

**Assignment Submission Form**

**Assignment Details**

Module Code: MS805 & MS804 Assignment Title:

**Group Members: Group 52 (please use BLOCK CAPITALS)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Student ID** | **Programme** | **Student Name** | **Contact Details (Email, Telephone)** |
| 21239586 | 1MBY1 | Soumik Deepak Ghosh | [s.ghosh1@nuigalway.ie](mailto:s.ghosh1@nuigalway.ie), 0899803529 |
| 21239826 | 1MBY1 | Jacyle Cristiane Souza | [j.souza1@nuigalway.ie](mailto:j.souza1@nuigalway.ie), 0833116748 |
| 21239610 | 1MBY1 | Jiawen Zhang | [j.zhang8@nuigalway.ie](mailto:j.zhang8@nuigalway.ie), 0894811797 |

I/We hereby declare that this assignment submission

is my/our own original work. I/We have read the University *Code of Practice for Dealing with Plagiarism[[1]](#footnote-1)\** and are aware that the possible penalties for plagiarism include expulsion from the University**. *I/We attach a list of all sources that were consulted in the preparation of this assignment e.g. books, journals, Web sites etc.***

|  |  |
| --- | --- |
| **Signature** | **Date** |
| Soumik Ghosh | 06/11/2021 |
| Jacyle Cristiane Souza | 06/11/2021 |
| Jiawen Zhang | 06/11/2021 |
|  |  |

Table of Contents

[**Assignment Details** 1](#_Toc87102805)

[Table of Figures 2](#_Toc87102806)

[Table of Tables 2](#_Toc87102807)

[SECTION A 3](#_Toc87102808)

[A1. Functional Decomposition Diagram of the system. 3](#_Toc87102809)

[A2. Data Flow Diagrams 4](#_Toc87102810)

[A3. Process Level Data Flow Diagram 6](#_Toc87102811)

[A4. UML Use Case Diagram 7](#_Toc87102812)

[A5. UML Activity Diagram 8](#_Toc87102813)

[SECTION B 9](#_Toc87102814)

[B1. Logical Entity Relationship Model 9](#_Toc87102815)

[B2. Normalized Relational Database Schema 10](#_Toc87102816)

[B3. SQL Statements to CREATE TABLEs 16](#_Toc87102817)

[Assumptions 18](#_Toc87102818)

## Table of Figures

[Figure 1: Functional Decomposition Diagram 3](#_Toc87102739)

[Figure 2: Context-Level Data Flow Diagram for Teton Whitewater Kayak System 4](#_Toc87102740)

[Figure 3: System-Level Data Flow Diagram for Teton Whitewater Kayak System 5](#_Toc87102741)

[Figure 4: Process Level Data Flow Diagram for Teton Whitewater Kayak Rent System 6](#_Toc87102742)

[Figure 5: Use Case for Equipment Rent System - Teton Whitewater Kayak System 7](#_Toc87102743)

[Figure 6: UML Activity Diagram of Rent System - Teton Whitewater Kayak System 8](#_Toc87102744)

[Figure 7: Logical Entity Relationship Model 9](#_Toc87102745)

[Figure 8: Entity Relationship Model for Teton Whitewater Kayak System 10](#_Toc87102746)

## Table of Tables

[Table 1: Database Schema for Teton Whitewater Kayak System 10](#_Toc87102787)

[Table 2: Database Table Normalisation Factors 13](#_Toc87102788)

# SECTION A

## A1. Functional Decomposition Diagram of the system.

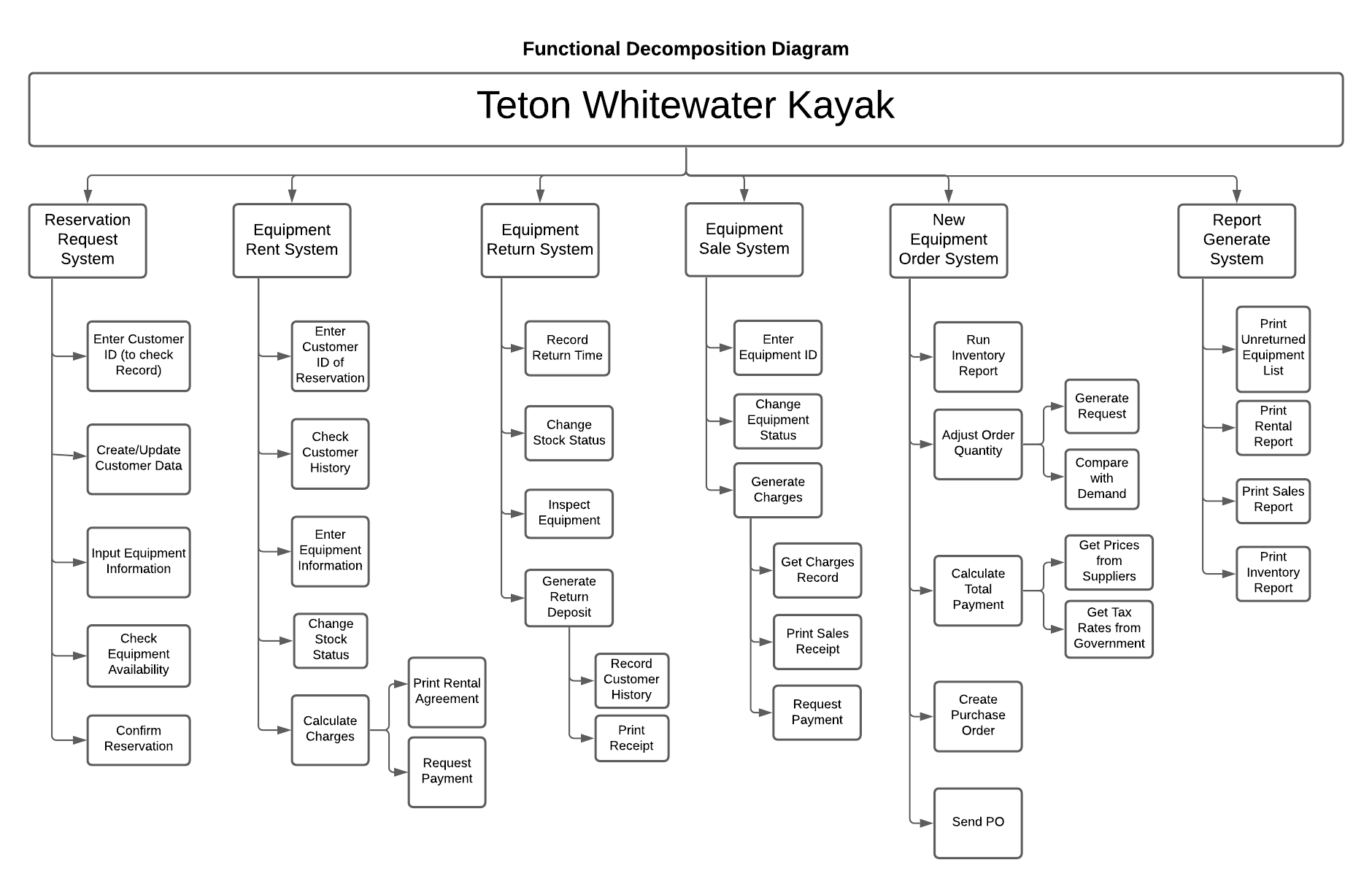


Figure 1: Functional Decomposition Diagram

The Functional Decomposition Diagram is decomposed into 6 functions each having their own sub-functions.

