OLLSCOIL NA GAILLIMHE
UNIVERSITY OF GALWAY

Assignment Submission Form

Group Number: 51

Assignment Details

Module Code: MS804 & MS805 **Assignment Title:** A Database Design Case: Teton Whitewater Kayak

Group Members:

Student ID	Programme	Student Name	Contact Details (Email, Telephone)
22225743	2022-2023	Dilip Venkatesan Sankar	D.Sankar1@nuigalway.ie
22220902	2022-2023	Mrinal Namdeo	m.namdeo1@nuigalway.ie
22220131	2022-2023	Sayantan Paul	S.Paul9@nuigalway.ie

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*[*] 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:** 11/11/2022

Dilip Venkatesan Sankar

Mrinal Namdeo

Sayantan Paul

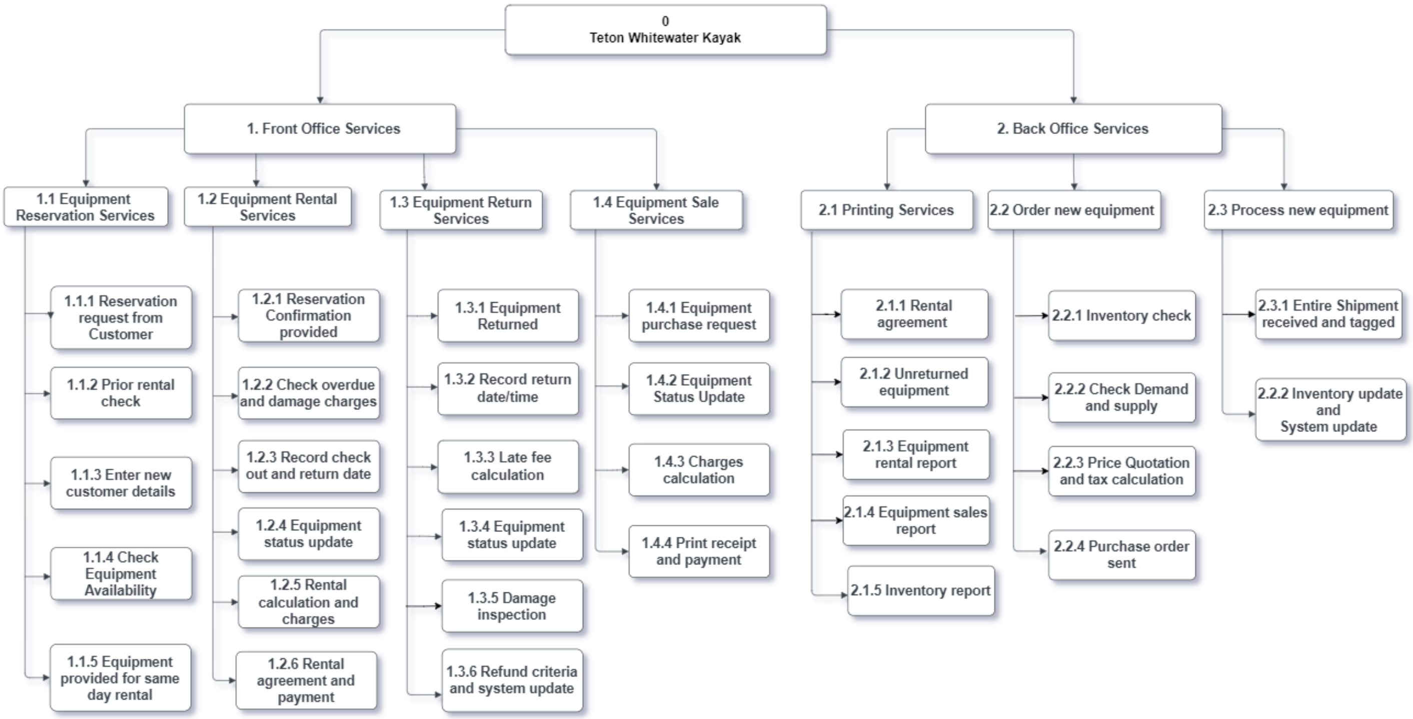
[*] http://www.nuigalway.ie/celt/teachinglearning/academichonesty/

