Software Engineering

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

Surname, name, matricola ____

Web Wedding Wishlist
In weddings a group of people aggregate to select, buy and send gift(s) to the couple. This process was paper based (for instance a couple going to marry selects gifts in a specific shop (the wishlist), friends come to the shop, decide what gift to send, pay for it. The shop maintains the wishlist, receives payments, and finally sends the gifts). Nowadays this can be web based.

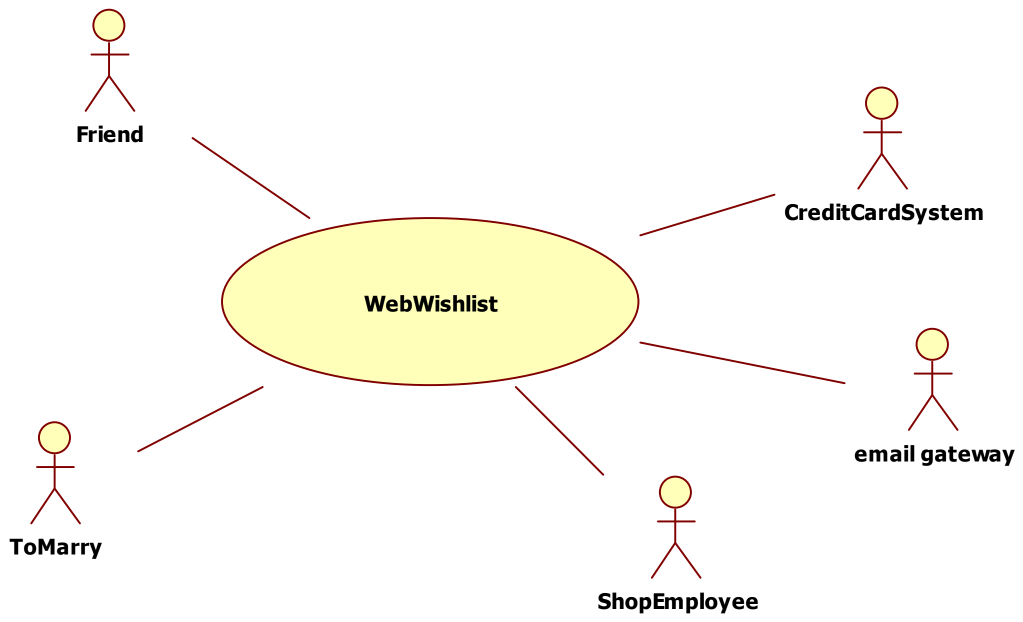
The main functions to be provided are:

- Management of the process (start and end dates, close / open group of friends, etc)
- Management of wishlist
- Management of group (couple, friends)
- Payments

In the following you should analyze and model the web wishlist system

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

Actor	Physical Interface	Logical Interface
Friend	Internet connection + client (PC, mobile)	GUI with functions to log in, browse catalogue, select and pay gift
He Couple, She Couple	Internet connection + client (PC, mobile)	GUI with functions to log in, browse catalogue, select gifts, define friends, invite friends, view status
ShopEmployee	Internet or LAN connection, PC	GUI with access to all functions
CreditCArdSystem	Secure Internet connection	Payment functions (debit account, credit account, refund account)
Email Gateway	Internet connection	Send / receive email



There could be two models of doing a gift. In one the gift is attached to a friend (a friend selects and fully pays a gift). This is the 'physical' model. In the other model a friend pays a contribution (either fixed – say 40 Euro -- , or not fixed), then all contributions are put together, the couple decides the ranking of objects, and therefore what to buy first. In both models it is possible that not all gifts in the wishlist are selected and paid for. The proposed solution uses the first model, but of course other models are acceptable. In both cases the information of what a friend does (what object buys, or how much contributes) should be stored.
In both cases there are sometimes complicated constraints to manage. For instance dishes are sold by the unity, but the couple should receive at least 6/12 dishes of the same type. For simplicity these cases are not considered.

Another choice is to attach the process to a physical shop, or to a virtual one. In the second case the entity offering the wishlist is a trader which buys products from other shops. In this case the catalogue is in fact a federation of catalogues from many retailers (aka resellers of hotel rooms or airline flights). The proposed solution uses the first, simpler option.