1. Reservation Request System.
2. Equipment Rent System
3. Equipment Return System
4. Equipment Sale System
5. New Equipment Order System
6. Report Generate System

## A2. Data Flow Diagrams

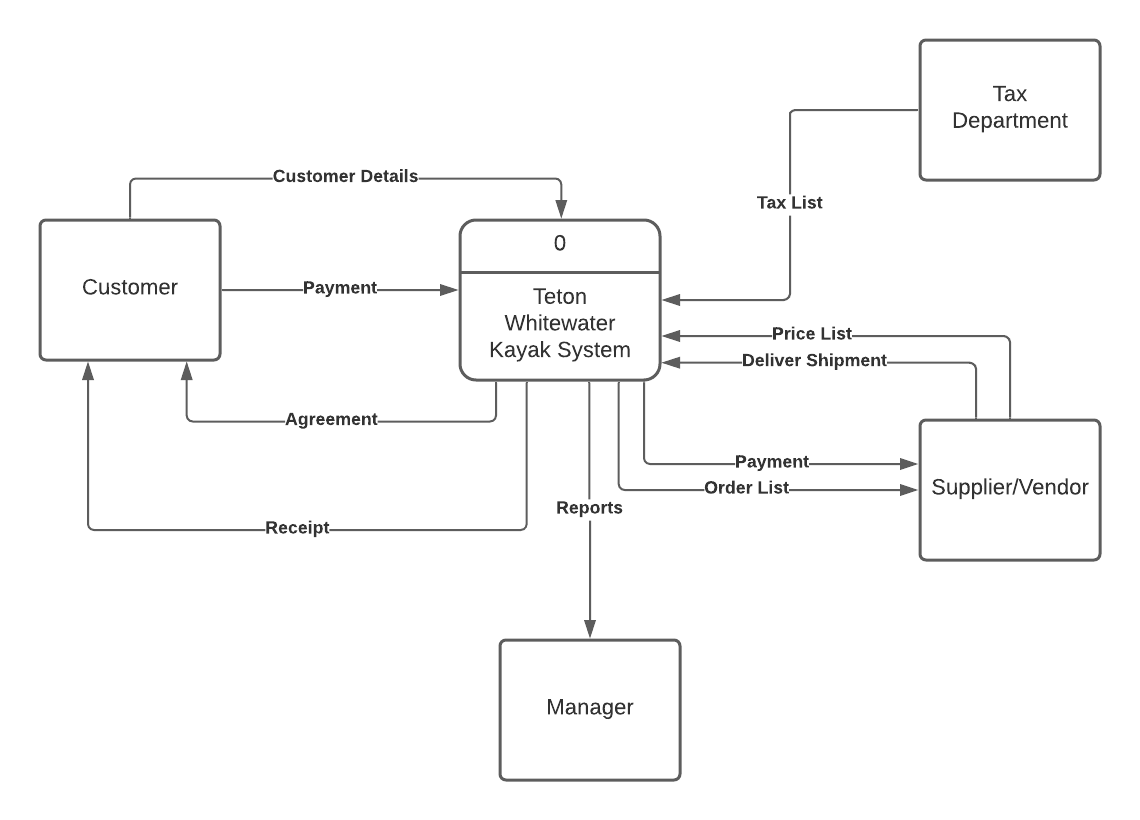


Figure 2: Context-Level Data Flow Diagram for Teton Whitewater Kayak System

In the Context Level Data Flow Diagram, Customer, Manager, Supplier and Tax Department are considered as External Agents of the Teton Whitewater Kayak System (System). Customer begins the flow by providing the Customer Details for reservation of equipment. The customer is provided with Agreement and upon receiving Payment from customer, the system outputs receipt to the customer. The System shares the Order list with the Supplier and in turn the Supplier shares the Price List and Delivery Shipment with the System. The System then Provides Payment information with the Supplier at the end of Purchase Cycle. The Manager gets various Reports from the System. The Tax Department sends the Tax information to the system which is required to create Purchase Orders.

