Software Engineering

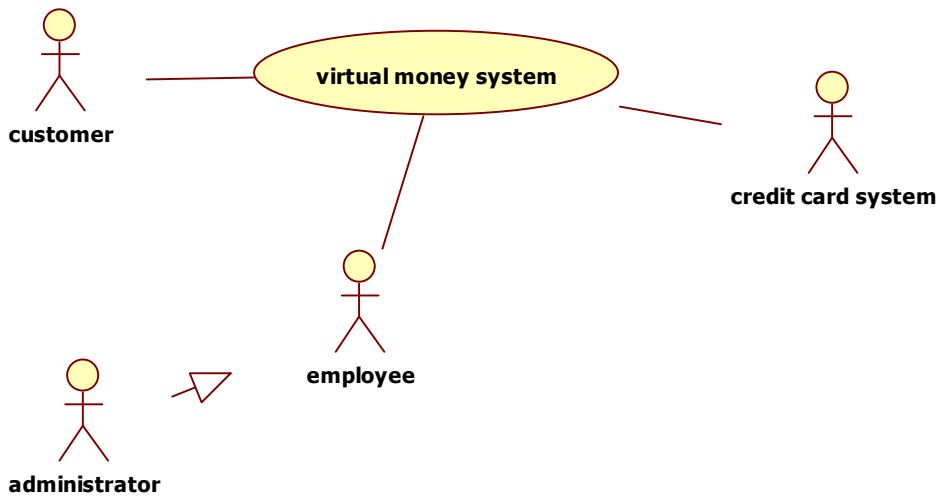
Books or notes are **not** allowed.
Write only on these sheets. **Concise** and **readable** answers please.

Surname, name, matricola _____

Virtual money in vacation resort
A vacation resort has many places where a customer can buy things or services for small amounts of money (ex. 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 attached to the ID. Then he can spend, buy, virtual money, exchange it back for real money. In all cases he has just to show the ID. In the resort, each point of sale of a service or good must be equipped with a register that reads 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) must be equipped for buying /selling virtual money versus cash or credit cards.

In the following you should analyze and model the virtual money service. This includes a server for managing IDs and virtual money attached to them, and clients for points of sales, and exchange points.

1 (15 points) – a. Define the context diagram (including relevant interfaces)



Remark: the bar code reader (to read bar code on bracelet) is inside the system (located in each POS or exchange point), so it should not be an actor.

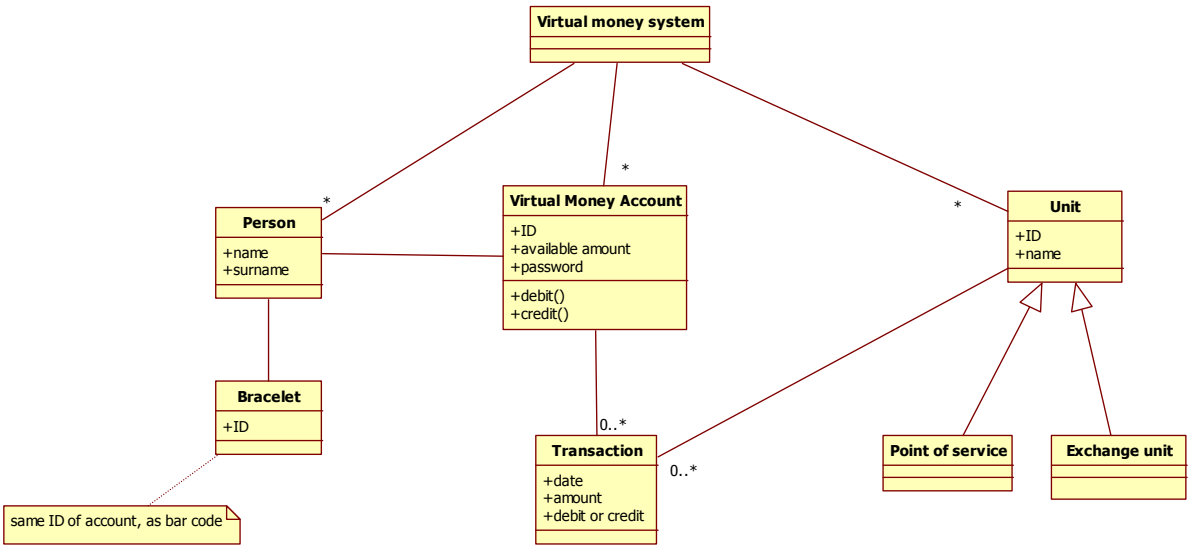
Actor	Physical interface	Logical
Customer	bar code reader (to read bar code on bracelet) display to show current credit	Value of bar code, value of credit
Administrator	PC	GUI to access all functions of system
Employee	PC	GUI to access most functions of the system
Credit card system	Internet connection	https + specific APIs to credit / debit a credit card

