Software Engineering

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

Surname, name, matricola ____

Bike sharing
A bike sharing system allows users to rent and use a bicycle in a city for a certain amount of time, for a small fee (cfr TO BIKE in Turin). First a user has to register in the system, giving his identity, and a credit card. Then the user can pay for the service (in the case of Turin a yearly fee plus an amount of money that corresponds to a certain amount of minutes available). The user receives at his address a specific RFID card to access the system. If the user already has a city transport card or other cards (ex the university student ID card), then these cards are used.

In the city many parking spots for bicycles are available. The parking spots are made of many stands, each controlling a bicycle. If the user has the card she can access a stand (equipped with an RFID card reader) and requests a bicycle. If authorized (credit available, no banning for previous damages, etc) the stand unlocks the bicycle and the user can use the bicycle. Then the user can return the bicycle to any other stand.

Administrators of the system must be able to monitor the system (bikes available per each parking spot, used bicycles, missing bicycles, definition of usage fees etc). Similar functions are available to users, via a web site, that shows the parking spots in the city and the number of bicycles available. The web site allows also administering one's account (check the history of usage for a user, check credit available, etc)

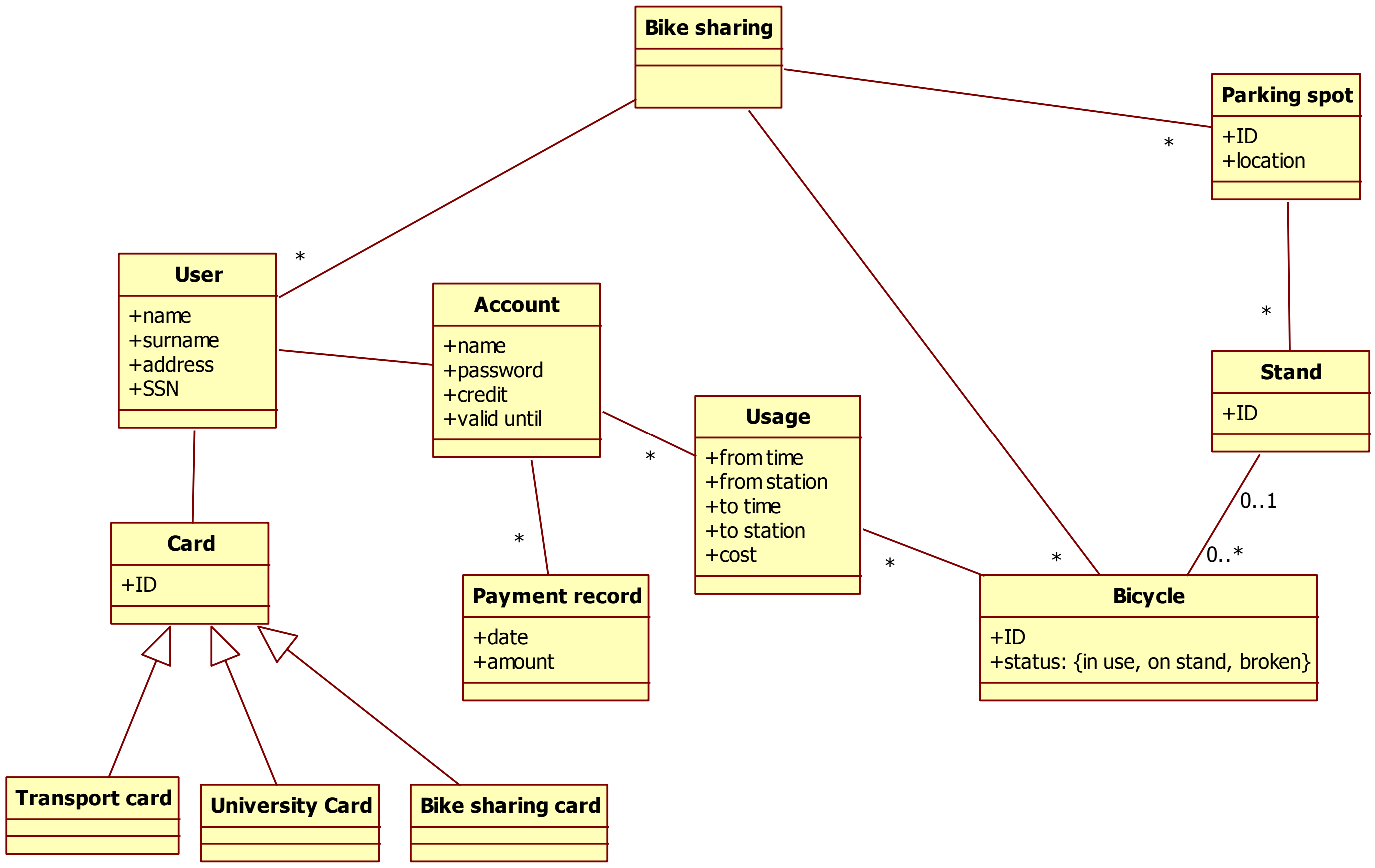
In the following consider the **system** for bike sharing (including bicycles, stands, web site).