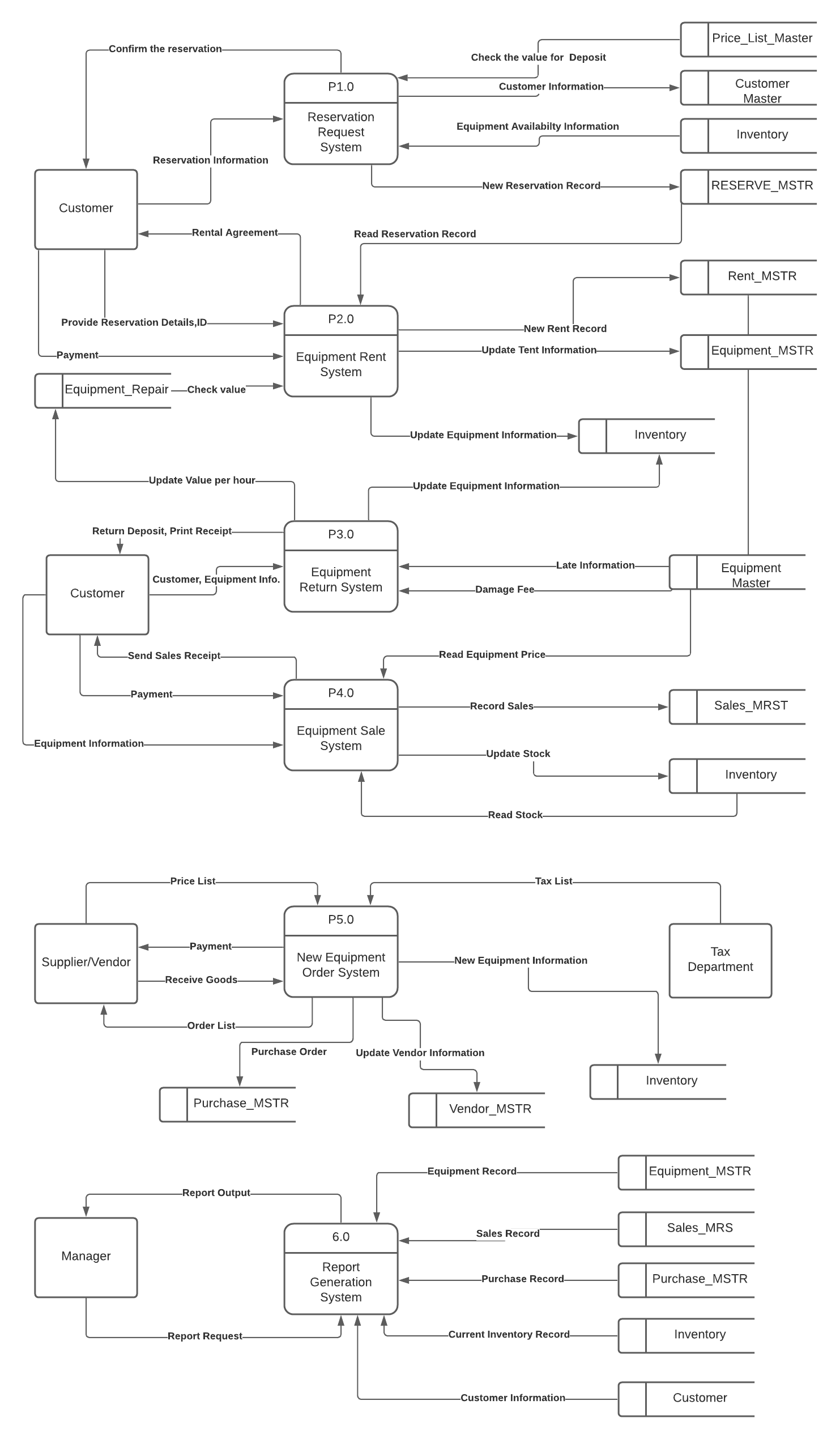


Figure 3: System-Level Data Flow Diagram for Teton Whitewater Kayak System

## A3. Process Level Data Flow Diagram

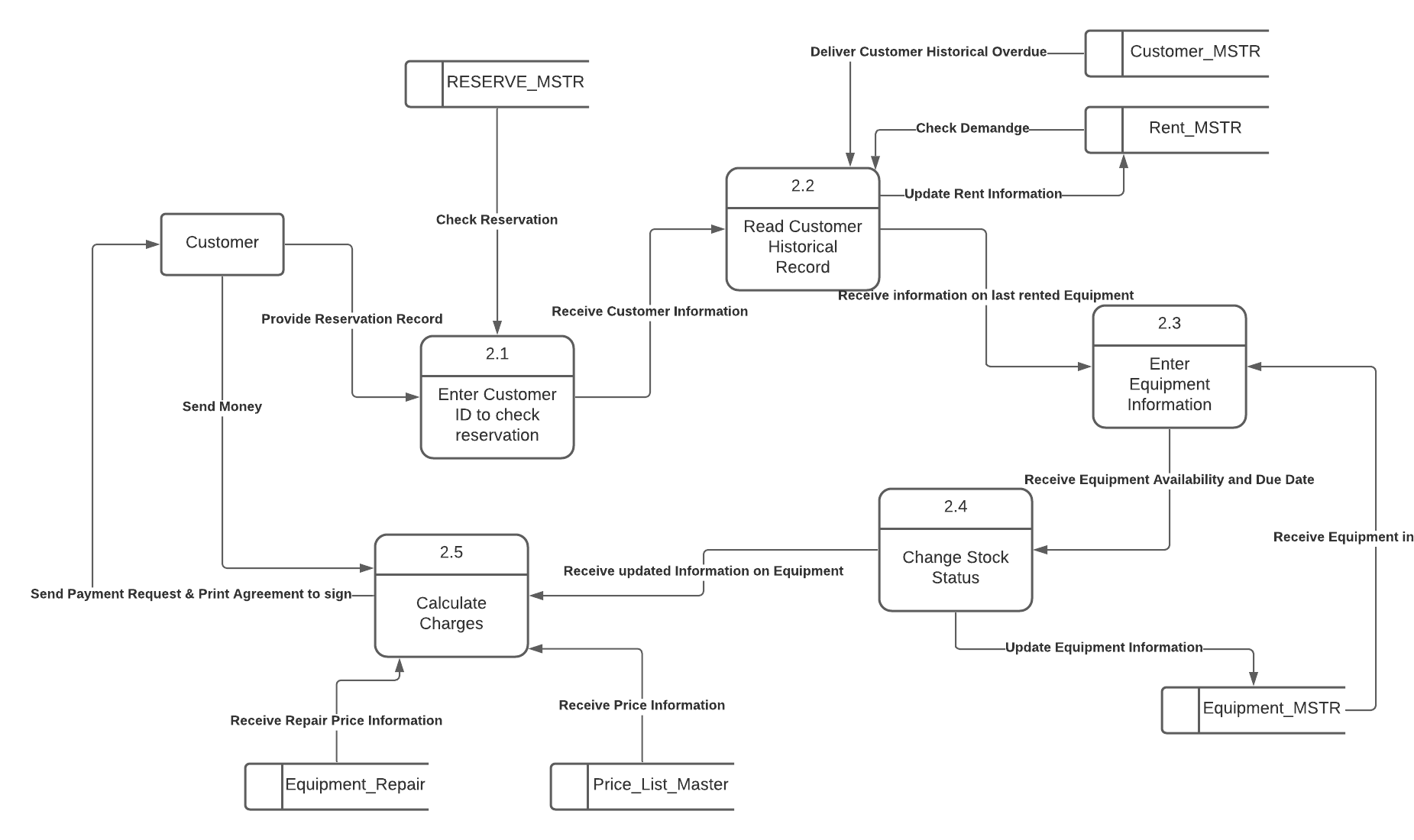


Figure 4: Process Level Data Flow Diagram for Teton Whitewater Kayak Rent System

The above Data Flow Diagram in Figure 4 describes the Rent System for Teton Whitewater Kayak. The process begins with Customer providing their reservation details to the system. Process 2.1 Checks the Reservation ID against the system records in RESERVE\_MSTR. Process 2.2 reads customer history from RENT\_MSTR and CUSTOMER\_MSTR entities to find any overdue amounts against the customer. Process 2.3 Records and fetches the equipment information from EQIUPMENT\_MSTR and passes the information to Process 2.4 when the Stock Status is updated to “rented”. Once the stock status is changed, process 2.5 calculates the charges from the PRICE\_LIST\_MSTR and EQUIPMENT\_REPAIR (in case of any past repair charges) and provides all the information about Rent to the Customer.

## A4. UML Use Case Diagram

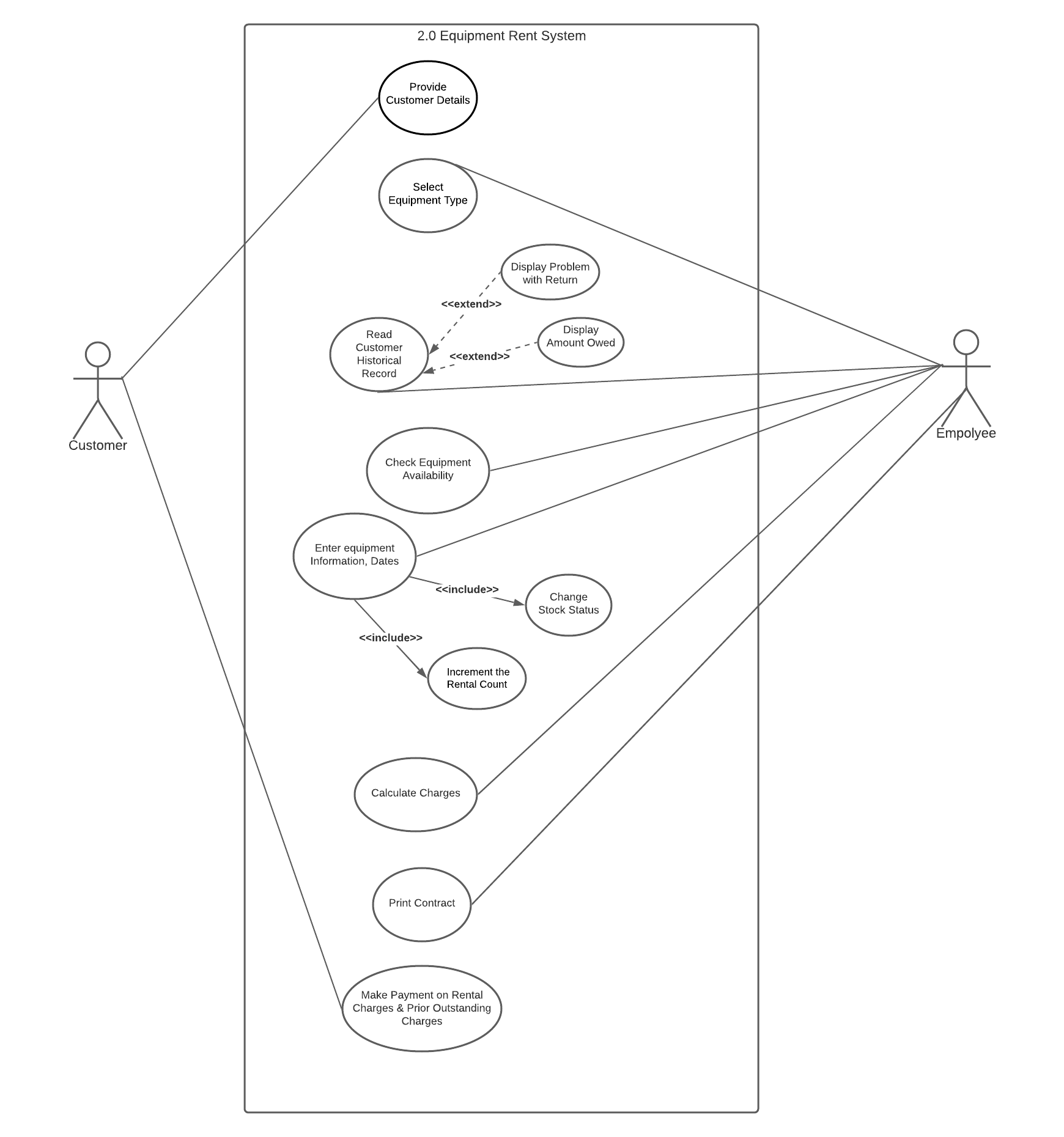


Figure 5: Use Case for Equipment Rent System - Teton Whitewater Kayak System

The UML Use Case Diagram in Figure 5 above shows how 2 actors, customer and employee, interact with the system to complete the working of the Rent System for Teton Whitewater Kayak System. The Employee is the major actor here with the most interactions with the processes. Customer provides their details to the system and the Employee selects the Equipment Type from the system database that the customer wishes to rent. Employee also reads the historical records of customer to determine any Damage or Overdue amounts. After checking the Equipment Availability and updating the Equipment Information in the system, the Employee proceeds to Calculate charges and print the Rent Contract. The customer makes the due payment and collects the equipment.

