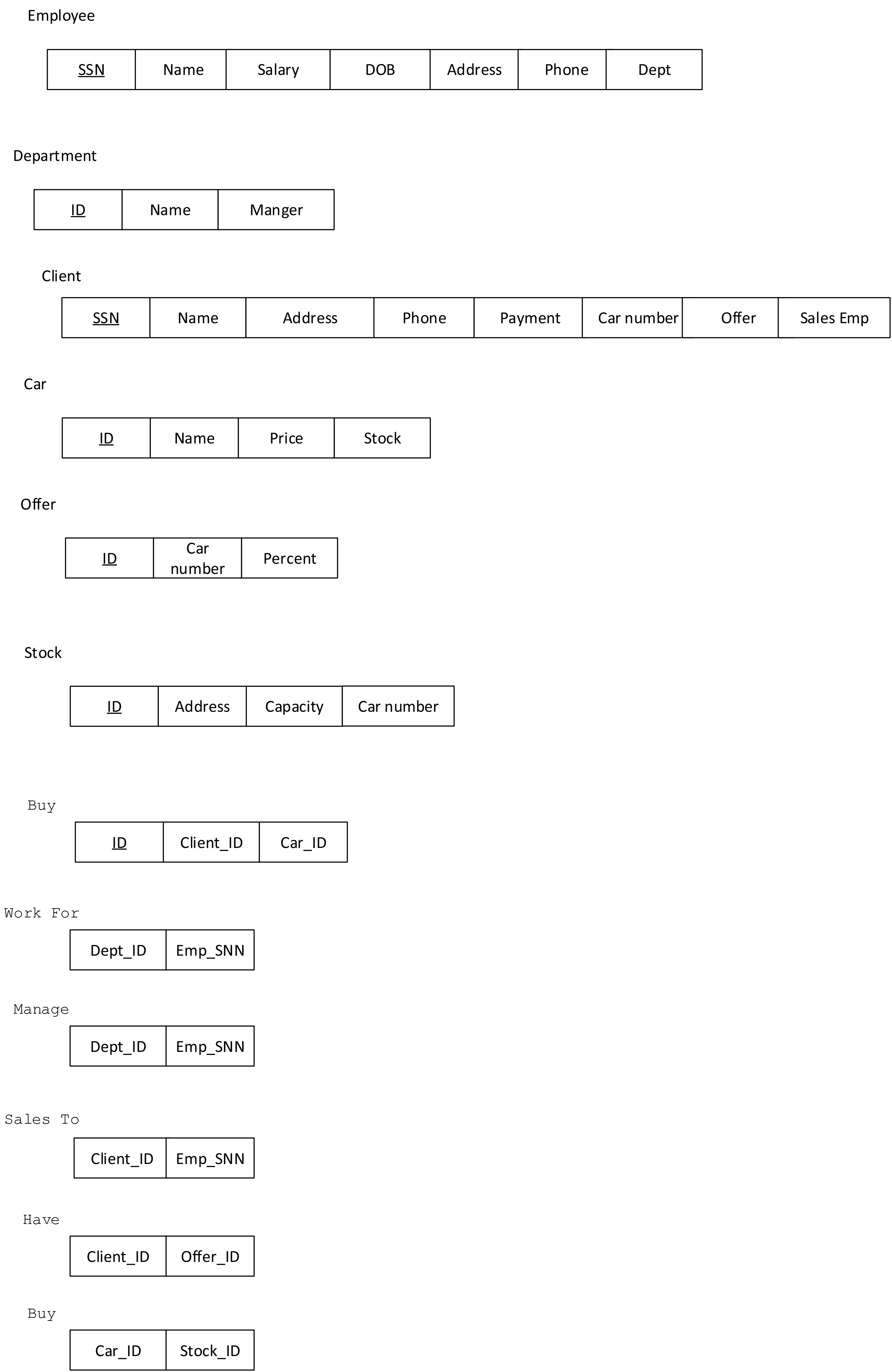


1. **Narrative Description/Swim lane Diagram of Use Cases**

|  |  |
| --- | --- |
| Use Case Name: | Order Car |
| Related Requirements: | Customer needs to order a specific number of cars and check offers. |
| Goals In Context : | 1. Customer Orders Car 2. System tells customer if available 3. System tells customer if there are any offers |
| Primary actors: | Customer, database and stock |
| Secondary: | Employee |
| Main Flow: | * 1. Customer asks the system to Reserve a car.   2. System asks customer about order details.   3. Customer fills a form about his details and order details.   4. System checks availability of the order. |
| Include::check offers | e. System checks if there are any offers. |
| Extensions: | * System sends message if order not available. * Failed orders must be recorded. |

