Virtual money in vacation resort

A vacation resort has many places where a customer can buy things or services for small amounts of money (e.g. drinks, ice creams, towels, etc.). Having to bring cash or credit cards for paying is inconvenient for customers. A virtual money system replaces cash or credit cards with 'virtual money'.

Each customer, upon arrival in the resort, receives a bracelet with a unique ID written on the bracelet as a bar code. The customer can then buy (using cash or credit card) virtual money to be accumulated on the ID.

Then she can spend virtual money to buy goods or services and exchange it back for real money. In all cases what she has to do is just show the ID.

In the resort, each point of sale of services or goods is equipped with a reader that scans the ID on the bracelet and performs a sale transaction withdrawing virtual money from the bracelet / ID. In the resort some specific points (exchange points) are equipped for buying virtual money by providing cash or credit card.

1.a. Define the use case context diagram (including relevant interfaces)

1.b. List the requirements in tabular form

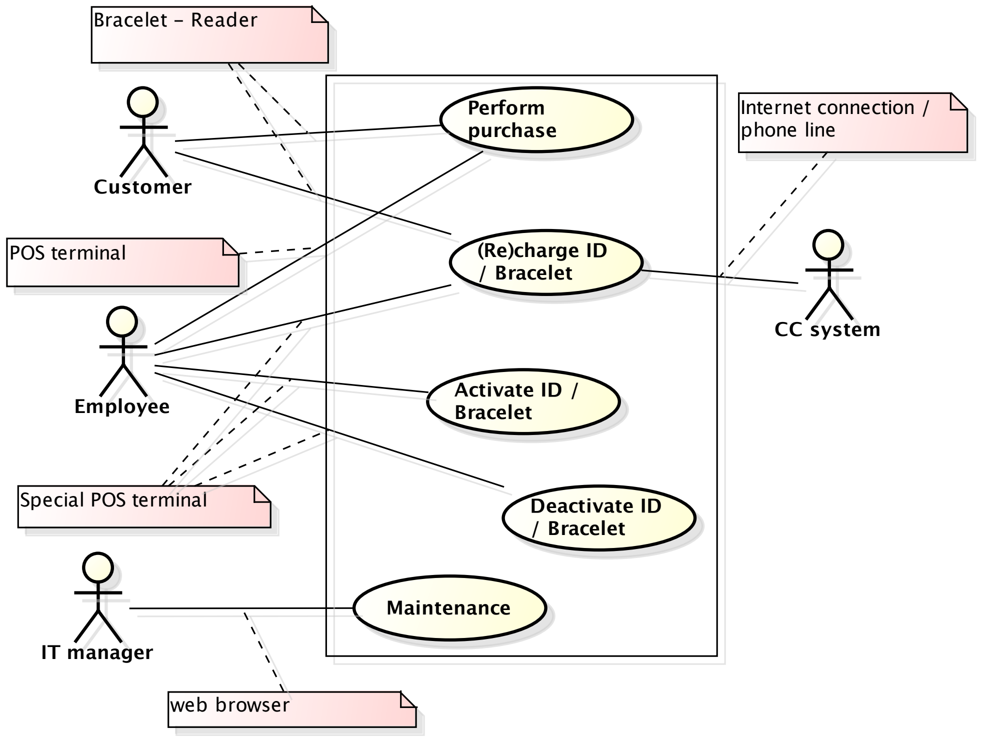
1.c. Define the conceptual information model (key concepts and entities and their relationships) (UML class diagram) for the system

1.d. Define one use case describing a user that spends virtual money at a point of sale