Table of Contents

Section A: Process Modelling

Section B: Data Modelling

1. Functional Decomposition Diagram (FDD) showing the hierarchy of all processes in the system

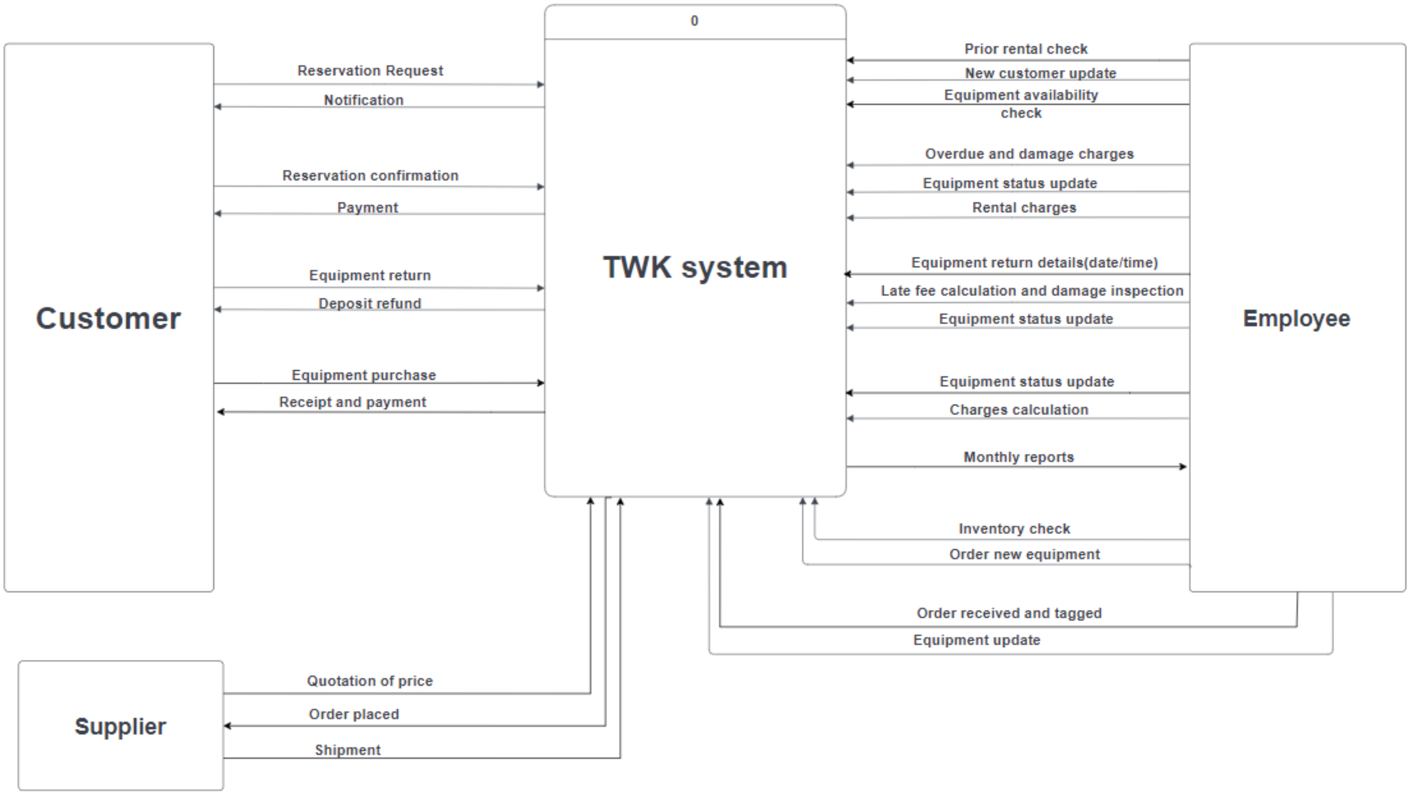

0
Teton Whitewater Kayak
1. Front Office Services
2. Back Office Services
1.1 Equipment Reservation Services
1.2 Equipment Rental Services
1.3 Equipment Return Services
1.4 Equipment Sale Services
2.1 Printing Services
2.2 Order new equipment
2.3 Process new equipment
1.1.1 Reservation request from Customer
1.1.2 Prior rental check
1.1.3 Enter new customer details
1.1.4 Check Equipment Availability
1.1.5 Equipment provided for same day rental
1.2.1 Reservation Confirmation provided
1.2.2 Check overdue and damage charges
1.2.3 Record check out and return date
1.2.4 Equipment status update
1.2.5 Rental calculation and charges
1.2.6 Rental agreement and payment
1.3.1 Equipment Returned
1.3.2 Record return date/time
1.3.3 Late fee calculation
1.3.4 Equipment status update
1.3.5 Damage inspection
1.3.6 Refund criteria and system update
1.4.1 Equipment purchase request
1.4.2 Equipment Status Update
1.4.3 Charges calculation
1.4.4 Print receipt and payment
2.1.1 Rental agreement
2.1.2 Unreturned equipment
2.1.3 Equipment rental report
2.1.4 Equipment sales report
2.1.5 Inventory report
2.2.1 Inventory check
2.2.2 Check Demand and supply
2.2.3 Price Quotation and tax calculation
2.2.4 Purchase order sent
2.3.1 Entire Shipment received and tagged
2.2.2 Inventory update and System update

The functional Decomposition diagram illustrates the overall process in a system as well as the sub-process related to each process respectively covering the top-down format
The Teton Whitewater Kayak system contains 2 main processes,

1. Front Office Services
   - Equipment Reservation Services
   - Equipment Rental Services
   - Equipment Return Services
   - Equipment Sale Services
2. Back Office Services
   - 2.1 Printing Services
   - 2.2 Order new equipment
   - 2.3 Process new equipment

And the above main processes consist of multiple sub-processes as seen in the FDD.

Context-Level Data Flow Diagram (DFD) -Level 0



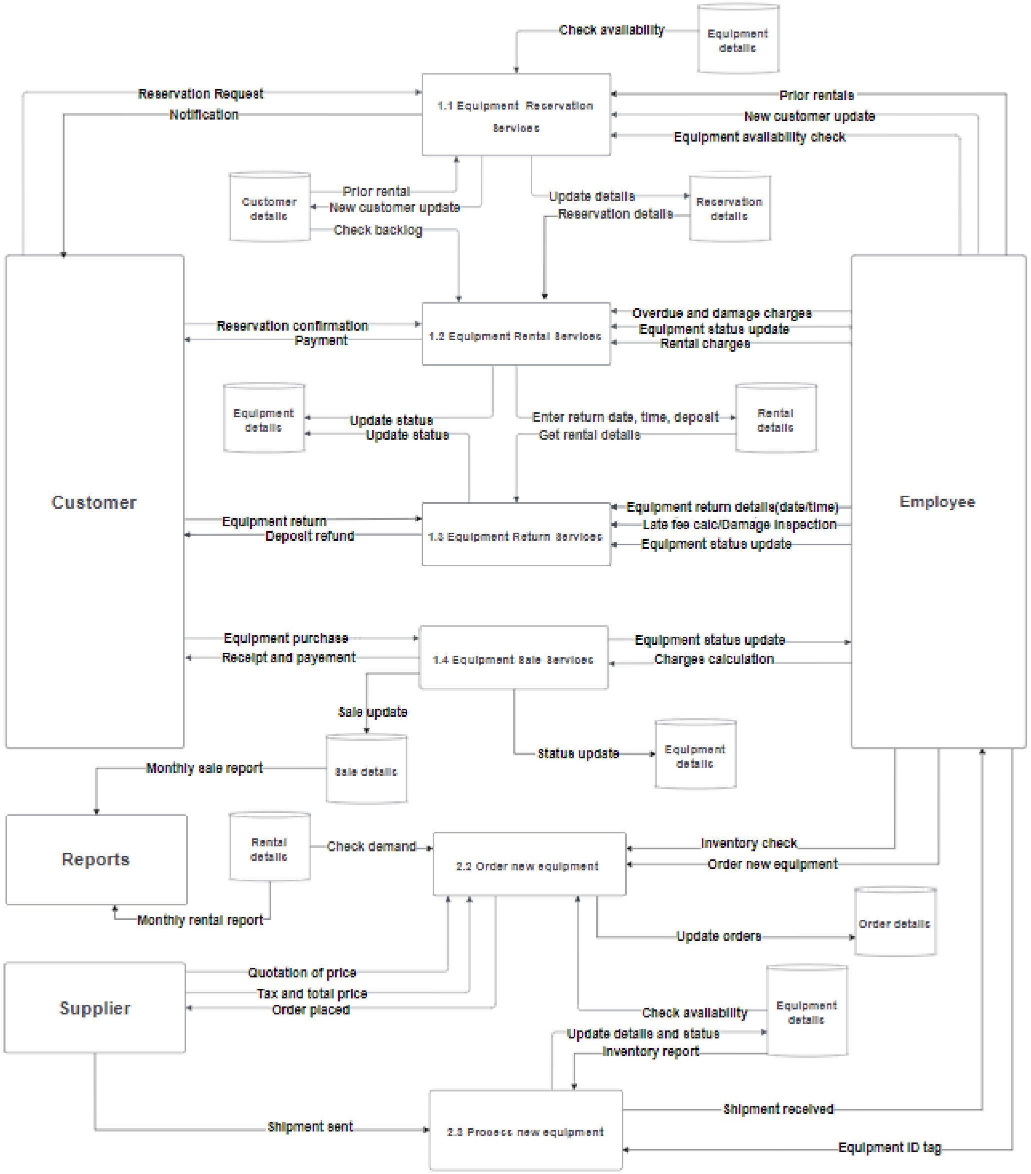
In level0 Context Level DFD, we have 2 main entities,

1. Customer
2. Employee

- In our procedure, the customer submits a request for a reservation, and an employee either verifies the system for existing records or inputs new customer information.
- The employee check for the availability of the equipment and we have assumed that the system sends a notification to the customer.
- The customer submits the reservation request, and the employee investigates the customer's history, updates the equipment's status in the system, and informs the client of the rental fees.
- If a customer wants to buy something, a purchase request is made, the employee updates the status of the equipment, and then receipt and payment notice are issued.
- A monthly report (Inventory, Sales, Rentals) is created and sent to the employee from the system
- Based on the inventory data and demand for the equipment, the employee requests new equipment from the supplier.
- The supplier offers a quote before sending the shipment. The employee tags newly acquired equipment and changes the system's status.

Note: We have considered customer notification in the reservation process if it is a same day rental and we have not considered Supplier as an entity.

