Software Engineering

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

Surname, name, matricola _____

Car entertainment system

Most cars today have an onboard computer, dedicated to entertainment and navigation. Consider a typical computer with a 12 inches touch screen, and an on off button. Consider also as interfaces: a Bluetooth connection, and a usb port. Further interfaces are 4 buttons placed on the steering wheel (volume up, volume down, answer phone call, close phone call), a microphone, and several speakers embedded in the car.

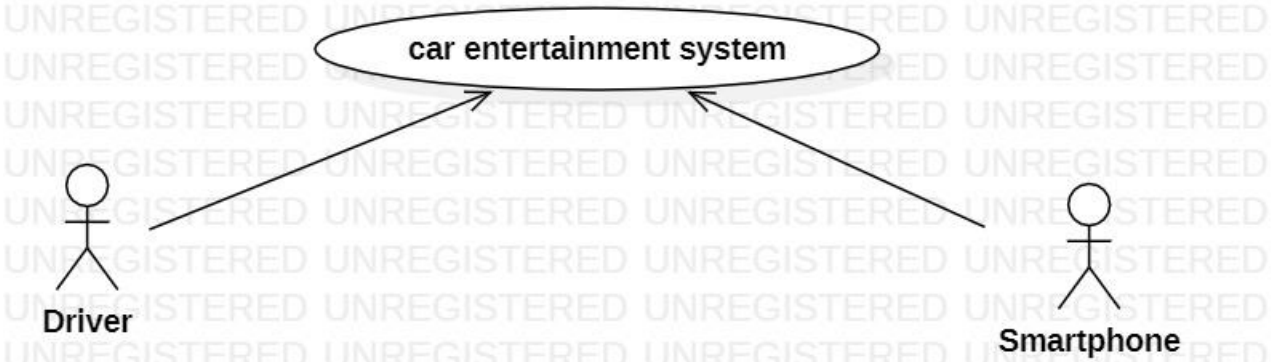
The onboard computer can run various applications, preloaded on it, for road navigation, for music playing (radio listening or music read from usb port).
Further, the computer can connect with an external smartphone, either via Bluetooth or via USB cable. For safety reasons, drivers must not use a smartphone while driving, however they can interact with the phone via the onboard computer, either through voice commands, or through the available buttons, or through the touch screen. Always for safety reasons, interactions on the touchscreen should be as limited as possible.

In the following you should analyze onboard computer functions specific to interfacing with the smartphone (so do not consider the applications for navigation and music) in order to manage phone calls and texting (send and receive messages on Whatsapp). As for safety regulations, received messages must not be shown on the screen but should be translated to voice, and similarly a message must not be written on the screen, but must be dictated by voice. As for phone calls, placing a call (either by entering a number to be dialed, or by selecting an entry in the phone book) must be done by voice too.

Assume that, on its side, the smartphone has standard interfaces to control phone calls and texting.