|  |  |
| --- | --- |
| Use Case Name: | Checking Target |
| Related Requirements: | Employee should be able to check if he achieved target |
| Goals in context: | Assigning bonus to employees who achieved their target |
| Primary actors: | Employee, database and Target |
| Secondary actors: | None |
| Main flow: | 1. Employee delivers order successfully. 2. The system increments his sales Count by 1. 3. The system checks if sales count of employee equals target or more. 4. The system assigns Bonus to Employee. |
| Include::Check login credentials | 5. Employee Must login first |
| Extensions: | * Employee should be able to view his personal Target. * Employee should be able to check if he is assigned bonus or not. |

1. **Data Model:**



1. **Requirements Validation**

|  |  |
| --- | --- |
| **Req.** | **1.1 1.2 1.3 1.4 1.5 1.6 1.7** |
| **1.1** |  |
| **1.2** |  |
| **1.3** | **D** |
| **1.4** | **D D** |
| **1.5** | **D** |
| **1.6** | **D D** |
| **1.7** | **D R** |

1. **Class Model**

Concise emphasis:

The software System manages the sales of a car trading company.

Informal strategy:

This is a software for a car trading Company each department contains at least 5 employees. Any employee should be able to logging in to announce his attendance on which his salary could be calculated, the company has 4 stocks each with capacity 100 cars the system should know the available car types and numbers in the stock. Sales employee deals with the client which may be simple client or a distributor. If the sales deal with a distributor he can make him offers, the payment may be by credit or in cash.

Nouns:

Car, company, department, stock, attendance, bonus, stock, salary, employee, type, numbers client, distributor, offers, payment, credit, cash.

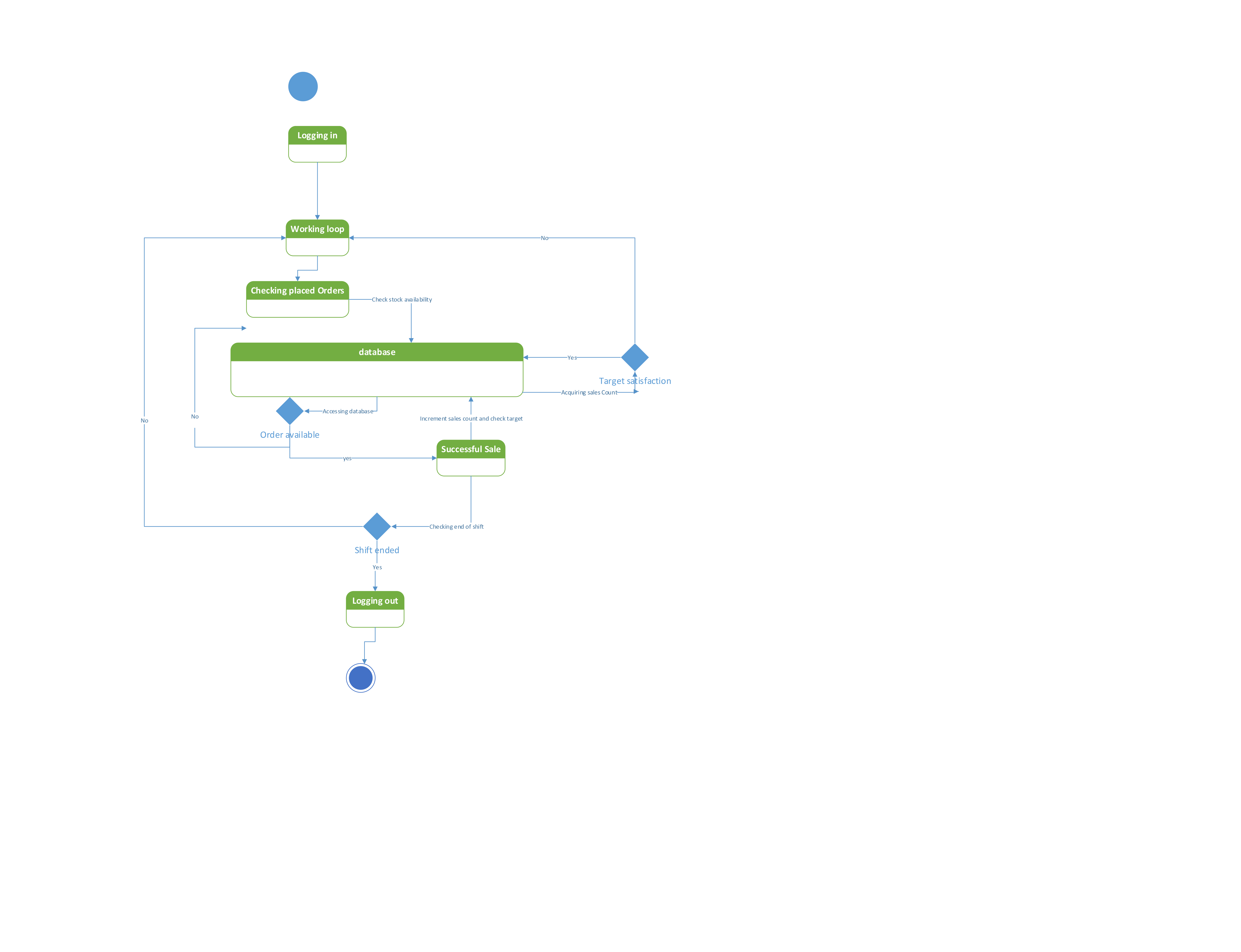
Nouns which can be attributes or important for operation:

Attendance, bonus, warning, salary, type, number, credit, cash.

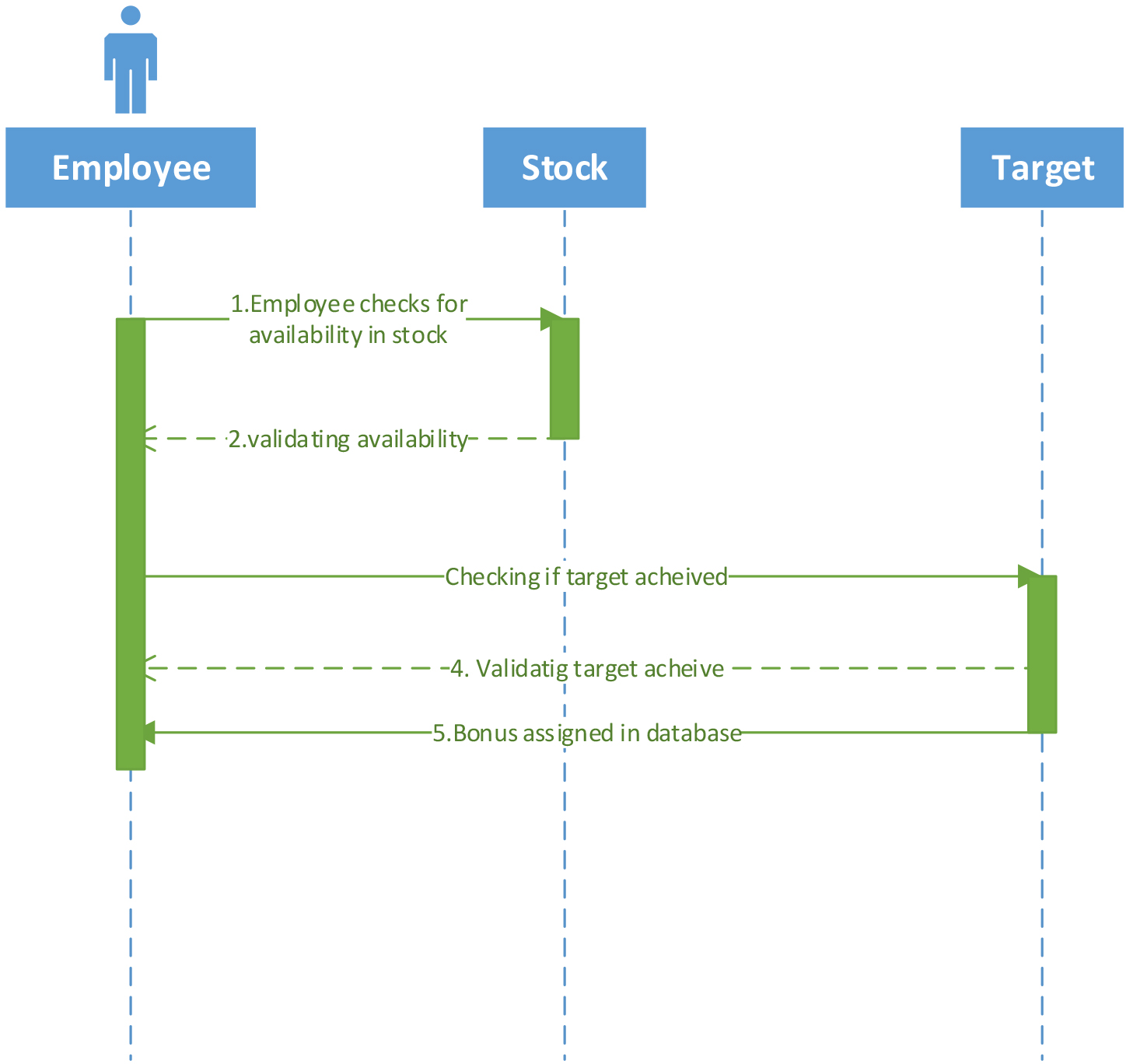