Systems-level Data Flow Diagram (DFD) – Level1

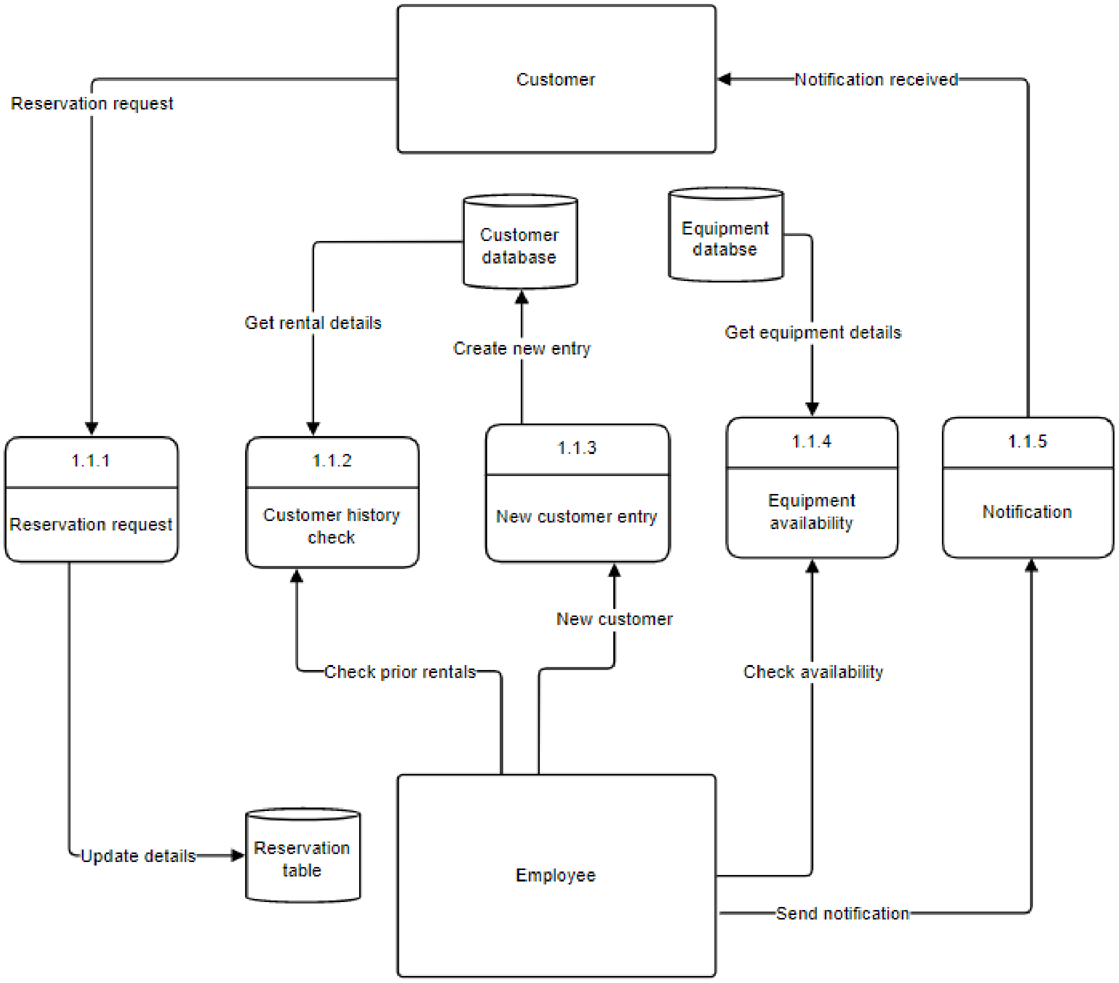


In the Level1 system level DFD, there will not be any changes related to the input and output in all the process compared to level0, but we have enhanced by adding databases and table.

- The employee updates or checks the customer table for prior and new customer information when a reservation request is submitted. He also looks at the equipment table to make sure it's available
- After submitting the reservation confirmation, the employee checks the customer tables for backlogs and updates the rental table with the revised date, time, and equipment information.
- As the equipment is being returned, the staff checks the rental information, determines the charge based on the information, changes the equipment tables, and requests the customer for payment.

- Equipment is regularly retired and put up for sale. When a purchase request is made, the employee updates the equipment table and sales table with the purchase details, but no customer information is kept.
- Based on demand and price, the employee periodically orders new equipment from the supplier and updates the order table. Once the shipment is received, the new equipment is tagged, and the equipment table is updated.

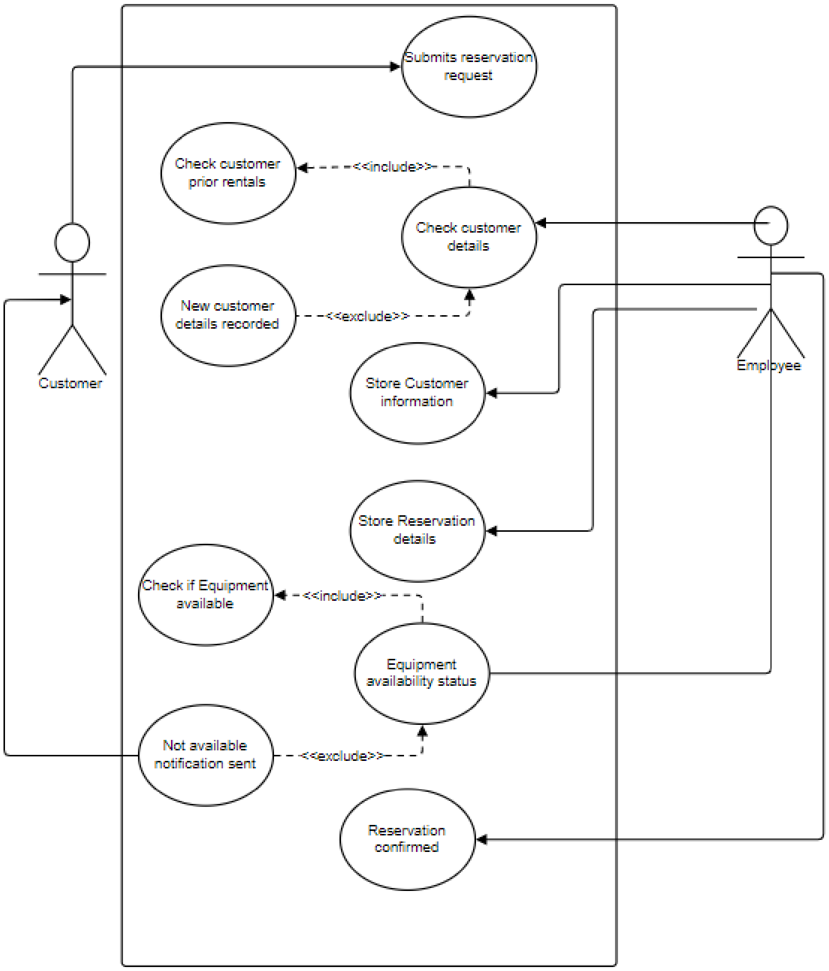
Systems-level Data Flow Diagram (DFD) – Level2



In level2 DFD we have taken Reservation request into account and explained in detail.

- When the customer submits a reservation request, the request is recorded in the reservation table and the employee checks prior rents in the Customer table and if it's a new customer he updates a new entry to the customer table.
- Employee checks for equipment availability from the table and send a notification to the customer after the reservation.
- Here we have considered notification as an extra process and incorporated in order to let the customer know about the reservation status if it is a same day rental.

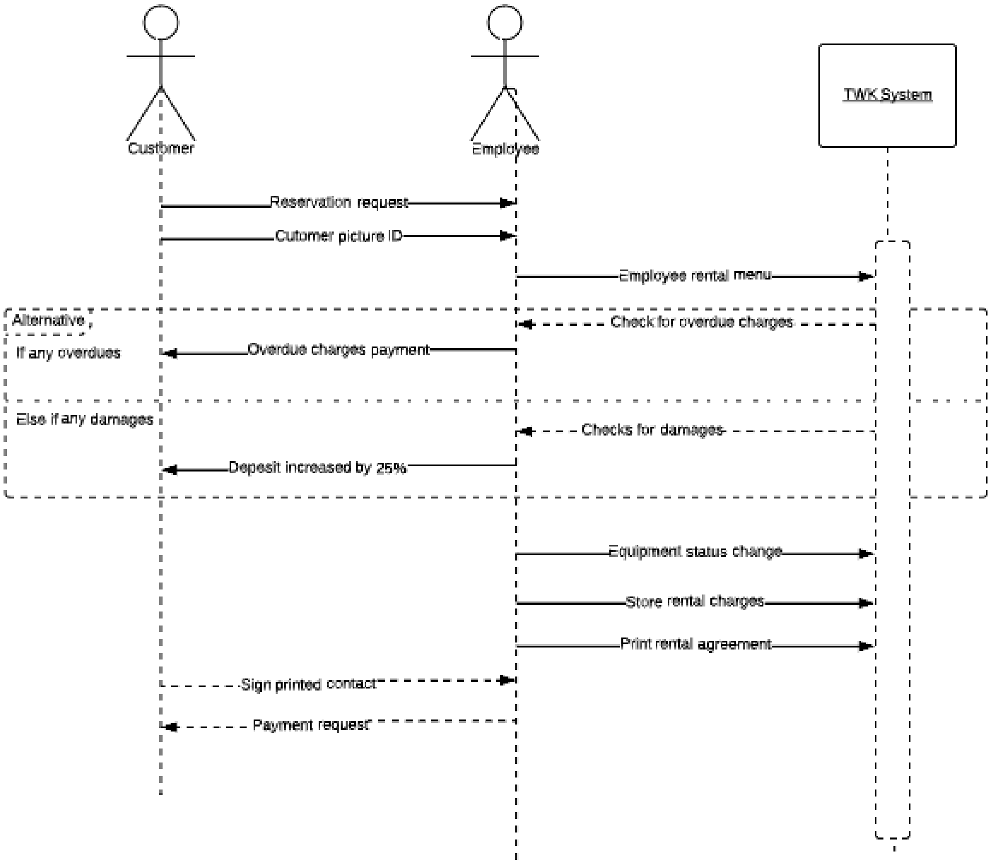
UML Use Case diagram for equipment reservation process