## A5. UML Activity Diagram

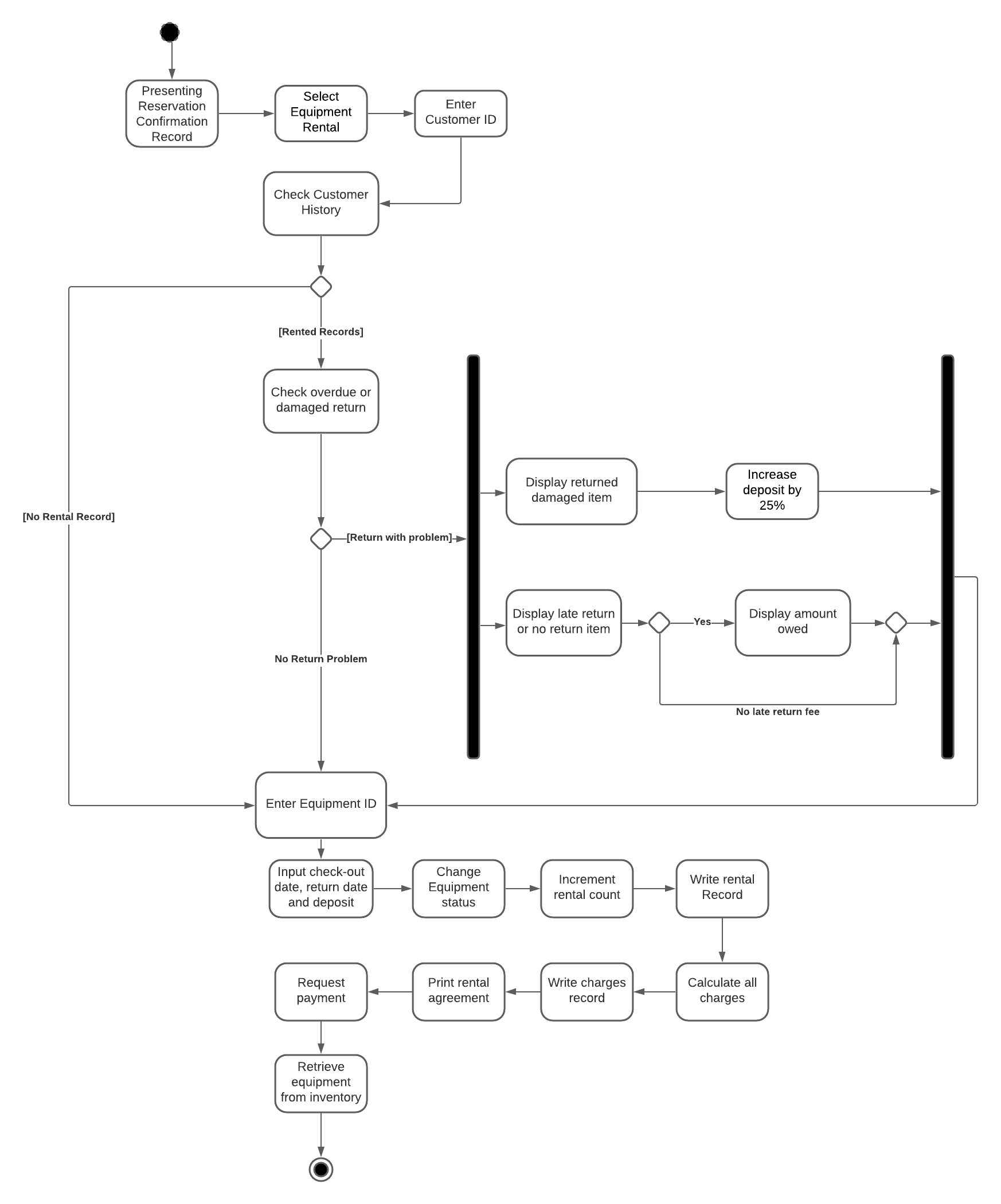


Figure 6: UML Activity Diagram of Rent System - Teton Whitewater Kayak System

# SECTION B

## B1. Logical Entity Relationship Model

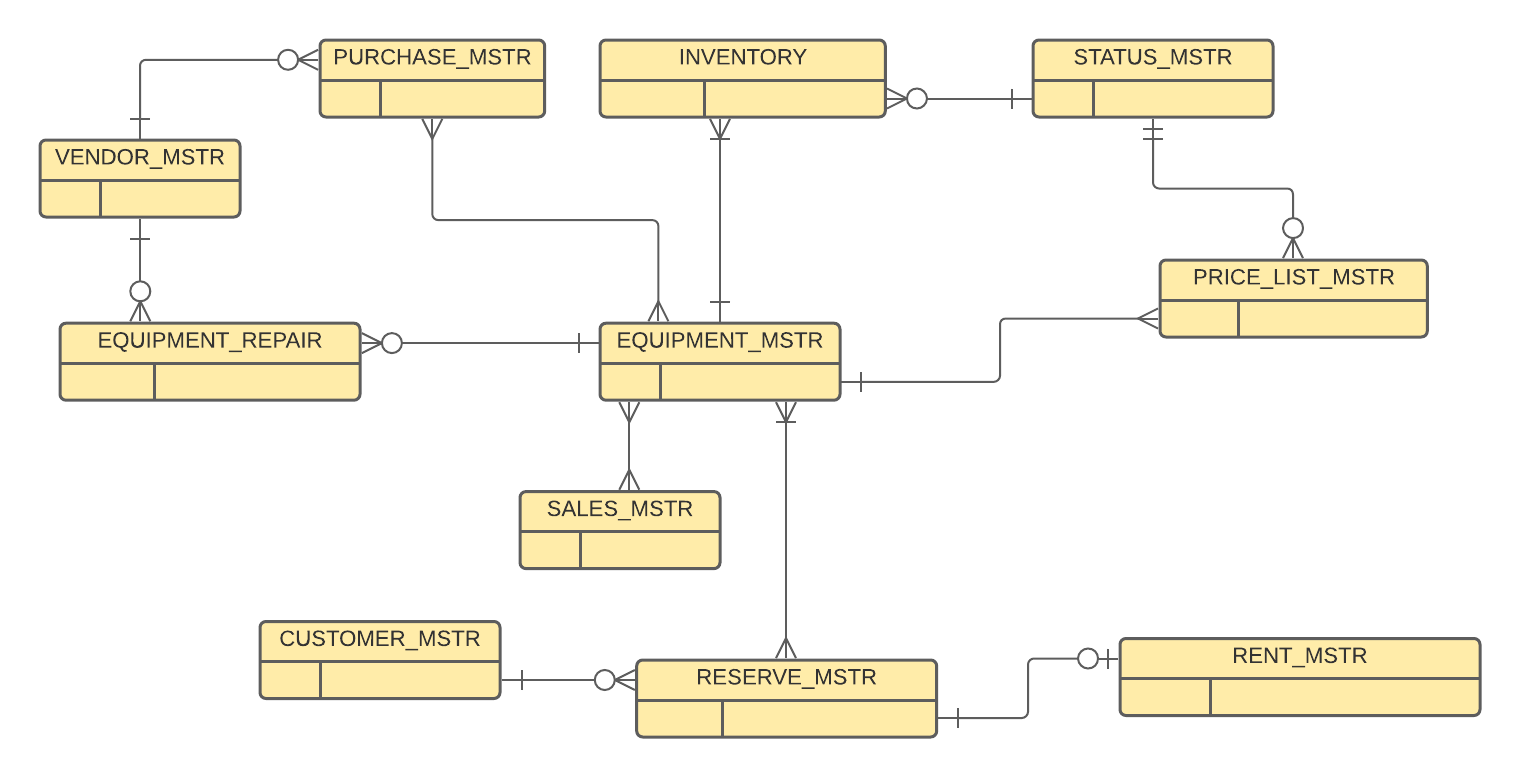


Figure 7: Logical Entity Relationship Model

Figure 7 above shows the logical Entity Relationship Model for the Teton Whitewater Kayak System. EQUIPMENT\_MSTR is the central entity which will contain all the equipment information like Equipment Type, Total available quantity in the store, Equipment ID, etc. Inventory Entity will hold specific inventory information like the quantities available for Sale, Rent and Reserved quantities. These inventory types will be based on different status from STATUS\_MSTR. PRICE\_LIST\_MSTR stores various prices related to an equipment based on different status like RENT, SALE, DAMAGE, etc. RESERVE\_MSTR refers EQUIPMENT\_MSTR for equipment information and since an equipment can be reserved and rented multiple times and also a reservation can be made for multiple equipment, there is a Many-To-Many relationship between EQUIPMENT\_MSTR and RESERVE\_MSTR. These many-to-many relationship will be broken down using join tables while normalising. EQUIPMENT\_REPAIR entity links with EQUIPMENT\_MSTR and VENDOR\_MSTR. It will store the equipment repair costs for each equipment that is provided by each vendor.