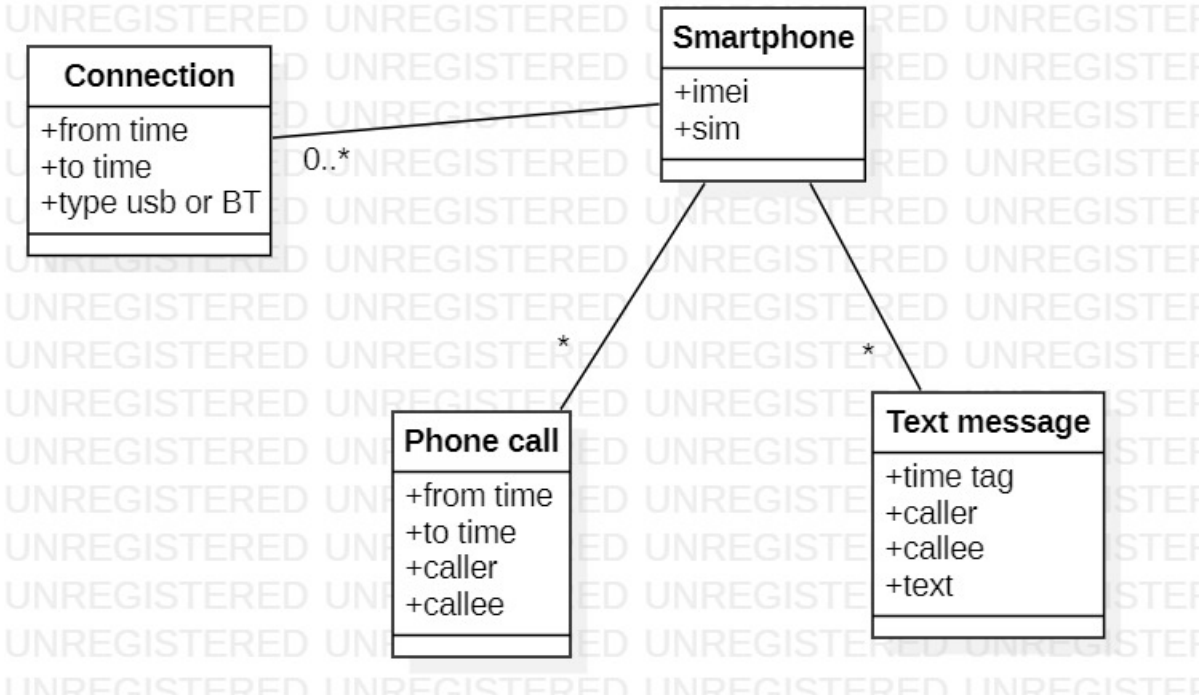
1 – a. Define the **context diagram** (including relevant interfaces)

Actor	Physical interface	Logical interface
driver	Buttons, voice, ears	Voice commands, button commands, text translated to speech, speech translated to commands
Smartphone	BT or usb cable	API on phone to control phone



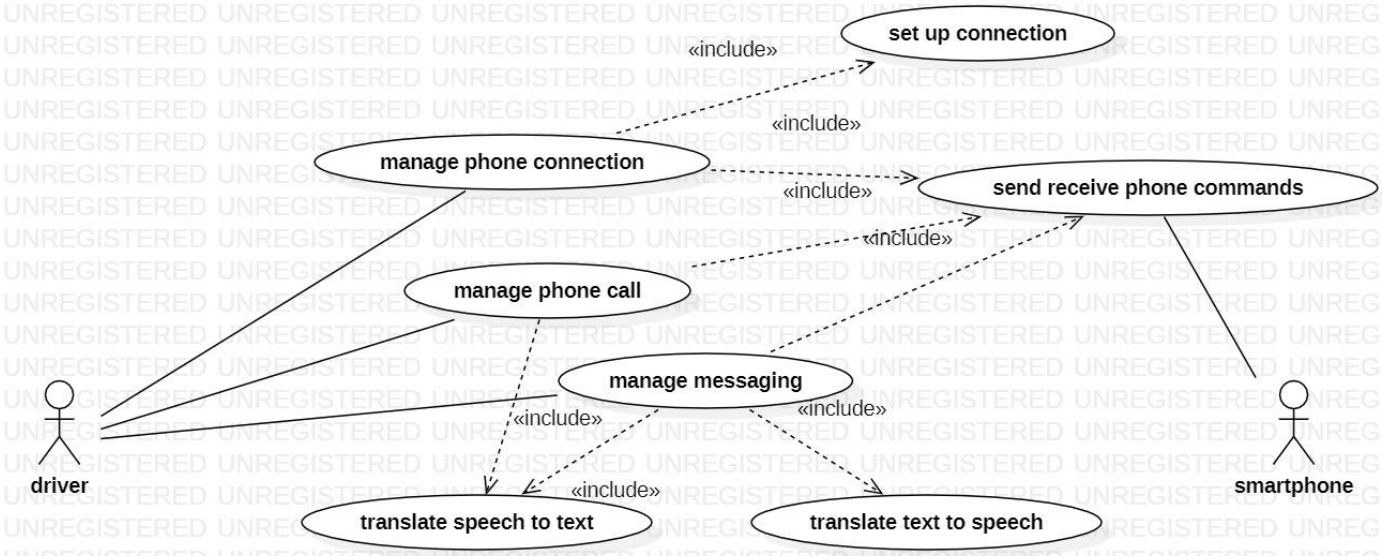
Typical errors on context diagram. Onboard computer, buttons, considered as actors (they are NOT actors). The Smartphone is an external actor, since the car entertainment system acts as a proxy for it, the driver interacts with the smartphone through the car entertainment system.