Submits reservation request
Check customer prior rentals
<<include>>
Check customer details
New customer details recorded
<<exclude>>
Store Customer information
Store Reservation details
Check if Equipment available
<<include>>
Equipment availability status
Not available notification sent
<<exclude>>
Reservation confirmed
Customer
Employee

In the UML use case diagram, we have shown what are the actions performed by two actors i.e., employee and customer in order to complete an equipment reservation process.

- After the customer submits his request for a reservation, the employee analyzes his information to determine whether he is an existing client who has previously rented equipment, or if he is a brand-new client for whom a new entry will be made.
- For each reservation, the employee stores customer information like name, address, telephone, and e-mail address.
- The employee then inputs the reservation information, including the kind of equipment, the pickup date, the return date, and the deposit provided by the client.
- The employee checks the equipment availability status, if the equipment is available and if its same day rental then the equipment is provided immediately. If not, the customer is notified that the equipment is not available.

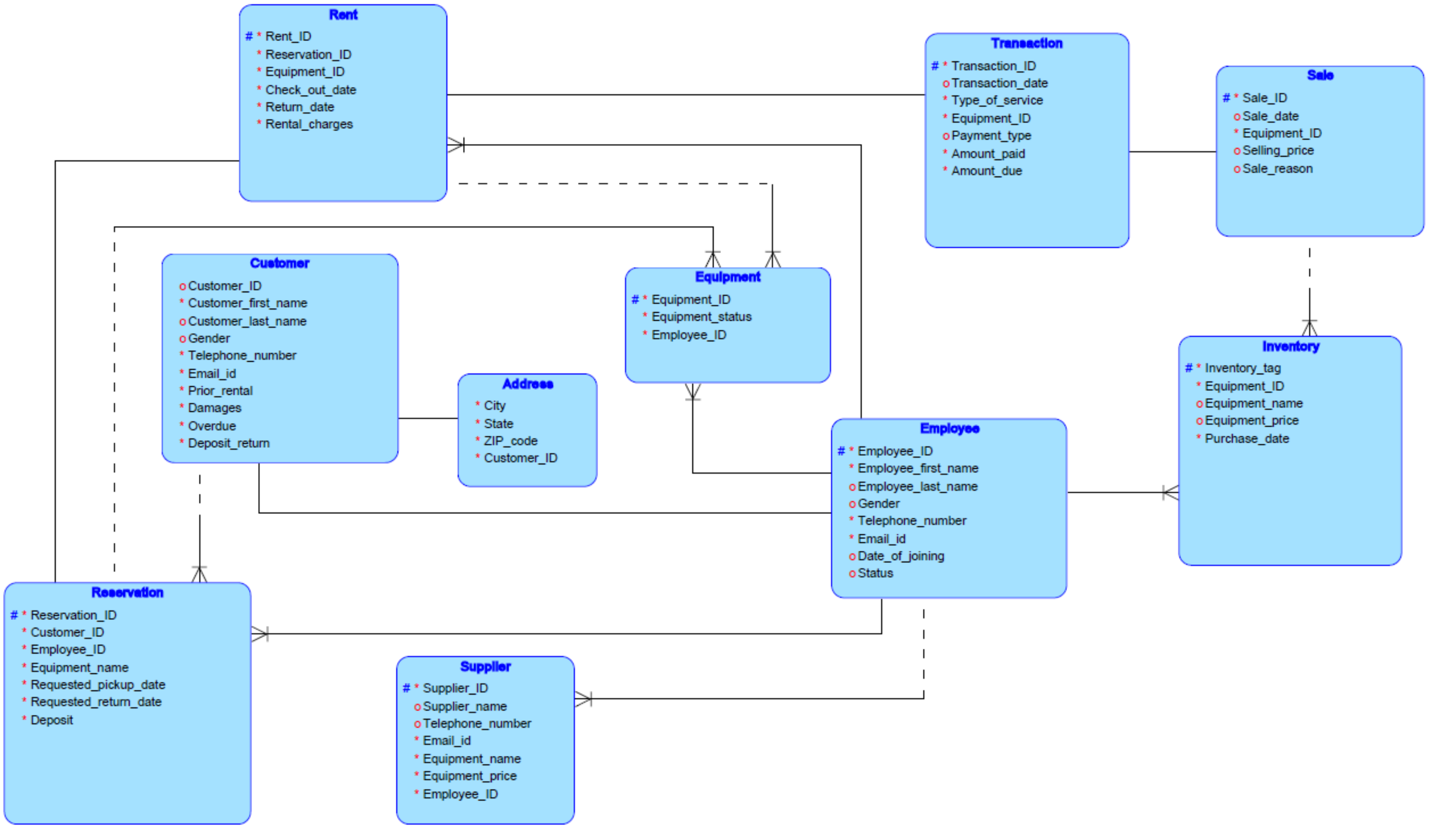
UML sequence diagram for Equipment Reservation process



In UML sequence diagram, the equipment rental process is explained.

- The customer after making a reservation provides the reservation request and a picture ID which is preferably a hardcopy to the employee.
- The employee chooses the rental options and checks for overdue fees and damages; if either is present, the customer is asked to pay the fees before to renting, and the deposit is raised by 25% in the event of damage, respectively.
- Moving forward, the employee enters details and changes the status of the equipment, calculate rental charges and prints agreement.
- The customer signs the contract, and a payment request is sent to the customer.

Logical Entity-Relationship Model (ERM)



Explanation:

The entity relationship model is a high-level conceptual model which shows the relationship of entities stored in a database.