## B2. Normalized Relational Database Schema

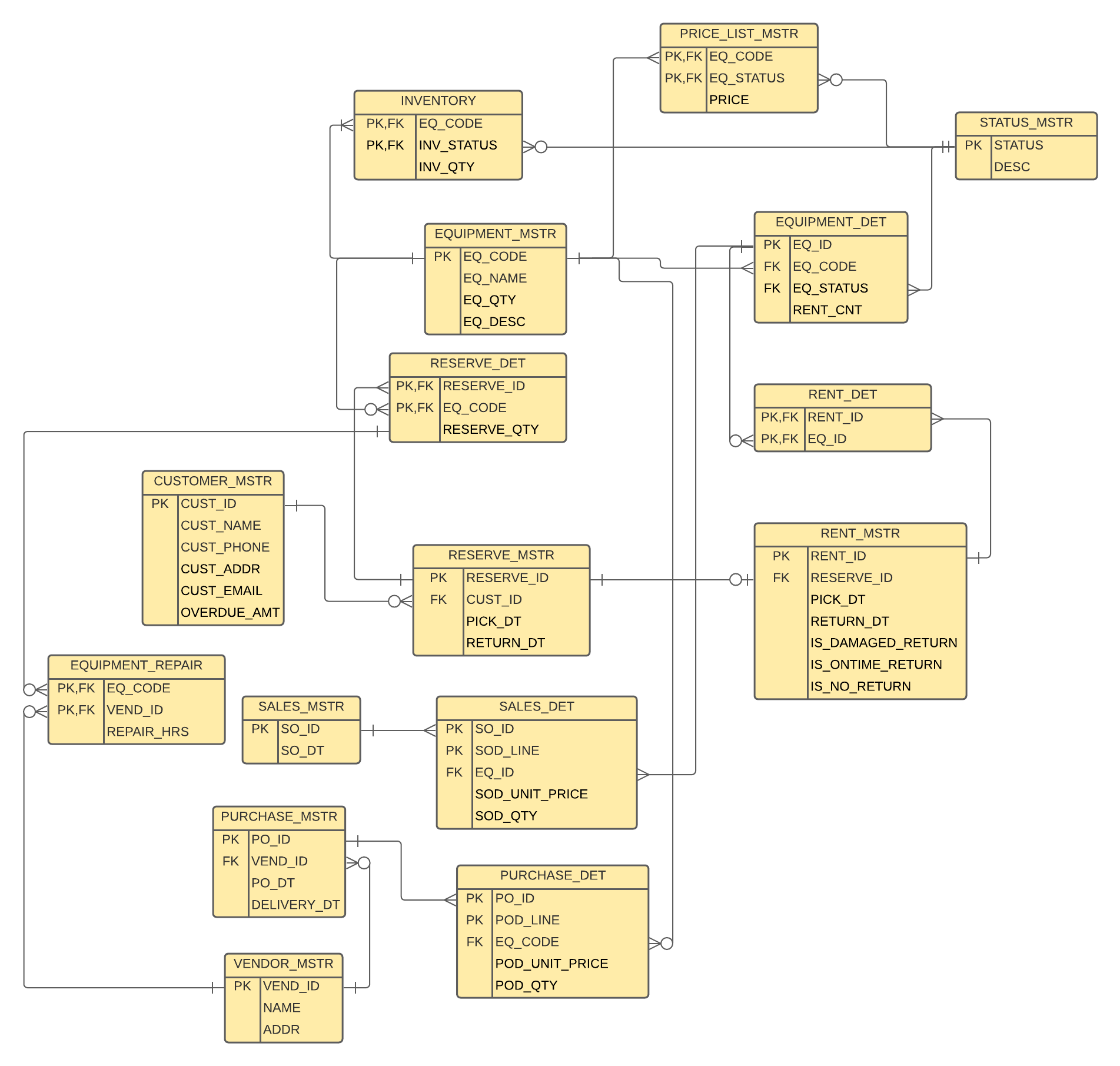


Figure 8: Entity Relationship Model for Teton Whitewater Kayak System

Assumptions:

1. An equipment type can be supplied from different suppliers. Each supplier has their own quote for equipment repairs.
2. Customer History can be found by using appropriate procedures by querying on RENT\_MSTR.
3. STATUS\_MSTR contains Equipment Status like “Reserved”, “Rent”, Inventory Status like “For Rent”, “For Sale”, “Damaged” and Price List status like “Rent”, “Deposit”, “Repair”, “Sale” etc. These statuses will be used to fetch different information based on the requirement.
4. Each Reservation will have only 1 corresponding Rent.