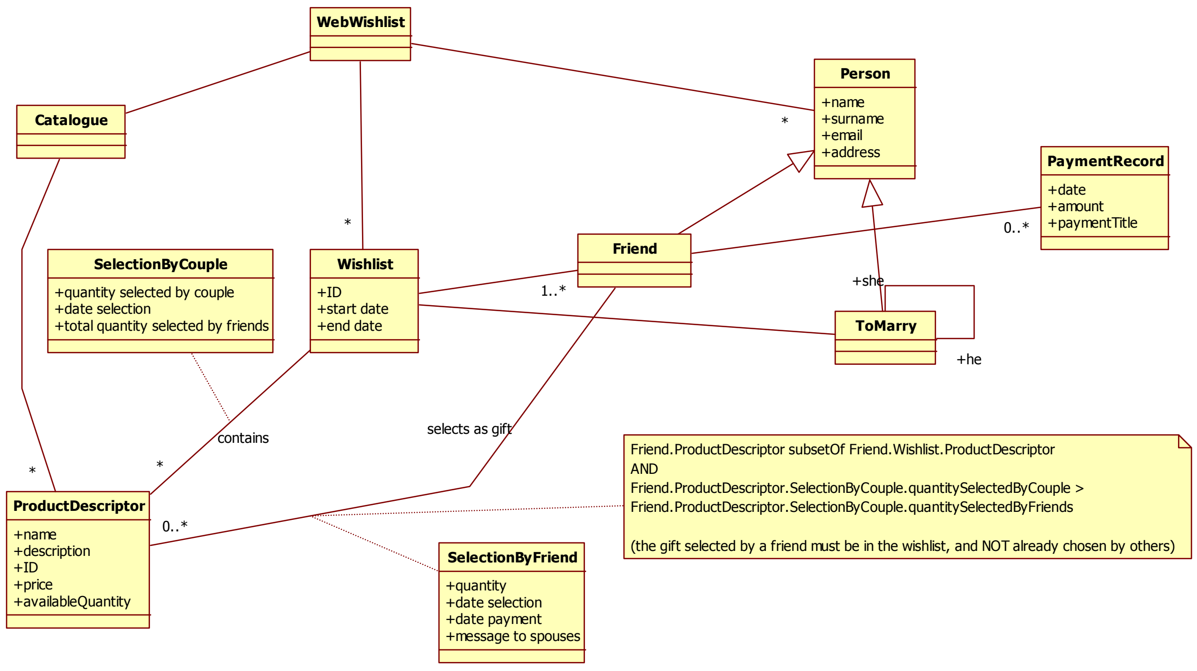
List the requirements in tabular form

ID	Type (Functional Non Functional)	Description
1	F	Manage wishlist
1.1	F	Create new wishlist
1.2	F	Define start date, end date (marriage date) of wishlist
1.3	F	Attach product to wishlist
1.4	F	Cancel product from wishlist

2	F	Manage group
2.1	F	Define new person (name, surname, email, account name, pwd)
2.2	F	Define role of person (to-marry, friend)
2.3	F	Attach person to wishlist
2.4	F	Invite person (send email) to wishlist
0	F	Authorize and authenticate
0.1	F	LogIn
0.2	F	LogOut
0.3	F	Manage roles and rights
3	F	Manage gift (product) catalogue
3.1	F	Browse catalogue
3.2	F	Define new product
3.3	F	Search product
3.4	F	Modify product
4	F	Manage contribution to wishlist
4.0	F	Browse wishlist (browse products in wishlist and their status)
4.1	F	Select product
4.2	F	Pay product (credit card, paypal, ..)
4.3	F	Change status of product (not more available for selection by another friend)
4.4	F	Show status of wishlist (list of products in wishlist, and their status)
	NF Privacy	Selection of friends should be private vs friends (other friends cannot know what gift a friend selects / pays) (couple can see the selections of all friends)
	NF Security	Payment functions should be secure. No one except the owner of a credit card should be able to use the credit card for payments
	NF Performance	All functions (except payment – this implies an external actor) should complete in < 0.5 sec