In the ERM we can see the below,

- Entities
- Attributes
- Relationships

In our ERM which involves Teton Whitewater Kayak, we have considered the below entities,

1. Customer
2. Employee
3. Reservation
4. Rent
5. Equipment
6. Supplier
7. Address
8. Sale
9. Inventory
10. Transaction

Based on our understanding we have considered the below relationships,

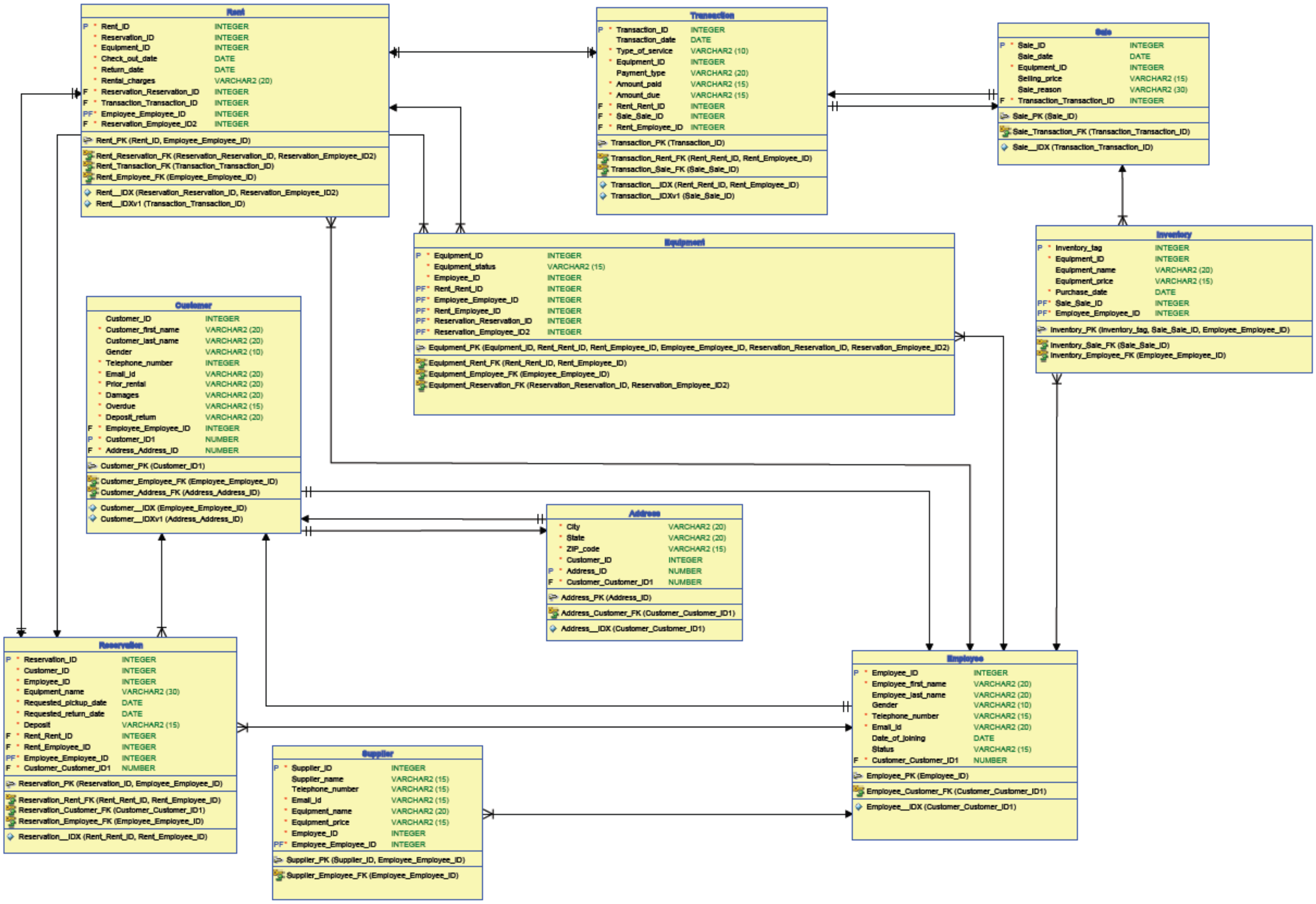
1. One to One
2. One to Many
3. Optional

Flow:

- In Teton Whitewater Kayak system, we have considered a customer can interact with one employee to enquire about the reservation process of an equipment **(One to One relationship)**
- After the inquiry, the customer may reserve one or more equipment's of his/her choice **(Optional One to Many)**
- Once the reservation requests are placed, one employee can work on either one or many reservations, but more than one employee cannot work on the same or many requests which will cause duplicate work **(One to Many)**
- Before the reservation request is confirmed, employee checks the equipment availability **(One to Many)**
- After the reservation is successful, we have considered that with one reservation request one rental request can be processed **(One to One)** but in a single request one or more equipment's can be processed which is explained later on.
- In order to rent an equipment, the customer provides the reservation request to the employee, but it can be optional because we have considered the customer might change his mind not to rent even after reservation, so we have used optional relationship **(Optional One to Many)**
- In order to process the rental request, one employee can work on either one or many request **(One to many)**
- When Teton Whitewater Kayak periodically retires equipment's for sale, Customer's may purchase one or more equipment's which is optional, but we haven't stored the customer details in the database when a sale takes place **(Optional One to Many)**
- And during the sale or rent of equipment's we have considered only on transaction takes place for billing for each customer **(One to One)**
- Teton Whitewater Kayak also purchases new equipment periodically based one demand, theft, and damage. In this case, one employee can interact with many suppliers to get the quotation but many supplier's cannot interact with employee's in the same company **(One to Many)**
- When the stock is sent from the supplier as a whole, one employee updates the inventory report and tag's much equipment. Many employees cannot work on the same or more inventory updates as it can cause duplicate values **(One to many)**

Note: All the above flow and relationship are based on the Teton Whitewater Kayak as well as assumptions.