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

Actor	Physical interface	Logical interface
User	PC (smartphone)	GUI
Admin	PC	GUI
Credit card system	Internet connection	Https, SSL, API with functions to check credit card, debit card etc

Bicycles and Cards are considered part of the system, so they are not here as actors. A logistic company (postal service or similar) could be another actor (for shipping cards). An email gateway could be another actor (to send emails to users).

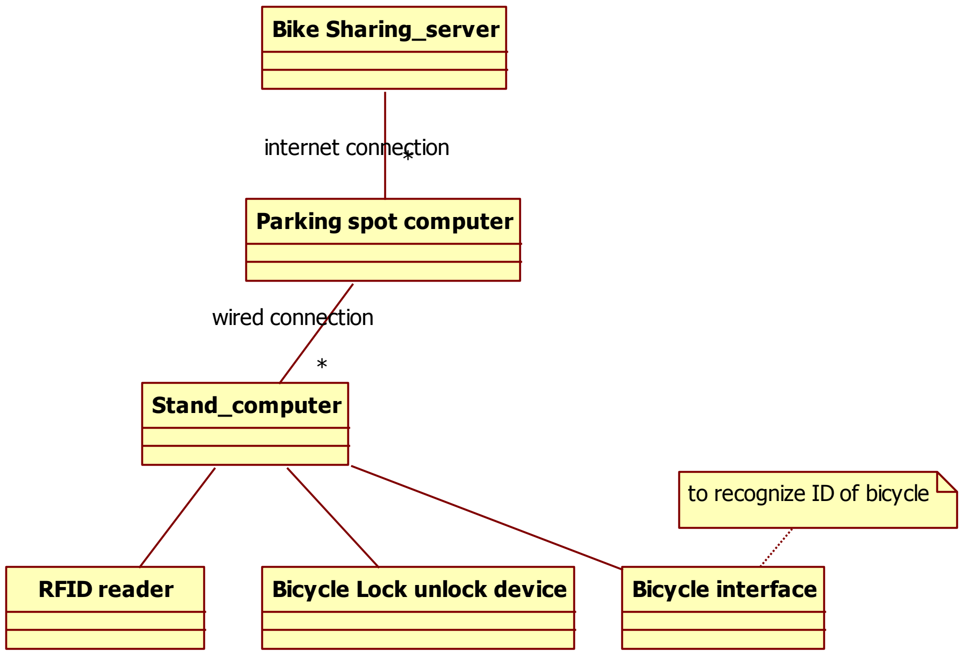
Define the glossary (key concepts and their relationships) (UML class diagram) for the system



List the requirements in tabular form (do not forget to list important NF requirements)