1-b Define the **glossary** (key concepts and their relationships) (UML class diagram) for the application

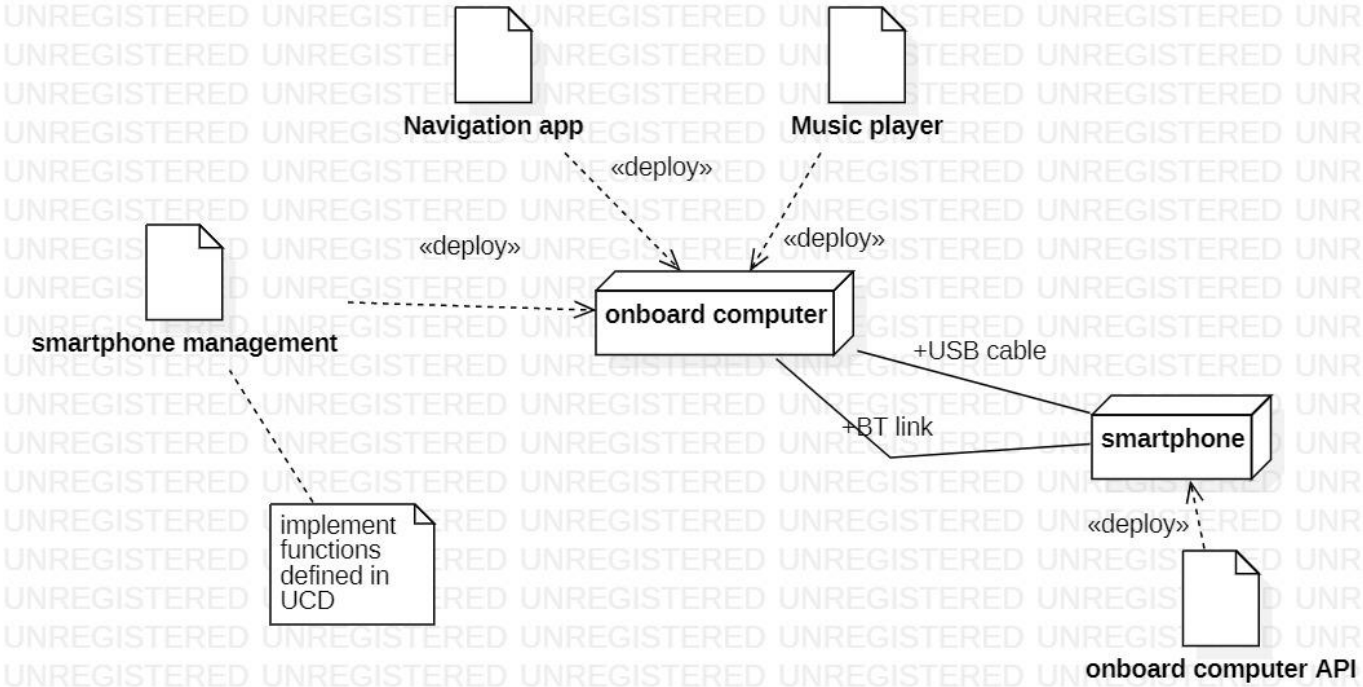


Typical errors: button, screen, microphone, speakers, usb, BT. These are not concepts, but components of the system. To be shown in system design model (see later)

1-c Draw the Use Case Diagram for the application.