Logical ERM into a fully normalized relational database schema


Rent
P * Rent_ID INTEGER
* Reservation_ID INTEGER
* Equipment_ID INTEGER
* Check_out_date DATE
* Return_date DATE
* Rental_charges VARCHAR2 (20)
F * Reservation_Reservation_ID INTEGER
F * Transaction_Transaction_ID INTEGER
PF* Employee_Employee_ID INTEGER
F * Reservation_Employee_ID2 INTEGER
Rent_PK (Rent_ID, Employee_Employee_ID)
Rent_Reservation_FK (Reservation_Reservation_ID, Reservation_Employee_ID2)
Rent_Transaction_FK (Transaction_Transaction_ID)
Rent_Employee_FK (Employee_Employee_ID)
Rent__IDX (Reservation_Reservation_ID, Reservation_Employee_ID2)
Rent__IDXv1 (Transaction_Transaction_ID)
Transaction
P * Transaction_ID INTEGER
Transaction_date DATE
* Type_of_service VARCHAR2 (10)
* Equipment_ID INTEGER
Payment_type VARCHAR2 (20)
* Amount_paid VARCHAR2 (15)
* Amount_due VARCHAR2 (15)
F * Rent_Rent_ID INTEGER
F * Sale_Sale_ID INTEGER
F * Rent_Employee_ID INTEGER
Transaction_PK (Transaction_ID)
Transaction_Rent_FK (Rent_Rent_ID, Rent_Employee_ID)
Transaction_Sale_FK (Sale_Sale_ID)
Transaction__IDX (Rent_Rent_ID, Rent_Employee_ID)
Transaction__IDXv1 (Sale_Sale_ID)
Sale
P * Sale_ID INTEGER
Sale_date DATE
* Equipment_ID INTEGER
Selling_price VARCHAR2 (15)
Sale_reason VARCHAR2 (30)
F * Transaction_Transaction_ID INTEGER
Sale_PK (Sale_ID)
Sale_Transaction_FK (Transaction_Transaction_ID)
Sale__IDX (Transaction_Transaction_ID)
Equipment
P * Equipment_ID INTEGER
* Equipment_status VARCHAR2 (15)
* Employee_ID INTEGER
PF* Rent_Rent_ID INTEGER
PF* Employee_Employee_ID INTEGER
PF* Rent_Employee_ID INTEGER
PF* Reservation_Reservation_ID INTEGER
PF* Reservation_Employee_ID2 INTEGER
Equipment_PK (Equipment_ID, Rent_Rent_ID, Rent_Employee_ID, Employee_Employee_ID, Reservation_Reservation_ID, Reservation_Employee_ID2)
Equipment_Rent_FK (Rent_Rent_ID, Rent_Employee_ID)
Equipment_Employee_FK (Employee_Employee_ID)
Equipment_Reservation_FK (Reservation_Reservation_ID, Reservation_Employee_ID2)
Inventory
P * Inventory_tag INTEGER
* Equipment_ID INTEGER
Equipment_name VARCHAR2 (20)
Equipment_price VARCHAR2 (15)
* Purchase_date DATE
PF* Sale_Sale_ID INTEGER
PF* Employee_Employee_ID INTEGER
Inventory_PK (Inventory_tag, Sale_Sale_ID, Employee_Employee_ID)
Inventory_Sale_FK (Sale_Sale_ID)
Inventory_Employee_FK (Employee_Employee_ID)
Customer
Customer_ID INTEGER
* Customer_first_name VARCHAR2 (20)
Customer_last_name VARCHAR2 (20)
Gender VARCHAR2 (10)
* Telephone_number INTEGER
* Email_Id VARCHAR2 (20)
* Prior_rental VARCHAR2 (20)
* Damages VARCHAR2 (20)
* Overdue VARCHAR2 (15)
* Deposit_return VARCHAR2 (20)
F * Employee_Employee_ID INTEGER
P * Customer_ID1 NUMBER
F * Address_Address_ID NUMBER
Customer_PK (Customer_ID1)
Customer_Employee_FK (Employee_Employee_ID)
Customer_Address_FK (Address_Address_ID)
Customer__IDX (Employee_Employee_ID)
Customer__IDXv1 (Address_Address_ID)
Address
* City VARCHAR2 (20)
* State VARCHAR2 (20)
* ZIP_code VARCHAR2 (15)
* Customer_ID INTEGER
P * Address_ID NUMBER
F * Customer_Customer_ID1 NUMBER
Address_PK (Address_ID)
Address_Customer_FK (Customer_Customer_ID1)
Address__IDX (Customer_Customer_ID1)
Reservation
P * Reservation_ID INTEGER
* Customer_ID INTEGER
* Employee_ID INTEGER
* Equipment_name VARCHAR2 (30)
* Requested_pickup_date DATE
* Requested_return_date DATE
* Deposit VARCHAR2 (15)
F * Rent_Rent_ID INTEGER
F * Rent_Employee_ID INTEGER
PF* Employee_Employee_ID INTEGER
F * Customer_Customer_ID1 NUMBER
Reservation_PK (Reservation_ID, Employee_Employee_ID)
Reservation_Rent_FK (Rent_Rent_ID, Rent_Employee_ID)
Reservation_Customer_FK (Customer_Customer_ID1)
Reservation_Employee_FK (Employee_Employee_ID)
Reservation__IDX (Rent_Rent_ID, Rent_Employee_ID)
Supplier
P * Supplier_ID INTEGER
Supplier_name VARCHAR2 (15)
Telephone_number VARCHAR2 (15)
* Email_Id VARCHAR2 (15)
* Equipment_name VARCHAR2 (20)
* Equipment_price VARCHAR2 (15)
* Employee_ID INTEGER
PF* Employee_Employee_ID INTEGER
Supplier_PK (Supplier_ID, Employee_Employee_ID)
Supplier_Employee_FK (Employee_Employee_ID)
Employee
P * Employee_ID INTEGER
* Employee_first_name VARCHAR2 (20)
Employee_last_name VARCHAR2 (20)
Gender VARCHAR2 (10)
* Telephone_number VARCHAR2 (15)
* Email_Id VARCHAR2 (20)
Date_of_joining DATE
Status VARCHAR2 (15)
F * Customer_Customer_ID1 NUMBER
Employee_PK (Employee_ID)
Employee_Customer_FK (Customer_Customer_ID1)
Employee__IDX (Customer_Customer_ID1)

Normalization

1st Normal Form:

- All the columns in the tables are unique and there are no duplicates
- Each cell contains a single value and are non-divisible
- There are no null values for the primary key and foreign key. To corroborate, we have Customer_ID in the customer table which is a primary key and it has been referred as a foreign key in the Address table which has been created as it has multiple values.

2nd Normal Form:

- We are moving to 2nd Normal Form as 1st Normal Form is satisfied
- We do not have any composite key due to which there is no partial dependencies

3rd Normal Form:

- Both 2nd Normal Form and 1st Normal Form are satisfied, hence we are moving to 3rd Normal Form
- We do not have any transitive dependency
- All fields are determinable by the primary key. To corroborate, we can get the customer details from the Customer table based on the Customer_ID which is a primary key.