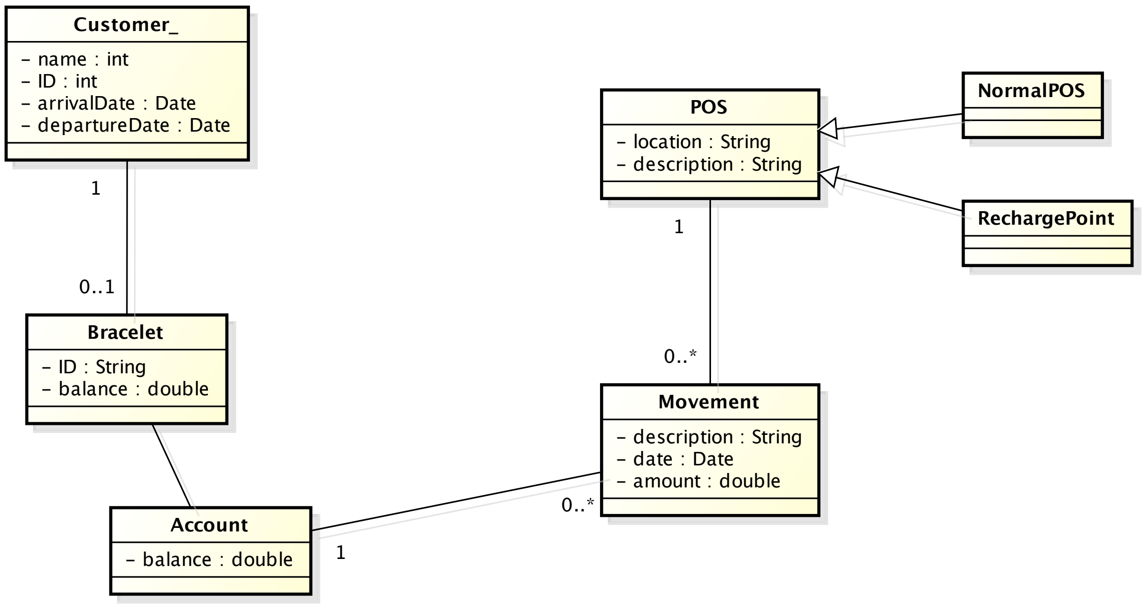
Precondition: user U has a virtual money account V. User has to pay amount A for a service and V.availableAmount >= A.

The system must ensure that: $V_{current} = V_{previous} - A$

1.a



1.c Conceptual information model



1.d: use case for goods/services payment

- Use case: Purchase goods / service OR get payment for goods / services
- Scope: Virual Money System
- Level: User gaol
- Intention in context: Receive payment for goods or service from the customer
- Primary actor: Employee
- Support actors: Customer
- Stakeholders' interests:
- Precondition: The ID/Bracelet has been activated, and has not been de-activated,
- (Minimum Guarantees)
- Success Guarantees: $V_{current} = V_{previous} - A$
- (Trigger)
- Main success scenario
  1. Employee enters amount (and description of purchase)
  2. The system requires payment
  3. The customer "shows" his/her bracelet
  4. The system validates the payment, subtract the amount, records the movement and shows a confirmation message
- Extensions

  1.a. the amount is not valid: the system notifies an error and return to step 1

  2,3,4.a: connection to the main server is lost: the system notifies the error
  and the use case terminates with failure

  3.b. error during bar code reading: system notifies error and failure

  3.c. timeout while waiting for bracelet information: system notifies
  the error and failure

  4.b. balance on account linked to bracelet < amount to be paied:
  the system notifies the problem and failure

Define black box tests for the following function, using equivalence classes and boundary conditions.

`double movingAverage(int x)`

receives an integer, returns the average of the last 4 numbers received. However x is used to compute the average only if > -10 and < 10, otherwise it is considered as a zero.When less than 4 numbers are available, the average is computed on the numbers available.

Ex. movingAverage(1) →1/1 = 1
movingAverage(2) →3/2= 1.5
movingAverage(3) →6/3 = 2
movingAverage(4) →10/4 = 2.5
movingAverage(5) →14/4 = 3.5
movingAverage(11) →12/4 = 3
movingAverage(-11) →9/4 = 2.25

For the following function define the control flow graph, and define test cases to obtain the highest possible node coverage, edge coverage, multiple condition coverage, loop coverage, path coverage.For the test cases, **write only the input value.**

```
1 int f (int num1, int num2, int num3) 2{

3  if ((num1 < num2) || (num1 < num3)) printf("no ok“);

4  while (num2 > 0)

5{

6  num1 = num1 - 1;

7  num2 = num2 - 1;

8}

9  if (num1 == num3) printf("ok“);

10  else printf("no ok“);

11 }
```