ID	Type (Functional Non Functional)	Description
1	F	User management CRUD user, CRUD account, CRUD card, attach account to user, attach card to account.
2	F	System configuration CRUD station, CRUD stand, CRUD bicycle, attach/detach stand to station, attach/detach station to system
3	F	Statistics Compute day/week/month usage per user, per bicycle, per station etc
4	F	Operation and accounting Authorize usage of bicycle, CRUD usage, compute cost per usage, attach usage to account, update fees per account, request payment for account, manage payment for account
5	F	Monitor and maintenance Monitor status of bicycle, stand; ask and monitor repairs. Show status of stations/bicycles on map
6	NF	Privacy. Use data should be visible only to user and administrator
7	NF	Usability. A user familiar with Internet tools (browser) should be able to use all functions on web site in at most 5 minutes, with no training.
8	NF	Performance. Authorization (or not) to collect a bicycle should be granted in less than ½ sec after rfid card read.

Define the system design model (UML class diagram)



The system design models physical entities (notably computers, devices and connections). A server is needed to manage all data and logic. Each parking spot has a computer to manage the stands, and another computer in the stand (for recognizing the bike, read card, lock unlock the bike). Another design could avoid a computer in each stand and have only one for the parking spot, directly connected to each stand. Anyway each stand has the same devices (lock unlock mechanism, card reader, bike reader).

Yet another design has only one interaction point per parking spot (screen, keyboard, card reader): the user interacts with it to operate on bikes. This design is cheaper but less convenient for users.

2 (7 points) -Define black box tests for the following function, using equivalence classes and boundary conditions.

```
double computeFee (int duration, int minRate, int minRate2)
```

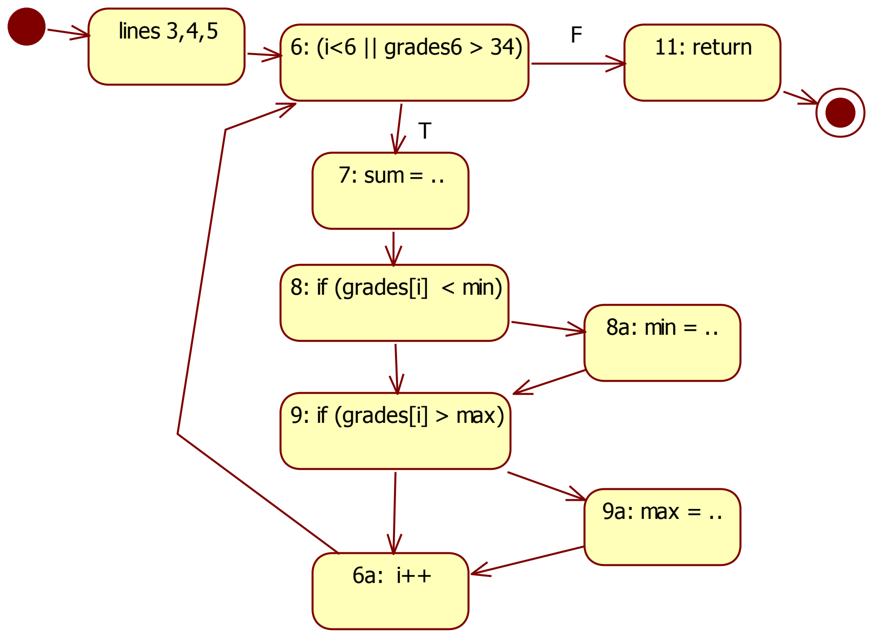
This function computes (in euros) the fee for a bicycle rental, using these parameters
- duration: minutes the bicycle has been used
- minRate: cost per minute, in cents of euro
- minRate2: cost per minute, in cents of euro

The fee is computed as follows: free the first 30 minutes. minRate per min for the first hour exceeding the first 30 min (30 to 90 minutes), minRate2 after 90 minutes