SQL Create Table Statements:

```
-- Generated by Oracle SQL Developer Data Modeler 22.2.0.165.1149
--   at:        2022-11-10 21:54:05 GMT
--   site:      Oracle Database 11g
--   type:      Oracle Database 11g



-- predefined type, no DDL - MDSYS.SDO_GEOMETRY

-- predefined type, no DDL - XMLTYPE

CREATE TABLE address (
    city                  VARCHAR2(20) NOT NULL,
    state                 VARCHAR2(20) NOT NULL,
    zip_code              VARCHAR2(15) NOT NULL,
    customer_id           INTEGER NOT NULL,
    address_id            NUMBER NOT NULL,
    customer_customer_id1 NUMBER NOT NULL
);


CREATE UNIQUE INDEX address__idx ON
    address (
        customer_customer_id1
    ASC );


ALTER TABLE address ADD CONSTRAINT address_pk PRIMARY KEY ( address_id );

CREATE TABLE customer (
    customer_id          INTEGER,
    customer_first_name  VARCHAR2(20) NOT NULL,
    customer_last_name   VARCHAR2(20),
    gender               VARCHAR2(10),
    telephone_number     INTEGER NOT NULL,
    email_id             VARCHAR2(20) NOT NULL,
    prior_rental         VARCHAR2(20) NOT NULL,
    damages              VARCHAR2(20) NOT NULL,
    overdue              VARCHAR2(15) NOT NULL,
    deposit_return       VARCHAR2(20) NOT NULL,
    employee_employee_id INTEGER NOT NULL,
    customer_id1         NUMBER NOT NULL,
    address_address_id   NUMBER NOT NULL
);

CREATE UNIQUE INDEX customer__idx ON
    customer (
        employee_employee_id
    ASC );


CREATE UNIQUE INDEX customer__idxv1 ON
    customer (
        address_address_id
    ASC );


ALTER TABLE customer ADD CONSTRAINT customer_pk PRIMARY KEY ( customer_id1 );
```

```
CREATE TABLE employee (
    employee_id           INTEGER NOT NULL,
    employee_first_name   VARCHAR2(20) NOT NULL,
    employee_last_name    VARCHAR2(20),
    gender                VARCHAR2(10),
    telephone_number      VARCHAR2(15) NOT NULL,
    email_id              VARCHAR2(20) NOT NULL,
    date_of_joining       DATE,
    status                VARCHAR2(15),
    customer_customer_id1 NUMBER NOT NULL
);


CREATE UNIQUE INDEX employee__idx ON
    employee (
        customer_customer_id1
    ASC );

ALTER TABLE employee ADD CONSTRAINT employee_pk PRIMARY KEY ( employee_id );
```

```
CREATE TABLE equipment (
    equipment_id               INTEGER NOT NULL,
    equipment_status           VARCHAR2(15) NOT NULL,
    employee_id                INTEGER NOT NULL,
    rent_rent_id               INTEGER NOT NULL,
    employee_employee_id       INTEGER NOT NULL,
    rent_employee_id           INTEGER NOT NULL,
    reservation_reservation_id INTEGER NOT NULL,
    reservation_employee_id2   INTEGER NOT NULL
);


ALTER TABLE equipment
    ADD CONSTRAINT equipment_pk PRIMARY KEY ( equipment_id,
                                              rent_rent_id,
                                              rent_employee_id,
                                              employee_employee_id,
                                              reservation_reservation_id,
                                              reservation_employee_id2 );
```

```
CREATE TABLE inventory (
    inventory_tag        INTEGER NOT NULL,
    equipment_id         INTEGER NOT NULL,
    equipment_name       VARCHAR2(20),
    equipment_price      VARCHAR2(15),
    purchase_date        DATE NOT NULL,
    sale_sale_id         INTEGER NOT NULL,
    employee_employee_id INTEGER NOT NULL
);


ALTER TABLE inventory
    ADD CONSTRAINT inventory_pk PRIMARY KEY ( inventory_tag,
                                              sale_sale_id,
                                              employee_employee_id );
```

```
CREATE TABLE rent (
    rent_id                    INTEGER NOT NULL,
    reservation_id             INTEGER NOT NULL,
    equipment_id               INTEGER NOT NULL,
    check_out_date             DATE NOT NULL,
    return_date                DATE NOT NULL,
    rental_charges             VARCHAR2(20) NOT NULL,
    reservation_reservation_id INTEGER NOT NULL,
    transaction_transaction_id INTEGER NOT NULL,
    employee_employee_id       INTEGER NOT NULL,
    reservation_employee_id2   INTEGER NOT NULL
);
```

```
CREATE UNIQUE INDEX rent__idx ON
    rent (
        reservation_reservation_id
    ASC,
        reservation_employee_id2
    ASC );


CREATE UNIQUE INDEX rent__idxv1 ON
    rent (
        transaction_transaction_id
    ASC );


ALTER TABLE rent ADD CONSTRAINT rent_pk PRIMARY KEY ( rent_id,
                                                      employee_employee_id );
```

```
CREATE TABLE reservation (
    reservation_id        INTEGER NOT NULL,
    customer_id           INTEGER NOT NULL,
    employee_id           INTEGER NOT NULL,
    equipment_name        VARCHAR2(30) NOT NULL,
    requested_pickup_date DATE NOT NULL,
    requested_return_date DATE NOT NULL,
    deposit               VARCHAR2(15) NOT NULL,
    rent_rent_id          INTEGER NOT NULL,
    rent_employee_id      INTEGER NOT NULL,
    employee_employee_id  INTEGER NOT NULL,
    customer_customer_id1 NUMBER NOT NULL
);


CREATE UNIQUE INDEX reservation__idx ON
    reservation (
        rent_rent_id
    ASC,
        rent_employee_id
    ASC );


ALTER TABLE reservation ADD CONSTRAINT reservation_pk PRIMARY KEY ( reservation_id,
                                                                    employee_employee_id );
```

```
CREATE TABLE sale (
    sale_id                    INTEGER NOT NULL,
    sale_date                  DATE,
    equipment_id               INTEGER NOT NULL,
    selling_price              VARCHAR2(15),
    sale_reason                VARCHAR2(30),
    transaction_transaction_id INTEGER NOT NULL
);

CREATE UNIQUE INDEX sale__idx ON
    sale (
        transaction_transaction_id
    ASC );


ALTER TABLE sale ADD CONSTRAINT sale_pk PRIMARY KEY ( sale_id );
```

```
CREATE TABLE supplier (
    supplier_id          INTEGER NOT NULL,
    supplier_name        VARCHAR2(15),
    telephone_number     VARCHAR2(15),
    email_id             VARCHAR2(15) NOT NULL,
    equipment_name       VARCHAR2(20) NOT NULL,
    equipment_price      VARCHAR2(15) NOT NULL,
    employee_id          INTEGER NOT NULL,
    employee_employee_id INTEGER NOT NULL
);

ALTER TABLE supplier ADD CONSTRAINT supplier_pk PRIMARY KEY ( supplier_id,
                                                              employee_employee_id );
```

```
CREATE TABLE transaction (
    transaction_id   INTEGER NOT NULL,
    transaction_date DATE,
    type_of_service  VARCHAR2(10) NOT NULL,
    equipment_id     INTEGER NOT NULL,
    payment_type     VARCHAR2(20),
    amount_paid      VARCHAR2(15) NOT NULL,
    amount_due       VARCHAR2(15) NOT NULL,
    rent_rent_id     INTEGER NOT NULL,
    sale_sale_id     INTEGER NOT NULL,
    rent_employee_id INTEGER NOT NULL
);

CREATE UNIQUE INDEX transaction__idx ON
    transaction (
        rent_rent_id
    ASC,
        rent_employee_id
    ASC );

CREATE UNIQUE INDEX transaction__idxv1 ON
    transaction (
        sale_sale_id
    ASC );
```

```
ALTER TABLE transaction ADD CONSTRAINT transaction_pk PRIMARY KEY ( transaction_id );

ALTER TABLE address
    ADD CONSTRAINT address_customer_fk FOREIGN KEY ( customer_customer_id1 )
        REFERENCES customer ( customer_id1 );

ALTER TABLE customer
    ADD CONSTRAINT customer_address_fk FOREIGN KEY ( address_address_id )
        REFERENCES address ( address_id );

ALTER TABLE customer
    ADD CONSTRAINT customer_employee_fk FOREIGN KEY ( employee_employee_id )
        REFERENCES employee ( employee_id );

ALTER TABLE employee
    ADD CONSTRAINT employee_customer_fk FOREIGN KEY ( customer_customer_id1 )
        REFERENCES customer ( customer_id1 );


ALTER TABLE equipment
    ADD CONSTRAINT equipment_employee_fk FOREIGN KEY ( employee_employee_id )
        REFERENCES employee ( employee_id );
```

```
ALTER TABLE equipment
    ADD CONSTRAINT equipment_rent_fk FOREIGN KEY ( rent_rent_id,
                                                   rent_employee_id )
        REFERENCES rent ( rent_id,
                          employee_employee_id );


ALTER TABLE equipment
    ADD CONSTRAINT equipment_reservation_fk FOREIGN KEY ( reservation_reservation_id,
                                                          reservation_employee_id2 )
        REFERENCES reservation ( reservation_id,
                                 employee_employee_id );



ALTER TABLE inventory
    ADD CONSTRAINT inventory_employee_fk FOREIGN KEY ( employee_employee_id )

   REFERENCES employee ( employee_id );

ALTER TABLE inventory
    ADD CONSTRAINT inventory_sale_fk FOREIGN KEY ( sale_sale_id )
        REFERENCES sale ( sale_id );


ALTER TABLE rent
    ADD CONSTRAINT rent_employee_fk FOREIGN KEY ( employee_employee_id )
        REFERENCES employee ( employee_id );


ALTER TABLE rent
    ADD CONSTRAINT rent_reservation_fk FOREIGN KEY ( reservation_reservation_id,
                                                     reservation_employee_id2 )
        REFERENCES reservation ( reservation_id,
                                 employee_employee_id );


ALTER TABLE rent
    ADD CONSTRAINT rent_transaction_fk FOREIGN KEY ( transaction_transaction_id )
        REFERENCES transaction ( transaction_id );


ALTER TABLE reservation
    ADD CONSTRAINT reservation_customer_fk FOREIGN KEY ( customer_customer_id1 )
        REFERENCES customer ( customer_id1 );


ALTER TABLE reservation
    ADD CONSTRAINT reservation_employee_fk FOREIGN KEY ( employee_employee_id )
        REFERENCES employee ( employee_id );


ALTER TABLE reservation
    ADD CONSTRAINT reservation_rent_fk FOREIGN KEY ( rent_rent_id,
                                                     rent_employee_id )
        REFERENCES rent ( rent_id,
                          employee_employee_id );


ALTER TABLE sale
    ADD CONSTRAINT sale_transaction_fk FOREIGN KEY ( transaction_transaction_id )
        REFERENCES transaction ( transaction_id );



ALTER TABLE supplier
    ADD CONSTRAINT supplier_employee_fk FOREIGN KEY ( employee_employee_id )
        REFERENCES employee ( employee_id );
```

```
ALTER TABLE transaction
    ADD CONSTRAINT transaction_rent_fk FOREIGN KEY ( rent_rent_id,
                                                     rent_employee_id )
        REFERENCES rent ( rent_id,
                          employee_employee_id );


ALTER TABLE transaction
    ADD CONSTRAINT transaction_sale_fk FOREIGN KEY ( sale_sale_id )

       REFERENCES sale ( sale_id );

CREATE SEQUENCE address_address_id_seq START WITH 1 NOCACHE ORDER;


CREATE OR REPLACE TRIGGER address_address_id_trg BEFORE
    INSERT ON address
    FOR EACH ROW
    WHEN ( new.address_id IS NULL )
BEGIN
    :new.address_id := address_address_id_seq.nextval;
END;
/


CREATE SEQUENCE customer_customer_id1_seq START WITH 1 NOCACHE ORDER;


CREATE OR REPLACE TRIGGER customer_customer_id1_trg BEFORE
    INSERT ON customer
    FOR EACH ROW
    WHEN ( new.customer_id1 IS NULL )
BEGIN
    :new.customer_id1 := customer_customer_id1_seq.nextval;
END;
/
```

```
__________________________________________________________________________________________________________


-- Oracle SQL Developer Data Modeler Summary Report:
--
-- CREATE TABLE                            10
-- CREATE INDEX                            10
-- ALTER TABLE                             29
-- CREATE VIEW                              0
-- ALTER VIEW                               0
-- CREATE PACKAGE                           0
-- CREATE PACKAGE BODY                      0
-- CREATE PROCEDURE                         0
-- CREATE FUNCTION                          0
-- CREATE TRIGGER                           2
-- ALTER TRIGGER                            0
-- CREATE COLLECTION TYPE                   0


-- CREATE STRUCTURED TYPE                   0
-- CREATE STRUCTURED TYPE BODY              0
-- CREATE CLUSTER                           0
-- CREATE CONTEXT                           0
-- CREATE DATABASE                          0

-- CREATE DIMENSION                         0
-- CREATE DIRECTORY                         0
-- CREATE DISK GROUP                        0
-- CREATE ROLE                              0



-- CREATE ROLLBACK SEGMENT                  0
-- CREATE SEQUENCE                          2
-- CREATE MATERIALIZED VIEW                 0
-- CREATE MATERIALIZED VIEW LOG             0
-- CREATE SYNONYM                           0
-- CREATE TABLESPACE                        0
-- CREATE USER                              0
--
-- DROP TABLESPACE                          0
-- DROP DATABASE                            0
--
-- REDACTION POLICY                         0
--
-- ORDS DROP SCHEMA                         0
-- ORDS ENABLE SCHEMA                       0
-- ORDS ENABLE OBJECT                       0
--
-- ERRORS                                   0
-- WARNINGS                                 0
```