Table 1: Database Schema for Teton Whitewater Kayak System

|  |  |  |  |
| --- | --- | --- | --- |
| **Teton Whitewater Kayak System** | |  |  |
| **KEY** | **FIELD NAME** | **DATATYPE** | **DESCRIPTION** |
|  |  |  |  |
| **EQUIPMENT\_MSTR** |  |  |  |
| PRIMARY KEY | EQ\_CODE | VARCHAR(20) | Code for each Equipment Type |
|  | EQ\_NAME | VARCHAR(50) | Readable name for each equipment type |
|  | EQ\_DESC | VARCHAR(30) | Equipment description |
|  | EQ\_QTY | DECIMAL | Total equipment qty in inventory (includes Reserved, For Rent, and Sale) |
|  |  |  |  |
| **VENDOR\_MSTR** |  |  |  |
| PRIMARY KEY | VEND\_ID | VARCHAR(20) | Unique ID for each Vendor |
|  | NAME | VARCHAR(50) | Name of Vendor |
|  | ADDR | VARCHAR(100) | Address of vendor |
|  |  |  |  |
|  | RENT\_CNT | INTEGER | Rental count |
|  |  |  |  |
| **CUSTOMER\_MSTR** |  |  |  |
| PRIMARY KEY | CUST\_ID | VARCHAR(20) | Unique ID for each Customer |
|  | CUST\_NAME | VARCHAR(50) | Customer name |
|  | CUST\_PHONE | VARCHAR(15) | Telephone contact of customer |
|  | CUST\_ADDR | VARCHAR(100) | Postal Address of customer |
|  | CUST\_EMAIL | VARCHAR(50) | E-Mail address of customer |
|  | OVERDUE\_AMT | DECIMAL | Any overdue payment from customer |
|  |  |  |  |
| **SALES\_MSTR** |  |  |  |
| PRIMARY KEY | SO\_ID | VARCHAR(20) | Unique ID for each Sales Orders |
|  | SO\_DT | VARCHAR(10) | Sales Date |
|  |  |  |  |
| **RESERVE\_MSTR** |  |  |  |
| PRIMARY KEY | RESERVE\_ID | VARCHAR(20) | Unique ID for each Reservation |
| FOREIGN KEY References CUSTOMER\_MSTR | CUST\_ID | VARCHAR(20) | Customer ID of the customer from CUSTOMER\_MSTR |
|  | PICK\_DT | VARCHAR(10) | Pickup Date for order |
|  | RETURN\_DT | VARCHAR(10) | Return Date for order |
|  |  |  |  |
| **RESERVE\_DET** |  |  |  |
| PRIMARY KEY FOREIGN KEY References RESERVE\_MSTR | RESERVE\_ID | VARCHAR(20) | Reservation ID |
| PRIMARY KEY FOREIGN KEY References EQUIPMENT\_MSTR | EQ\_CODE | VARCHAR(20) | Equipment code for each equipment |
|  | RESERVE\_QTY | DECIMAL | Reserved qty of each equipment for a reservation |
|  |  |  |  |
| **RENT\_MSTR** |  |  |  |
| PRIMARY KEY | RENT\_ID | VARCHAR(20) | Unique ID for each Rent order |
| FOREIGN KEY References RESERVE\_MSTR | RESERVE\_ID | VARCHAR(20) | Reserve ID against which a Rent is availed |
|  | PICK\_DT | VARCHAR(10) | Actual Rent Pickup Date |
|  | RETURN\_DT | VARCHAR(10) | Actual Rent Return Date |
|  | IS\_DAMAGED\_RETURN | BOOLEAN | Whether returned equipment is damaged |
|  | IS\_ONTIME\_RETURN | BOOLEAN | Whether returned equipment is late |
|  | IS\_NO\_RETURN | BOOLEAN | Whether equipment is not returned |
|  |  |  |  |
| **RENT\_DET** |  |  |  |
| PRIMARY KEY, FOREIGN KEY  References RENT\_MSTR | RENT\_ID | VARCHAR(20) | Rent ID of the rent order |
| PRIMARY KEY, FOREIGN KEY  References EQUIPMENT\_DET | EQ\_ID | VARCHAR(20) | Equipment ID unique to an equipment |
|  |  |  |  |
| **INVENTORY** |  |  |  |
| PRIMARY KEY,  FOREIGN KEY  References EQUIPMENT\_MSTR | EQ\_CODE | VARCHAR(20) | Equipment code for an equipment type |
| PRIMARY KEY,  FOREIGN KEY  References STATUS\_MSTR | INV\_STATUS | VARCHAR(10) | Inventory Status i.e STOCK, RENT, FOR SALE |
|  | INV\_QTY | DECIMAL | Qty of Equipment available in inventory |
|  |  |  |  |
| **PRICE\_LIST\_MSTR** |  |  |  |
| PRIMARY KEY, FOREIGN KEY References EQIUPMENT\_MSTR | EQ\_CODE | VARCHAR(20) | Equipment code for an equipment type |
| PRIMARY KEY, FOREIGN KEY References STATUS\_MSTR | EQ\_STATUS | VARCHAR(10) | Status of Equipment; Values = FOR RENT, FOR SALE, DAMAGED, DEPOSIT,REPAIR |
|  | PRICE | DECIMAL | Price for an equipment; Depending on Equipment and Status, prices is different |
|  |  |  |  |
| **STATUS\_MSTR** |  |  |  |
| PRIMARY KEY | STATUS | VARCHAR(10) | Status Code; Values: STOCK, FOR RENT, FOR SALE, DAMAGED, DEPOSIT, REPAIR, SOLD, RESERVED |
|  | DESC | VARCHAR(30) | Description |
|  |  |  |  |
| **SALES\_DET** |  |  |  |
| PRIMARY KEY,  FOREIGN KEY references SALES\_MSTR | SO\_ID | VARCHAR(20) | Unique ID for each Sales Orders |
| PRIMARY KEY | SOD\_LINE | INTEGER | Unique Line for each Sales Order Line item |
| FOREIGN\_KEY  References EQUIPMENT\_MSTR | EQ\_CODE | VARCHAR(20) | Equipment Code for each equipment type |
|  | SOD\_UNIT\_PRICE | DECIMAL | Price at of equipment sold |
|  | SOD\_QTY | DECIMAL | Quantity of equipment sold |
|  |  |  |  |
| **PURCHASE\_MSTR** |  |  |  |
| PRIMARY KEY | PO\_ID | VARCHAR(20) | Unique ID for each Purchase Order |
| FOREIGN KEY References VENDOR\_MSTR | VEND\_ID | VARCHAR(20) | ID of Vendor |
|  | PO\_DT | VARCHAR(10) | Purchase Order Date |
|  | DELIVERY\_DT | VARCHAR(10) | Delivery Date of purchase order |
|  |  |  |  |
| **PURCHASE\_DET** |  |  |  |
| PRIMARY KEY,  FOREIGN KEY References PURCHASE\_MSTR | PO\_ID | VARCHAR(20) | Unique ID for each Purchase Order |
| PRIMARY KEY | POD\_LINE | INTEGER | Unique Line for each Purchase Order Line item |
| FOREIGN KEY References EQUIPMENT\_MSTR | EQ\_CODE | VARCHAR(20) | Equipment Code for each equipment type |
|  | POD\_UNIT\_PRICE | DECIMAL | Price at of equipment purchased |
|  | POD\_QTY | DECIMAL |  |
|  |  |  |  |
| **EQUIPMENT\_REPAIR** |  |  |  |
| PRIMARY KEY, FOREIGN KEY References EQIUPMENT\_MSTR | EQ\_CODE | VARCHAR(20) | Unique Equipment ID for each equipment |
| PRIMARY KEY, FOREIGN KEY References VENDOR\_MSTR | VEND\_ID | VARCHAR(20) | Unique Vendor ID for each vendor |
|  | REPAIR\_HRS | INTEGER | Number of hours required to repair the equipment. |