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

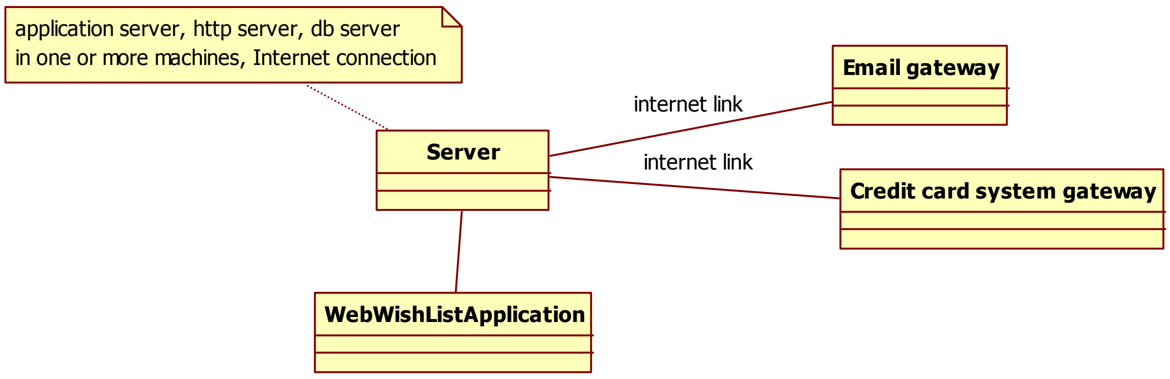
The key concepts are: a Catalogue of ProductDescriptors, the wishlist (containing a certain number of product descriptors), Persons. Persons can be friends, or the couple going to marry. Friends select (from the catalogue) one or more products.



WebWishlist
Person
+name
+surname
+email
+address
Catalogue
PaymentRecord
+date
+amount
+paymentTitle
SelectionByCouple
+quantity selected by couple
+date selection
+total quantity selected by friends
Wishlist
+ID
+start date
+end date
Friend
ToMarry
+she
+he
*
*
0..*
1..*
*
*
0..*
contains
selects as gift
ProductDescriptor
+name
+description
+ID
+price
+availableQuantity
SelectionByFriend
+quantity
+date selection
+date payment
+message to spouses
Friend.ProductDescriptor subsetOf Friend.Wishlist.ProductDescriptor
AND
Friend.ProductDescriptor.SelectionByCouple.quantitySelectedByCouple >
Friend.ProductDescriptor.SelectionByCouple.quantitySelectedByFriends
(the gift selected by a friend must be in the wishlist, and NOT already chosen by others)

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

The system design is straightforward, since this is standard web application. We need a server (say a PC) with a web server (Apache http server or MS IIS) and an application server (could be JBoss or similar). Within the application server the application runs (class WebWishListApplication). The software design would contain most classes found in the UML for key concepts (Catalogue, WishList, Person etc).



application server, http server, db server
in one or more machines, Internet connection
Email gateway
internet link
Server
internet link
Credit card system gateway
WebWishListApplication

Define one scenario describing a friend selecting and paying a gift
Precondition: Friend F has been invited to wishlist W, Gift G is in wishlist W and still available

Postcondition: Gift G is not anymore available in W. Payment record PR exists for G.

Step	Description	Req ID
1	Friend F logins	0.1
2	F browses wishlist W	4.0
3	F selects product G	4.1
4	F pays for G	4.2
5	System changes status of G	4.3
6	F logs out	0.2

2 (8 points) -Define black box tests for the following function, using equivalence classes and boundary conditions.

The function receives the value declared at cadaster for a property, and computes its fiscal yield. The formula applied is

cadasterValue *0,1 if the property was built before or in year 1960
cadasterValue *0,2 if the property was built after 1960

double yield(int year, int cadasterValue);

ex. yield(1955, 1000) → 100
yield(1965, 1000) → 200

year	cadasterValue	Valid / invalid	Test case
[Minint, 0[[0, 1960]]1960, 2013]]2013, maxint]	[minint , 0[[0, maxint]		
[Minint, 0[	[minint , 0[	I	T(-10, -20, error)
	[0, maxint]	I	T(-10, 20, error)
[0, 1960]	[minint , 0[	I	T(1956, -20, error)

	[0, maxint]	V	T(1956, 200, 20)
			Boundary:
			T(1960, 200, 20)
			T(1961, 200, 40)
			T(1959, 200, 20)
			T(1956, 0, 0)
			T(1956, -1, error)
			T(1956, 1, 0,1)
			T(1956, maxint, 0,1*maxint)
			T(1956, maxint-1, 0,1* (maxint-1))
			T(1956, maxint+1, error)

]1960, 2013]	[minint , 0[	I	T(2000, -30, error)
	[0, maxint]	V	T(2000, 300, 60)
]2013, maxint]	[minint , 0[	I	T(2015, -200, error)
	[0, maxint]	I (also possible to consider this valid, otherwise the function should be re-written every year)	T(2015, 400, error)