Feedback from Group53

PART A

A1: Functional Decomposition Diagram (FDD)

- The 'Printing' function includes 5 different reports. Reports is just an option for the reports, and they aren't necessarily tied to printing. Perhaps this heading could change? - Printing is done by the employee as a back-office service during a sale, rent, etc. It can be mandatory or option, but we have included it as a service.
- Reservation is not a service; we feel that reservation is a step of rental rather than a service itself. Return is the same as above. There are only two services 1) Rental 2) Sales. Perhaps there is no need for the customer service section? Both can be included in the diagram, but we feel they don't fall under services. – In our opinion we have considered Reservation, Rental, Return and Purchase as a front office service provided to the customer
- 1.3.1 There is no need for a return request as this is something that is determined by the rental return date. – Since couple of separate operations takes place during rental and return, we have not merged them together so considered them as separate services
- Could some of the boxes be merged? E.g., 2.3.2 Equipment update & inventory update – We have incorporated these changes as this seems redundant
- Lack of action to report the date and time on return date, could be added in – We have added these details to the rental and return section as the payment will be based on these actions of the customer

A2: Context-level Data Flow Diagram & Systems-level Data Flow Diagram

- Both level 0 and 1 are complicated, could functionality be simplified? Perhaps giving the overall flow of the system rather than specifying the details.
- Focus should be on the main entity e.g., customer, equipment, rental, contract.
- No need to go into minor details such as e.g., 'Late Fee Charges Calculation', 'Damage Inspection' and 'Equipment Status Update'.

Taking all the three points into consideration we have simplified level0 and level1 by making the flow clear and simple. We have removed couple of minor details and focused the process more on the major entities.

A3: Systems-level Diagram Level 2

- Good and easy to follow.
- For the Notification Process 1.1.5, unsure as to what this means? How would a customer receive a notification from the system? If it's to be included then it would need to be explained, perhaps as an assumption. If customers are informed something in person from employees, this action should not appear in the DFD – We have assumed notification as a process, and it will inform customer about the reservation request. We have added this point in the notes as an assumption

A4: UML Use Case Diagram

- Could actions be better described? An Use Case is a "set of activities that produce some output result" which means it should be a verb, not a noun e.g. Proceed for overdue or damaged equipment; Proceed payment for charges.
  "A use case describes a function that a system performs to achieve the user's goal. A use case must yield an observable result that is of value to the user of the system" (References: https://www.ibm.com/docs/en/rational-soft-arch/9.7.0?topic=diagrams-use-case) – Very well notified, we have changed all the nouns to verbs which represents an action

A5: UML Sequence Diagram

- Handled conditions well and correctly.
- Flow is very clear and easy to follow.
- Is there a need to include actions such as 'Retrieve Equipment from Storage'? – We assumed it was an interaction with the system and now realized it is a physical work. Good catch

PART B

B1: Entity-Relationship Model & B2: Relational Database Schema

There are lots of optional attributes that must be mandatory. - We have added all the mandatory and optional values which was missed out in the initial draft

- You don't need to have two tables for customer and customer history because each customer should have one unique ID. With two tables there would be two unique ID customer ID numbers which wouldn't make sense. – Yes, thank you for notifying. Now we have created a single customer table which contains all the customer details including the prior rentals and history data. So now if the employee queries the customer table he will get unique values and not duplicates
- We see that there is a relationship between Reservation and Customer table but in fact, there is no customer information in the Reservation table. Similarly, there is no reservation information in the Customer table. So how are they related to each other?- Yes, we have linked and added appropriate relationships and cardinality functions.
- The above point is the same issue with Employee table and Supplier table
- Overall, you should review the relationships between tables and see if they're actually connected or not, and how they are related to each other.
- There is "Equipment ID" in the Sale table, but in fact, you have no relationship between the Equipment Table and the Sale Table.

The ERD has been carefully modified with the above said points. The tables have also been normalized as per requirement now

B3: Create Table Statements

Not included as mentioned

Feedback from Group53

Section-BLOGICAL ERD

1) 'Customer Name' of Table 'Customer Table', 'Employee Name' of 'Employee Table' can be divided into 'Name' and 'Surname' attributes as
we dont want to store multiple data in a single attribute. It will violate 1st Normal Form. It may also help in improving searchability of data by 'Name'/'Surname' using where clauses. – We have made this change since it was not satisfying 1st normal form. So, we have split the name which will also be easy to query using where clause

2) A naming suggestion would be to remove the 'Table' from Table names - for example 'Employee table' should be 'Employee'. – We used it as an initial draft but in the final submission we have made the changes

3) A lot of 'Optional' data types have been used. They can be changed to mandatory so the database structure and function ability is not affected. – Added all the relationship and cardinality as per the requirement

References:

1. For the Entity Relationship Diagram, we referred the below links to gain more understanding,
   https://www.youtube.com/watch?v=jznwgqCCjto
   https://www.youtube.com/watch?v=QpdhBUYk7Kk&t=96s

2. For DFD context level and system level, we referred the below links to gain more understanding,
   https://www.youtube.com/watch?v=NuJ-Id-F5WA&list=PL-zXMQnQzGe9pG6yZ-JYx3UeKKdCEHzfI&index=11
   https://www.youtube.com/watch?v=MMAo5JGTz_k
   https://www.youtube.com/watch?v=p72mCYQBM8o

3. For Normalization, we referred the below links to gain more understanding,
   https://www.youtube.com/watch?v=J-drts33N8g

4. For UML Case diagram and sequence diagram, we referred the below links to gain more understanding,
   https://www.youtube.com/watch?v=zid-MVo7M-E&t=637s
   https://www.youtube.com/watch?v=uzWda-RmD3o
   https://www.youtube.com/watch?v=pCK6prSq8aw