Table 2: Database Table Normalisation Factors

|  |  |  |  |
| --- | --- | --- | --- |
| **TABLES** | **1NF** | **2NF** | **3NF** |
| EQUIPMENT\_MSTR | All attributes are atomic. No multi valued groups. Non-Key attributes are dependent on EQ\_CODE. As EQ\_CODE changes, EQ\_NAME & EQ\_QTY change As well. EQ\_CODE is unique for each record. | Only 1 Primary Key attribute on which all other non-key attributes depend. Automatically in 2NF | There are no transitive dependencies among non-key attributes. EQ\_NAME, EQ\_QTY & EQ\_DESC do not depend on each other. |
| EQUIPMENT\_DET | All attributes are atomic. No multi valued groups. Non-Key attributes are dependent on EQ\_ID. EQ\_CODE, EQ\_STATUS & RENT\_CNT can be determined using EQ\_ID. EQ\_ID is unique for each record. | Only 1 Primary Key attribute on which all other non-key attributes depend. Automatically in 2NF | There are no transitive dependencies among non-key attributes. EQ\_CODE, EQ\_STATUS & RENT\_CNT do not depend on each other. |
| CUSTOMER\_MSTR | CUST\_ID is unique Primary key for each record.  All non-key attributes depend on CUST\_ID. Using CUST\_ID, values of all other attributes can be determined. | Only 1 Primary Key attribute on which all other non-key attributes depend. Automatically in 2NF | There are no transitive dependencies among non-key attributes. CUST\_NAME, CUST\_PHONE, CUST\_ADDR, CUST\_EMAIL & OVERDUE\_AMT do not depend on each other. |
| STATUS\_MSTR | STATUS is unique Primary Key for each record. There are no multi-valued attributes. | Only 1 Primary Key attribute on which all other non-key attributes depend. Automatically in 2NF | There is ONLY ONE non-key attribute. No transitive dependencies. Automatically in 3NF |
| PRICE\_LIST\_MSTR | Primary key is identified. No repeating groups.  Non-Key attributes (PRICE) depend on EQ\_CODE & EQ\_STATUS | Non-Key attributes (PRICE) does not depend partially on COMPOSITE PRIMARY KEY. | There is ONLY ONE non-key attribute. No transitive dependencies. Automatically in 3NF |
| INVENTORY | Primary key is identified. No repeating groups.  Non-Key attributes (INV\_QTY) depend on EQ\_CODE & INV\_STATUS | Non-Key attributes (INV\_QTY) does not depend partially on COMPOSITE PRIMARY KEY. EQ\_CODE & INV\_STATUS determine the value of INV\_QTY | There is ONLY ONE non-key attribute. No transitive dependencies. Automatically in 3NF |
| RESERVE\_MSTR | Primary key is identified- RESERVE\_ID. No Repeating groups. | Only 1 Primary Key attribute on which all other non-key attributes depend. Automatically in 2NF | There are no transitive dependencies among non-key attributes. PICK\_DT, RETURN\_DT & CUST\_ID do not depend on each other. |
| RESERVE\_DET | Primary key is identified - RESERVE\_ID, EQ\_CODE No Repeating groups | RESERVE\_QTY is fully dependent on COMPOSITE PRIMARY KEY - RESERVE\_ID, EQ\_CODE | There is ONLY ONE non-key attribute. No transitive dependencies. Automatically in 3NF |
| RENT\_MSTR | Primary key is identified - RENT\_ID No Repeating groups | Only 1 Primary Key attribute on which all other non-key attributes depend. Automatically in 2NF | There are no transitive dependencies among non-key attributes. PICK\_DT,RETURN\_DT,IS\_DAMAGED\_RETURN,IS\_LATE\_RETURN do not depend on each other. |
| RENT\_DET | Primary key is identified - RENT\_ID,EQ\_ID No repeating groups | No non-key attributes. Automatically in 2NF | No non-key attributes so no transitive dependencies. |
| SALES\_MSTR | Primary key is identified. No repeating groups.  Non-Key attributes (SO\_DT) depend on ONLY on SO\_ID | Only 1 Primary Key attribute on which all other non-key attributes depend. Automatically in 2NF | There is ONLY ONE non-key attribute. No transitive dependencies. Automatically in 3NF |
| SALES\_DET | Primary key is identified - SO\_ID, SOD\_LINE No repeating groups | EQ\_ID, SOD\_UNIT\_PRICE & SOD\_QTY all depend fully on COMPOSITE PRIMARY KEY (SO\_ID, SOD\_LINE) | There are no transitive dependencies between non-key attributes. EQ\_ID, SOD\_UNIT\_PRICE & SOD\_QTY are not dependent on each other |
| PURCHASE\_MSTR | Primary key is identified. No repeating groups.  Non-key attributes depend fully on primary key | Only 1 Primary Key attribute on which all other non-key attributes depend. Automatically in 2NF | VEND\_ID, PO\_DT & DELIVERY\_DT do not depend on each other. No transitive dependencies |
| PURCHASE\_DET | Primary key is identified - PO\_ID, POD\_LINE No repeating groups | EQ\_ID, POD\_UNIT\_PRICE & POD\_QTY all depend fully on COMPOSITE PRIMARY KEY (SO\_ID, SOD\_LINE) | There are no transitive dependencies between non-key attributes. EQ\_ID, POD\_UNIT\_PRICE & POD\_QTY are not dependent on each other |
| VENDOR\_MSTR | Primary key is identified - VEND\_ID No Repeating groups | Only 1 Primary Key attribute on which all other non-key attributes depend. Automatically in 2NF | There are no transitive dependencies between non-key attributes. NAME & ADDR are not dependent on each other |
| EQUIPMENT\_REPAIR | Primary key is identified - EQ\_CODE, VEND\_ID No Repeating groups | REPAIR\_HR is dependent fully on COMPOSITE PRIMARY KEY (EC\_CODE, VEND\_ID) | There is ONLY ONE non-key attribute. No transitive dependencies. Automatically in 3NF |

## B3. SQL Statements to CREATE TABLEs

**CREATE** **TABLE** STATUS\_MSTR **(**

STATUS VARCHAR**(**8**),**

STATUS\_DESC TINYTEXT**,**

**PRIMARY** **KEY** **(**STATUS**));**

**CREATE** **TABLE** EQUIPMENT\_MSTR **(**

EQ\_CODE VARCHAR**(**24**),**

EQ\_NAME VARCHAR**(**50**),**

EQ\_QTY DECIMAL**(**10**,**2**)** **NOT** **NULL,**

EQ\_DESC TINYTEXT**,**

**PRIMARY** **KEY** **(**EQ\_CODE**));**

**CREATE** **TABLE** EQUIPMENT\_DET **(**

EQ\_ID VARCHAR**(**20**),**

EQ\_CODE VARCHAR**(**24**),**

EQ\_STATUS VARCHAR**(**8**),**

RENT\_CNT SMALLINT**,**

**PRIMARY** **KEY** **(**EQ\_ID**),**

**FOREIGN** **KEY** **(**EQ\_STATUS**)** **REFERENCES** STATUS\_MSTR **(**STATUS**)** **ON** **DELETE** **CASCADE,**

**FOREIGN** **KEY** **(**EQ\_CODE**)** **REFERENCES** EQUIPMENT\_MSTR **(**EQ\_CODE**)** **ON** **DELETE** **CASCADE);**

**CREATE** **TABLE** SALES\_MSTR **(**

SO\_ID VARCHAR**(**20**),**

SO\_DT DATE**,**

**PRIMARY** **KEY** **(**SO\_ID**));**

**CREATE** **TABLE** SALES\_DET **(**

SO\_ID VARCHAR**(**20**),**

SOD\_LINE INTEGER**,**

EQ\_ID VARCHAR**(**20**),**

SOD\_UNIT\_PRICE DECIMAL**(**10**,**2**)** **NOT** **NULL,**

SOD\_QTY DECIMAL**(**10**,**2**)** **NOT** **NULL,**

**PRIMARY** **KEY** **(**SO\_ID**,** SOD\_LINE**),**

**FOREIGN** **KEY** **(**EQ\_ID**)** **REFERENCES** EQUIPMENT\_DET **(**EQ\_ID**)** **ON** **DELETE** **CASCADE,**

**FOREIGN** **KEY** **(**SO\_ID**)** **REFERENCES** SALES\_MSTR **(**SO\_ID**)** **ON** **DELETE** CASCADES**);**

**CREATE** **TABLE** CUSTOMER\_MSTR **(**

CUST\_ID VARCHAR**(**20**),**

CUST\_NAME VARCHAR**(**50**),**

CUST\_PHONE VARCHAR**(**16**),**

CUST\_ADDR VARCHAR**(**100**),**

CUST\_EMAIL VARCHAR**(**30**),**

OVERDUE\_AMT DECIMAL**(**10**,**2**),**

**PRIMARY** **KEY** **(**CUST\_ID**));**

**CREATE** **TABLE** RESERVE\_MSTR **(**

RESERVE\_ID VARCHAR**(**20**),**

CUST\_ID VARCHAR**(**20**),**

PICK\_DT DATETIME**,**

RETURN\_DT DATETIME**,**

**PRIMARY** **KEY** **(**RESERVE\_ID**),**

**FOREIGN** **KEY** **(**CUST\_ID**)** **REFERENCES** CUSTOMER\_MSTR**(**CUST\_ID**)** **ON** **DELETE** **CASCADE);**

**CREATE** **TABLE** RESERVE\_DET **(**

RESERVE\_ID VARCHAR**(**20**),**

EQ\_CODE VARCHAR**(**24**),**

RESERVE\_QTY DECIMAL**(**10**,**2**),**

**PRIMARY** **KEY** **(**RESERVE\_ID**,** EQ\_CODE**),**

**FOREIGN** **KEY** **(**RESERVE\_ID**)** **REFERENCES** RESERVE\_MSTR **(**RESERVE\_ID**)** **ON** **DELETE** **CASCADE,**