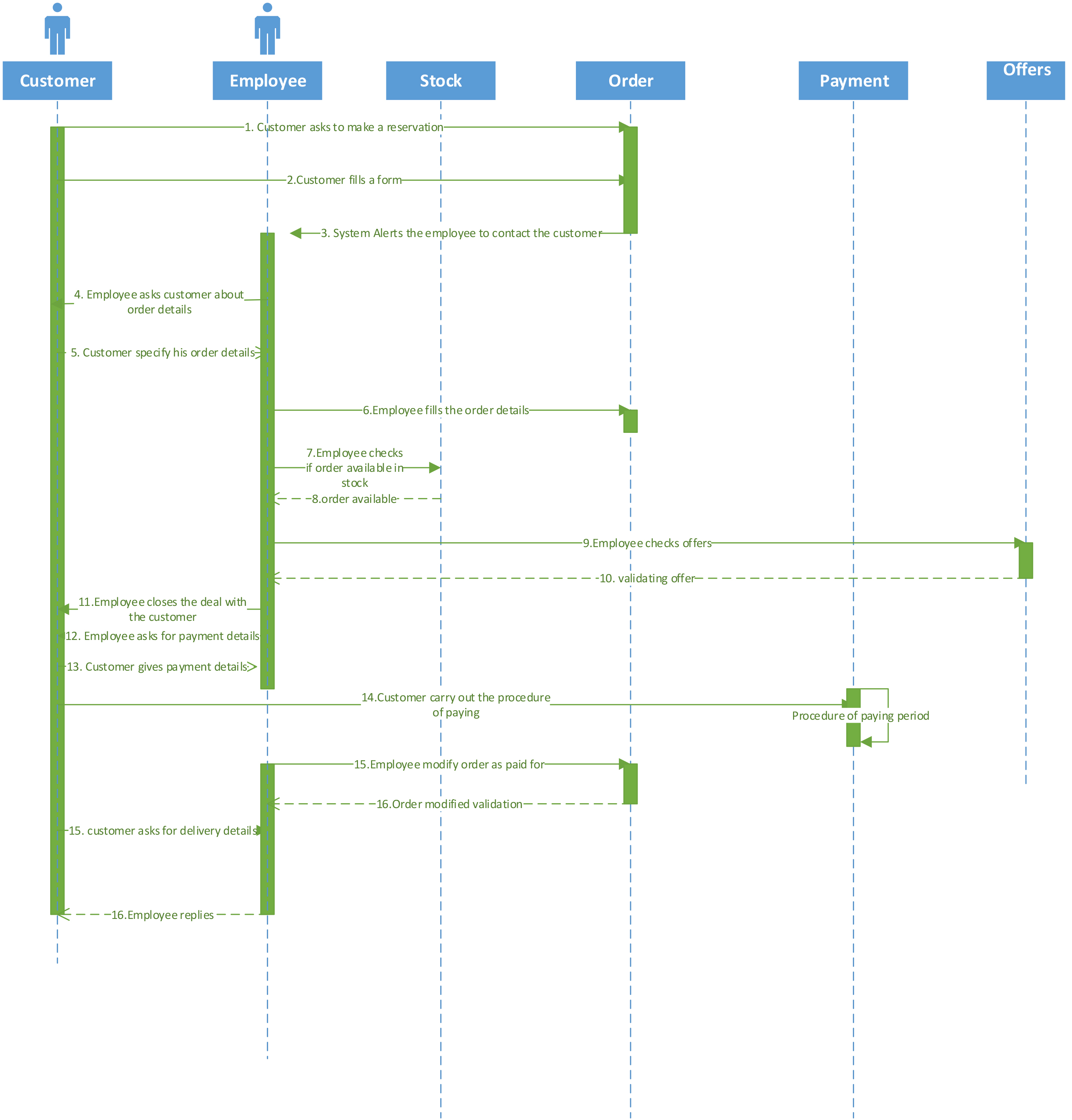
Nouns that could be name of classes:

Car, department, employee, stock, customer, distributor, offers, payment, target

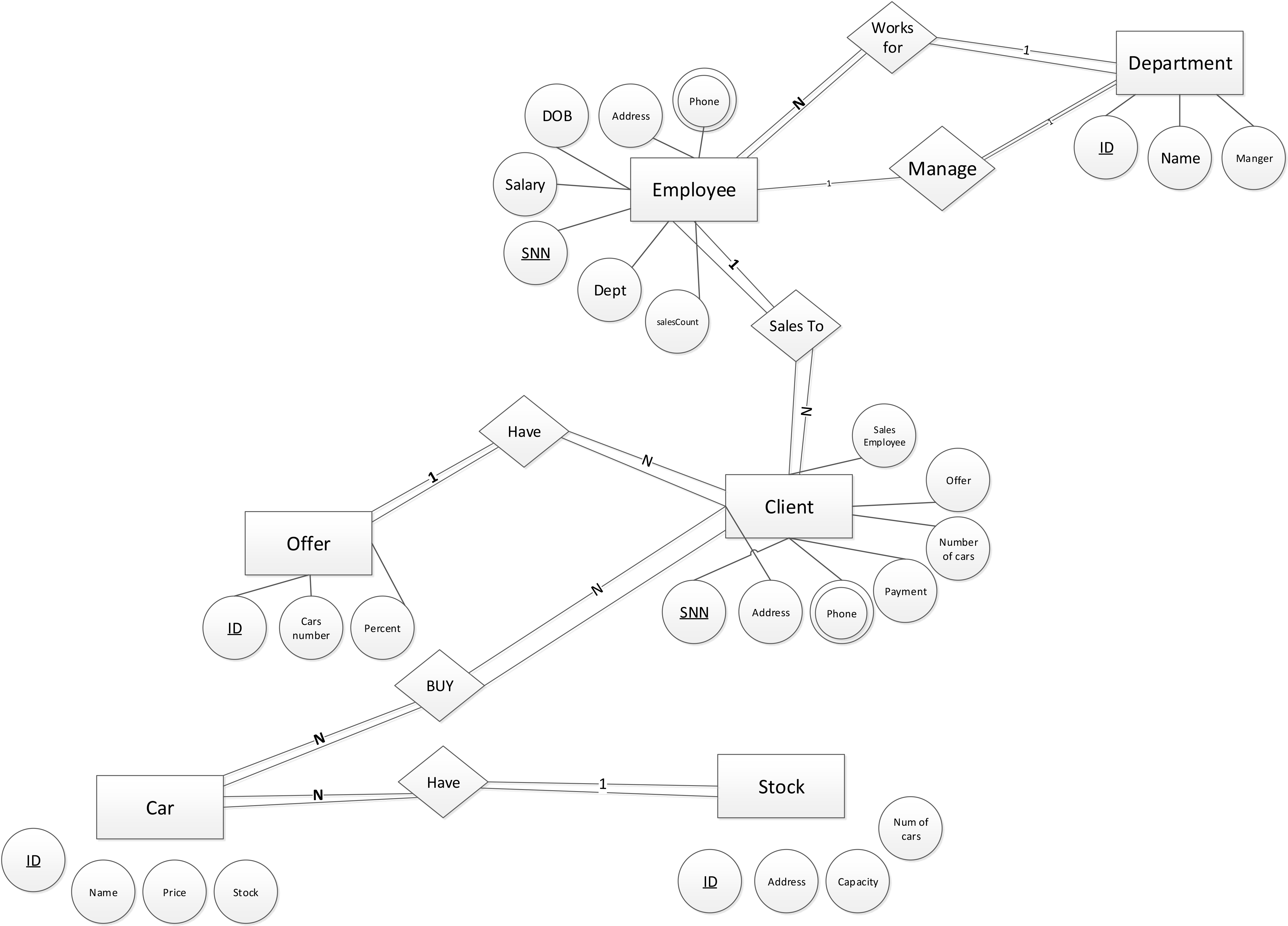
1. **State Diagram**



1. **Interaction Diagram**



1. **Detailed Class Diagram**
2. **Data Model Design**



1. **User Interface Design**

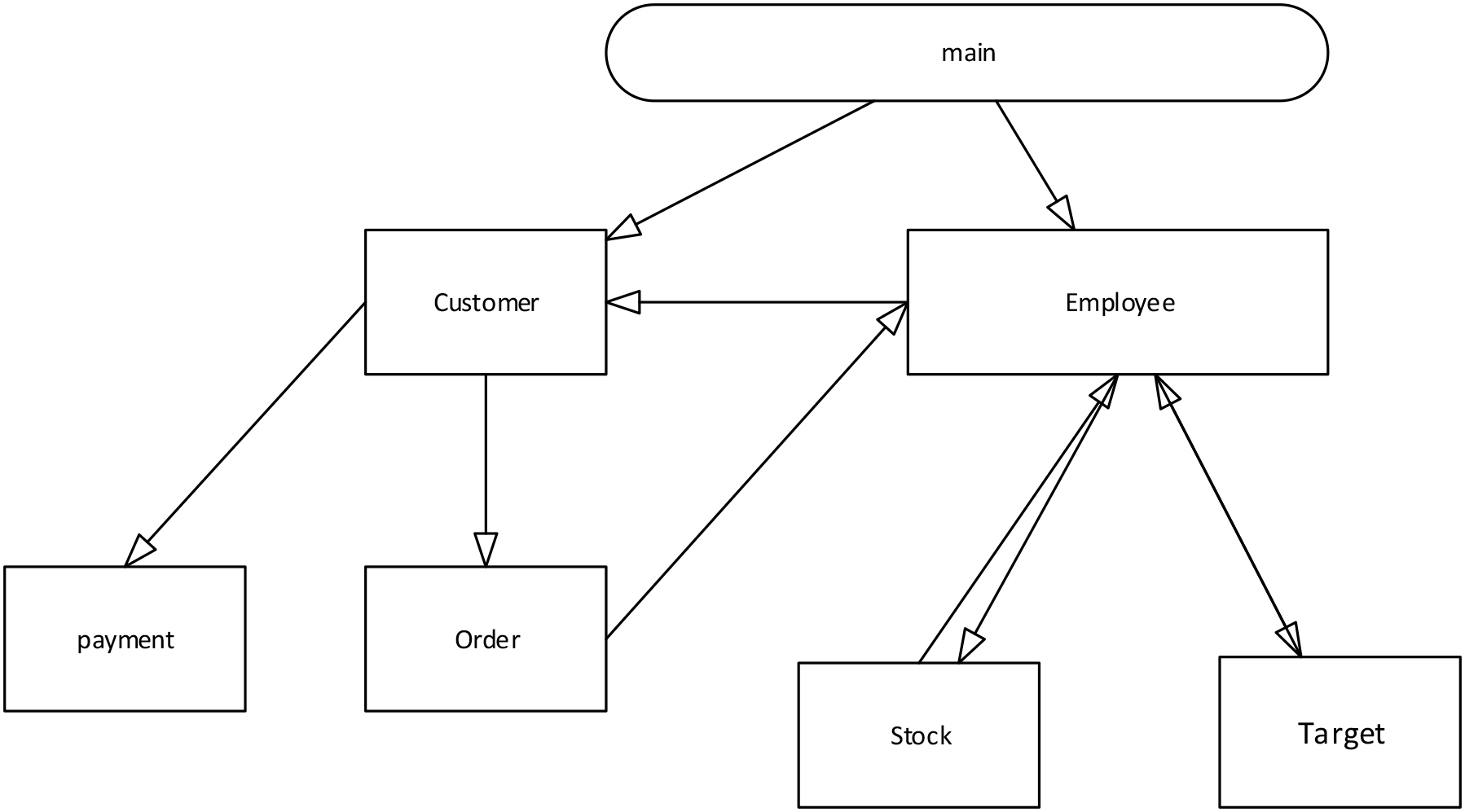
*Employee User Interface*

|  |  |
| --- | --- |
|  |  |
| UI styles adopted | **Button:**  Initial Login, Placed orders, check stock.  **Fill in form:**   * Login form (User name and password). * text area showing message show salary before submitting password. * Order Details form (number of cars, type, color, etc.   -**Display information via** Text with word processor in addition to -mostly- images. |
| Actions causing Interface state to change: | -Pressing Login Button displays profile.  -Pressing text area allows user to write.  -pressing placed orders button displays a fill in form for order details. |
| Interruptions | 1. If Order available: Message shown to notify employee. 2. Wrong credentials: Message notifying the user (wrong password or username). |

Customer User Interface

|  |  |
| --- | --- |
|  |  |
| UI adopted styles | **Button:**  -Submit button for the fill in form, Submit button for the payment details.  **Fill in form:**  **-**Personal information form.  -Payment details form. |
| Actions causing interface  State to change | -pressing the submit button changes interface. |