List the requirements in tabular form

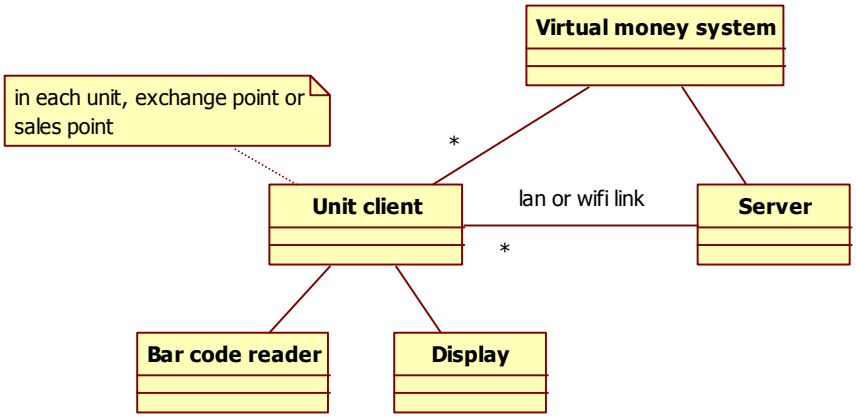
ID	Type (Functional Non Functional)	Description
1	F	Person management (add, modify, delete person record)
2	F	Virtual money account management (add virtual money account, attach to a person, add/subtract/read money, get-put money from-to credit card)
3	F	Transaction management (add transaction, attach to unit, attach to virtual money account)
4	F	Bracelet management (read bar code, retrieve attached virtual money amount)
5	F	Authorization authentication (add user, define privileges of user, login, logout)
	NF privacy	Credit card number should not be stored
	NF performance	All functions (and notably function 2) should complete in less than 0.2 sec

Remark, here only functions for the virtual money system are listed. Most likely the same system will support also all other services at units (such as sale of goods or services). However here we limit the list to what is specific to virtual money only.

Define the key concepts and entities and their relationships (UML class diagram) for the system



Define the system design (key hardware and software components of the system, UML class diagram) for the system. Describe the key design choices.



A typical client server system. The server manages all virtual money accounts and deals with credit card system. Clients read bar code from bracelet, get amount of money available, ask for adding / subtracting virtual money on associated virtual money account. Each client has a very simple display (no color, 4- 6 inches) to show basic info to the customer (bar code number, amount of money available).

Define one scenario 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.

Postcondition: $V.availableAmount_{current} = V.availableAmount_{previous} - A$

Step	Description	Req ID
1	User shows bracelet at bar code reader, reader reads bar code	4
2	System retrieves attached virtual money account V	4
3	System checks V.availableAmount >= A	2
4	System subtracts A to V	2
5	System records transaction on V	3

2 (8 points) -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

Range of x (minint to -10 to 10 to maxint)	Number of elements for average (<4, >= 4)	Valid / Invalid	Test case
]minit, -10]	<4	V	T(-20, -30) → -25 Tb (-20, -10) → -15
	>= 4	V	T(-20, -30, -50,-60,-70) → -25 Tb(-20, -30, -50,-60) → -40 Tb(-20, -30, -50) → -33.3 Tb(-20, -30, -50,-10) → -27.5 Tb(-20, -30, -10) → -20
]-10, 10[	<4	V	T(-8, 8) → 0
	>= 4	V	
[10, maxint[	<4	V	
	>= 4	V	

In test cases,

T(a,b,c) → result

stands for

movingAverage(a); movingAverage(b); movingAverage(c);→ result

3 (7 points) – 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          }
```

Coverage type	Number of test cases needed to obtain 100% coverage	Coverage obtained with test cases defined (%)	Test cases defined
Node	2	100%	T1(1, 2, -1) T2(1 ,2 , 4)
Edge	3, also 2	100%	T1, T2, T3(2,1,1)
Multiple condition (line 3)	4	100%	T1 TF T2 TT T3 FF T4 FT
Loop	3	100%	T1 enters many times T3 enters once T4(1, 0, 2) no enters
Path	Depends on num2, in general 100% unfeasible		

4 (1 points) – Describe briefly the ‘layered’ architectural style

5 (1 points) – What are the possible type of defects in a requirement document?

6 (1 points) – In the context of configuration management, what is a configuration item?

7 (1 points) – In the context of project management, give the definition of ‘milestone’