1-d Draw the deployment diagram



Typical errors: button, screen, speaker, microphone are NOT nodes (they have no computing capability)

System design (was not requested, however the components, especially hw, of the system should be modeled here, not in the glossary)

Onboard Computer
- Cpu, memory
- Touchscreen
- USB port
- BT channel
- Speakers
- Microphone
- Input buttons (volume up, volume down, answer phone call, close phone call)

2 Define black box tests for the following function, using equivalence classes and boundary conditions.
The function receives a number and returns true if the number is prime

function `isPrime(number: number): boolean`

```
criteria: sign of number, prime or not
classes: sign of number > 0, sign < 0
          is prime, ! is prime
```

sign	Is prime	Test cases	
negative	-	T1(-10, error) Tb1(-lowestinteger, error) Tb2(-1, error) Tb2(0, error)	

positive	Yes	T2(2, true) Tb3(1, false) Tb4(+highestInteger, ?) Tb5(+highestInteger +1, ?) Tb6(+highestInteger -1, ?)	Tb4, Tb5, Tb6 result true or false should be computed knowing the actual value of highestInteger Tb3 can be considered boundary both in sense of sign (0-1,0, 0+1) and in sense of prime (1 is a particular non prime number, since it has only one factor in division, definition of prime requires min 2 max 2 factors in division)
positive	Not	T3(4, false)	

Typical error: confounding prime numbers with even/odd numbers.

3 For the function above and the implementation below, 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.  class PrimeChecker {
2.
3.    public static isPrime(numero: number): boolean {
4.
5.       if (numero < 2) {
6.           return false;
7.       }
8.
9.       for (let i = 2; i <= Math.sqrt(numero); i++) {
10.         if (numero % i === 0) {
11.             return false;
12.         }
13.      }
14.
15.      return true;
16.  }
17.
18. }
```

Coverage type	Number of test cases needed to	Coverage obtained	Test cases defined

	obtain 100% coverage	with test cases defined (%)	
statement	3	100	T1 (line 5,6) T2 (line 5, 9, 15) T3 (line 5, 9, 10, 11)
decision	3	100	Line 5: T1 true, T2 false Line 10: T2 true, T7 false (and true) T7(77, false)
Loop	3	100	Loop zero: T2 Loop once: T3 Loop many: T7
Path	The negative number path 1-5-6 can be tested by just one negative number The positive paths can be split in two patterns [9-10] n times, 11 (non prime) And [9-10] n times, 15 (prime) Using brute force these paths are tried using all positive numbers from 1 to highest integer, so if number is represented as integer on 32 bits, it means 2^(31) test cases, roughly 10^10, making path coverage unfeasible in practice. However non primes having the same factor (ex 4 and 6 and 8, or 9, 12, 15) have the same path, so if we test 4 we could avoid testing all following numbers with 2 as a factor, if we test 9 we could avoid all following numbers having 3 as a factor and so on. In all cases 100% path coverage is hardly feasible.		

Typical error. A test like T1(-10) does NOT tries the loop without entering (zero times loop) because the flow exits at line 6. T2 flows till line 9, tries the loop, but does not enter it.

4 A good requirement document should be

A. Complete and consistent
B. Ambiguous and complete
C. Redundant and consistent
D. Reliable and complete
E. Measurable and complete

5 Correctness of a software application is defined as
A. Complete satisfaction of all customers
B. Complete satisfaction of the majority of users
C. Actual output corresponds to the expected output for most inputs
D. Actual output corresponds to the expected output for all inputs
E. Successful certification by the specific domain authority

6 Considering non functional requirements of a software application
A. Reliability is more important than usability and maintainability
B. Usability is more important than reliability and maintainability
C. Maintainability is more important than usability and reliability
D. Maintainability reliability and usability are equally important
E. Ranking of non functional properties must be defined by the stakeholders

7 Function A calls function B, that calls function C. You want to apply bottom up integration. How do you proceed?

A. Test C, then test B+C, then test A+B+C
B. Test C, then test A+B+C
C. Test A+B+C
D. Test C, test B+mock of C, test A + mock of B, Test A+B+C
E. Test A+mock of B, test A+B+ mock of C, test A+B+C