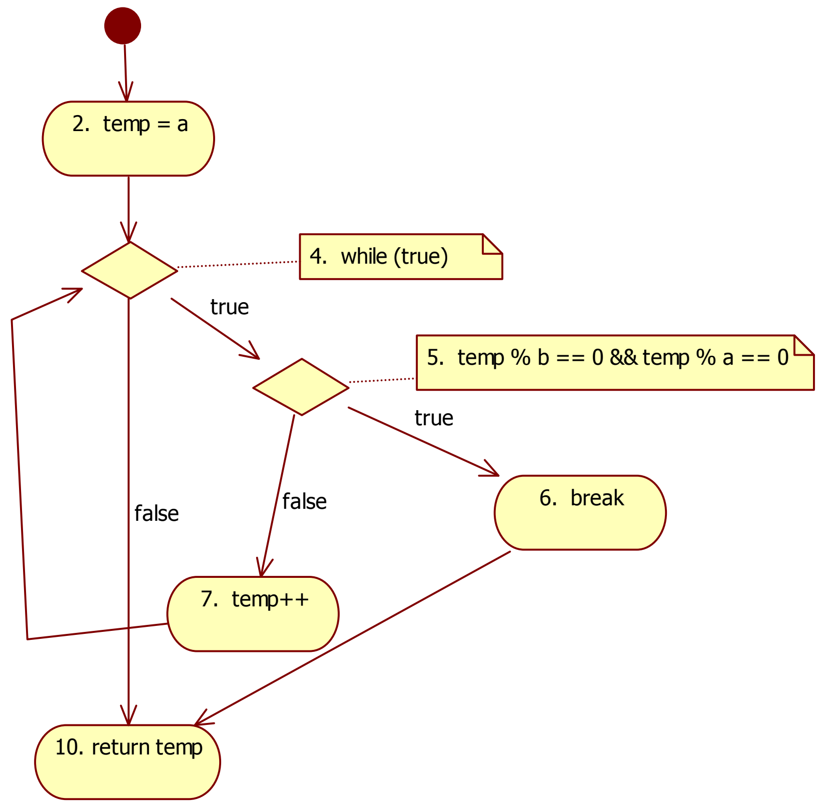
Boundary test cases written only in one class, similarly for other classes

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 lcm(int a,int b){
2    int temp = a;
3
4    while(1){
5         if(temp % b == 0 && temp % a == 0)
6             break;
7         temp++;
8    }
9
10   return temp;
11}
```

Coverage type	Number of test cases needed to obtain 100% coverage	Coverage obtained with test cases defined (%)	Test cases defined
Node	1 enough with loops	100%	T(2,3)
Edge	1 enough with loops	6/7 (if edge 4-10 in the control flow) or 100% (if no edge 4-10)	T(2,3)
Multiple condition (line 5)	4 in theory, 1 enough with loops	100%	T(2,3) performs F T T F F T F F T T
Loop	0 loops impossible to achieve	2/3	T(1,1) one loop T(2,3) many loops
Path	Number of loops in the while 2 456 2 457 456 2 457 457 456 In general 2 $\{457\}^n$ 456 with n from 0 to a number that depends on the max LCM storable in temp before overflow So in the order of millions	100% not practically achievable	

For the control flow it is possible also not to draw the edge 4-10, that is not feasible.



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

Several applications communicate only through exchange of (standardized) data files. No direct interaction between applications, no call to APIs. Example: Eclipse.

(The question was about software architectures, not about repositories in context of configuration management)

5 (1 points) – Provide an example of two conflicting non-functional properties of a software architecture

Very high precision of computation (ex square root precision 10^{-10}) conflicts with Performance. The higher the precision, the slower the response time.

6 (1 points) – Describe briefly the ‘Scrum’ agile process

Sprints iterations of fixed duration (max 4 weeks) producing a working application in increments.
Ranking of requirements by end user / customer (requirements backlog).
Stand up meeting (15’) every day for coordination.

7 (1 points) – A project is estimated to require 40 person months. Give an estimated range of the required calendar time, and explain how you compute it.

A reasonable team for such a project could be 2 to 5 people. Assuming the team has the same number of people from start to end this gives a calendar range of 20 to 8 calendar months (computed as person * months / persons).
A team of 40 people is unreasonable (too much coordination and communication overhead) so 1 month duration would be unfeasible. Similarly for 20, 30 people.