| Interruptions | -If order not available: customer should be notified by a message that his order is not currently available.  -If payment details validated: message should be sent to notify of the successful transaction and of expected delivery time. |

1. **Client-Object Relation Diagram**



1. **Detailed Design**

**Void** Working\_Loop(**void**){

**While**(True)

{

**Employee**::Login;

If (**Order** is placed){

Update Order details;

**Stock**::check order availability();

Access database to check if order available;

**If**(Cars Ordered less than Cars in stock){

**Stock**::alerts employee order is available;

**While** (True){

**If**(Payment was completed){

**Employee**::increment sales Count+=1;

**Target**::check target and compare to sales Count;

Access database to check;

**If**(sales Count was more or equal to Target){

Employee target satisfaction is True;

**Target**::Assign Bonus;

Access database and increment the salary += Bonus;

}

Break;

}

}

}

**Else**{

**Stock**::alerts employee order is not available;

}

}

**If**(Employee::Logout){

Break;

}

}

}

1. **Testing**

*Testing*

# Strategy:

* Testing strategy would be to assume most of the cases where the integrated performance takes place.
* Testing for the break down case.
* Testing for a series of continuous operations.

**Class Testing:**

* Testing for class stock to check for number of a certain type of cars.

**Integration Testing:**

* Testing for class employee a function to check order availability where class employee integrates with class stock to confirm if number of ordered cars less or equal cars of the same subclass as required.

**System Testing:**

* Checking many of the general cases.

Example of General cases:

Case1:

The customer has ordered a number of cars not available in stock.

Case 2:

The customer has placed an available order but the sale would not satisfy required target.

Case3:

The customer has placed an available order but the sale would satisfy required target making the database Assign bonus to the employee’s SSn

1. **Estimated Project Cost**

\*Effort = A (Size) B \* EM

A = 2.94  
 the estimated size = 3 KSLOC

B = 1

**EM:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Identifier** | Name | **Ranges**  **(VL – EH)** | **Assessment**  **VL/L/N/H/VH/EH** | **Values** |
| RCPX | product Reliability and Complexity | 0.5 – 1.5 | low | 0.75 |
| RUSE | required reusability | 0.5 – 1.5 | nominal | 1.0 |
| PDIF | Platform Difficulty | 0.5 – 1.5 | nominal | 1.0 |
| PERS | Personnel capability | 1.5 – 0.5 | high | 0.8 |
| PREX | Personnel Experience | 1.5 – 0.5 | nominal | 1.0 |
| FCIL | Facilities available | 1.5 – 0.5 | nominal | 1.0 |
| SCED | Schedule pressure | 1.5 – 0.5 | low | 1.2 |
|  | | | Product | 0.72 |

So:

EM = 0.72

Effort = 2.94 (3)1 \* 0.72 = 6.3504

\*Estimated Cost = PM \* Number of developers \* number of hours for 2 weeks

**Estimated** Cost = 6.3504 \*6 \* 160 = 6,096$