**FOREIGN** **KEY** **(**EQ\_CODE**)** **REFERENCES** EQUIPMENT\_MSTR **(**EQ\_CODE**)** **ON** **DELETE** **CASCADE);**

**CREATE** **TABLE** VENDOR\_MSTR **(**

VEND\_ID VARCHAR**(**20**),**

NAME VARCHAR**(**50**),**

ADDR TINYTEXT**,**

**PRIMARY** **KEY** **(**VEND\_ID**));**

**CREATE** **TABLE** EQUIPMENT\_REPAIR **(**

EQ\_CODE VARCHAR**(**24**),**

VEND\_ID VARCHAR**(**20**),**

REPAIR\_HRS SMALLINT**,**

**PRIMARY** **KEY** **(**EQ\_CODE**,** VEND\_ID**),**

**FOREIGN** **KEY** **(**EQ\_CODE**)** **REFERENCES** EQUIPMENT\_MSTR **(**EQ\_CODE**)** **ON** **DELETE** **CASCADE,**

**FOREIGN** **KEY** **(**VEND\_ID**)** **REFERENCES** VENDOR\_MSTR **(**VEND\_ID**)** **ON** **DELETE** **CASCADE);**

**CREATE** **TABLE** PRICE\_LIST\_MSTR **(**

EQ\_CODE VARCHAR**(**24**),**

EQ\_STATUS VARCHAR**(**8**),**

PRICE DECIMAL**(**10**,**2**),**

**PRIMARY** **KEY** **(**EQ\_CODE**,** EQ\_STATUS**),**

**FOREIGN** **KEY** **(**EQ\_CODE**)** **REFERENCES** EQUIPMENT\_MSTR **(**EQ\_CODE**)** **ON** **DELETE** **CASCADE,**

**FOREIGN** **KEY** **(**EQ\_STATUS**)** **REFERENCES** STATUS\_MSTR **(**STATUS**)** **ON** **DELETE** **CASCADE);**

**CREATE** **TABLE** RENT\_MSTR **(**

RENT\_ID VARCHAR**(**20**),**

RESERVE\_ID VARCHAR**(**20**),**

PICK\_DT\_ACTUAL DATETIME**,**

RETURN\_DT\_ACTUAL DATETIME**,**

IS\_DAMAGED\_RETURN BOOLEAN**,**

IS\_ONTIME\_RETURN BOOLEAN**,**

IS\_NO\_RETURN BOOLEAN**,**

**PRIMARY** **KEY** **(**RENT\_ID**),**

**FOREIGN** **KEY** **(**RESERVE\_ID**)** **REFERENCES** RESERVE\_MSTR **(**RESERVE\_ID**)** **ON** **DELETE** **CASCADE);**

**CREATE** **TABLE** RENT\_DET **(**

RENT\_ID VARCHAR**(**20**),**

EQ\_ID VARCHAR**(**20**),**

**PRIMARY** **KEY** **(**RENT\_ID**,** EQ\_ID**),**

**FOREIGN** **KEY** **(**RENT\_ID**)** **REFERENCES** RENT\_MSTR **(**RENT\_ID**)** **ON** **DELETE** **CASCADE,**

**FOREIGN** **KEY** **(**EQ\_ID**)** **REFERENCES** EQUIPMENT\_DET**(**EQ\_ID**)** **ON** **DELETE** **CASCADE);**

**CREATE** **TABLE** INVENTORY **(**

EQ\_CODE VARCHAR**(**24**),**

INV\_STATUS VARCHAR**(**8**),**

INV\_QTY DECIMAL**(**10**,**2**),**

**PRIMARY** **KEY** **(**EQ\_CODE**,** INV\_STATUS**),**

**FOREIGN** **KEY** **(**EQ\_CODE**)** **REFERENCES** EQUIPMENT\_MSTR **(**EQ\_CODE**)** **ON** **DELETE** **CASCADE,**

**FOREIGN** **KEY** **(**INV\_STATUS**)** **REFERENCES** STATUS\_MSTR **(**STATUS**)** **ON** **DELETE** **CASCADE);**

**CREATE** **TABLE** PURCHASE\_MSTR **(**

PO\_ID VARCHAR**(**20**),**

VEND\_ID VARCHAR**(**20**),**

PO\_DT DATETIME**,**

DELIVERY\_DT DATETIME**,**

**PRIMARY** **KEY** **(**PO\_ID**),**

**FOREIGN** **KEY** **(**VEND\_ID**)** **REFERENCES** VENDOR\_MSTR **(**VEND\_ID**)** **ON** **DELETE** **CASCADE);**

**CREATE** **TABLE** PURCHASE\_DET **(**

PO\_ID VARCHAR**(**20**),**

POD\_LINE INTEGER**,**

EQ\_CODE VARCHAR**(**24**)** **NOT** **NULL,**

POD\_UNIT\_PRICE DECIMAL**(**10**,**2**),**

POD\_QTY DECIMAL**(**10**,**2**),**

**PRIMARY** **KEY** **(**PO\_ID**,** POD\_LINE**),**

**FOREIGN** **KEY** **(**PO\_ID**)** **REFERENCES** PURCHASE\_MSTR **(**PO\_ID**),**

**FOREIGN** **KEY** **(**EQ\_CODE**)** **REFERENCES** EQUIPMENT\_MSTR **(**EQ\_CODE**)** **ON** **DELETE** **CASCADE);**

# Assumptions

1. Rent can be taken only when we have a reservation.
2. Each Rent is collected against a SINGLE reservation.
3. It is possible for each reservation to reserve different equipment types, but they can only be collected or returned on the same date.
4. Previous rentals that were returned damaged will result in an increase of 25% on the deposit. Damaged records can be found in Rent IDs, which makes counting for the increment on deposit based on Rent IDs.
5. Our schema allows us to ask different suppliers to repair a piece of equipment, even though it is not ordered from them. For example, each type of equipment will have a standard repair time determined by a table provided by a supplier. To find out how long a repair will take, we can select the type of equipment and see what the time is listed by each vendor.
6. We deduct the late fee from the equipment deposit once the item is returned late. The deposit will be retained until the inspection results are available. In the event that the product is damaged, we must calculate the repair fee and deduct it from the deposit. If the deposit is not enough to cover the entire repair fee, we must record the amount owed in the CUSTOMER\_MSTR table and inform the customer. Prior to the next reservation, the customer must pay the owed amount.

# Feedback from Peers

## Group 45:s

* **Feedback for Data Modelling:**

1. The logical ER is not provided, per first question of the assignment

there should be a seperate logical ER diagram before arriving at the final fully normalized diagram.

2. The type information is not provided in the EER diagram.

3. It would be better if you consider to explain how each table is normalized.

4. DamageDeposit might have to be included in RENT\_MSTR as it depends on the history of each customer.

* **Feedback for Process Modelling:**

1. FDD:

Good flow of process in FDD, easy to understand.

1. DFD:
   * The DFD context level is explained properly but for external entity tax department and manager there must be at least one input and one output flow, according to the rules of DFD.
   * Good explanation of DFD system level-1 for all the process but databases should have one input and one output connectivity to satisfy the condition of database.
2. UML:

* The change Stock status and Increment Rental count can be included along with the Check equipment availability use case.

## Group 49:

OBSERVATIONS AND SUGGESTIONS:

FDD AND DATA FLOW DIAGRAM:

* A few processes which are merely data routing processes such as print report(s) are mentioned in the Functional decomposition diagram which can be removed to show a simpler FDD with more focus on core activities
* Context level diagram is excellent in terms of data flow and very clear
* Process Level – few data flows can be seen directly emerging from the data store and interact with the external entities which violates the rule – data flow can happen through a process with the external entities as the process is link between the two

**Use case:**

* The heading should be 'UML Use case diagram'
* Probably, only the customer should provide the customer details
* Equipment type should be selected in the system by the employee as the employee is the user of the system

**Activity diagram:**

* The diagram seems well made and presentable
* However, it's not mentioned which process the activity diagram is for. Mentioning that would help understand the diagram better

ENTITY RELATIONSHIP DIAGARM:

* Sorry for not being able to find complete clarity on “INV\_STATUS” and “EQ\_STATUS”, I would suggest recording a single equipment status column since it’s a repetition which is causing a confusion (to me at least).
* Consider recording the damaged amount or the amount owed by customers (sorry if I’ve missed out if it’s already added).
* Customer reservation is done based on equipment type (as per the case study), please consider recording which equipment type is being reserved.

## Consideration of Feedback:

1. According to the feedback from Group 45, we chose to include a simple logical ERM for the solution of B1. Earlier, we had used a Normalised ERM diagram for B1 (now Figure 8).
2. Common feedback between Group45 and 49 was that we missed providing an explanation to some of the solutions above. We have added these explanations based on the feedback.
3. Feedback from Group 49 about Process Level DFD does not seem to be correct as there are no External Entities interacting directly with any Data Store in out initial draft. Hence, no change required on it.
4. We have updated the UML Use Case diagram based on the feedback from Group 49 that Customer must be the actor to provide the customer details without interaction of the Employee. Earlier, Customer and Employee were interacting for the process of “Provide Customer Details”.

1. \* http://www.nuigalway.ie/celt/teachinglearning/academichonesty/ [↑](#footnote-ref-1)