Ex. computeFee(35, 10, 20) → = (35-30) * 10
computeFee(65, 10, 20) → = (65-30) * 10
computeFee(95, 10, 20) → = (90-30) * 10 + (95-90) * 20

duration	minRate	minRate2	valid	Test cases
[minint, 0[	[minint, 0[	[minint, 0[	N	T(-10, -10, -10; err)
		[0, maxint]	N	T(-10, -10, 100; err)
	[0, maxint]	[minint, 0[	N	T(-10, 10, -10; err)
		[0, maxint]	N	T(-10, 10, 10; err)
[0, 30]	[minint, 0[	[minint, 0[	N	T(10, -10, -10; err)
		[0, maxint]	N	T(10, -10, 10; err)
	[0, maxint]	[minint, 0[	N	T(10, 10, -10; err)
		[0, maxint]	Y	T(10, 10, 10; 0.0) T_B(0, 10, 10; 0.0) T_B(1, 10, 10; 0.0) T_B(30, 10, 10; 0.0)
[31,90]	[minint, 0[	[minint, 0[	N	T(60, -10, -10; err)
		[0, maxint]	N	T(60, -10, 10; err)
	[0, maxint]	[minint, 0[	N	T(60, 10, -10; err)
		[0, maxint]	Y	T(60, 10, 10; (60-30)*10 = 3.0) T_B(31, 10, 10; 0.1) T_B(90, 10, 10; (90-30)*10 = 6.0)
[91, maxint]	[minint, 0[	[minint, 0[	N	T(60, -10, -10; err)
		[0, maxint]	N	T(60, -10, 10; err)
	[0, maxint]	[minint, 0[	N	T(60, 10, -10; err)
		[0, maxint]	Y	T(100, 10, 20; 60*10 + 10*20 = 8.0) T_B(91, 10, 20; 1*10 + 0*20 = 0.1) T_B(100, 0, 20; 60*0 + 10*20 = 2.0) T_B(100, 10, 0; 60*10 + 10*0 = 6.0)

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     double average (int grade1, int grade2, int grade3, int grade4, int
                                                  grade5, int grade6){
2           int grades[6]; grades[0]= grade1; grades[1]= grade2;
3           grades[2]= grade3; grades[3]= grade4;
4           grades[4]= grade5; grades[5]= grade6;
5           double sum = 0.0; double min =33; double max = 0;
6           for (int i=0; i<6 || grade6 > 34; i++){
7                sum = sum + grades[i];
8                if (grades[i] < min) min = grades[i];
9                if (grades[i] > max) max = grades[i];
10          }
11          return (sum -max - min)/4;
```

Coverage type	Number of test cases needed to obtain 100% coverage	Coverage obtained with test cases defined (%)	Test cases defined
Node	1	100%	T1 (iteration 0 and 1 cover nodes 8a, 9a)
Edge	1	100%	T1 (iterations 2 to 6 cover edges 8-9 and 9-6a)
Multiple condition line 6	4 in general, only 2 in this case	100%	T1 : T F T2: T T T1: F F (T1 at iteration 7) T2: F T (T2 at iteration 7)
Loop line 6	3	33% no way to force one iteration or zero iterations only	T1 6 iterations
Path	If grade6 <= 34 then the loop iterates 6 times, in this case the possible paths are <= $4^6 = 2^{12} = 4096$ If grade6 > 34 then the loops never ends. However if the language has array		

	bound control (ex Java) an exception will be raised at iteration 7 and the loop will stop. So in the best case the paths are <= 4096, not impossible to do but still challenging..

T1(20, 30, 21, 21,21,21)
T2(20, 30, 21, 21,21,35)

4 (2 points) – Describe shorty the Scrum technique

See slides

5 (1 point) – Provide an example of a configuration item

See slides

6 (1 point) — Describe the copy modify merge strategy for controlling changes. List advantages and disadvantages

See slides

7 (1 point) – Describe shortly the Pair Programming technique

See slides

8 (1 point) – Describe shortly the MVC Design pattern

See slides