Software Engineering – 02GSP

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

Surname, name, matricola _____

Car speed control (Tutor system)

Traditional speed monitoring systems measure the instantaneous speed of a car on a street, following the car for few meters with radar based appliances.

The idea of the Tutor system instead is to measure the average speed of a car on a long distance (in the order of 1-10 kilometers). This is made by measuring the time when a car passes at point A, the time when the same car passes at point B, and doing simple math to compute the average speed. Given to this design the speed can be computed only in highways, where cars cannot change direction over the measured distance.

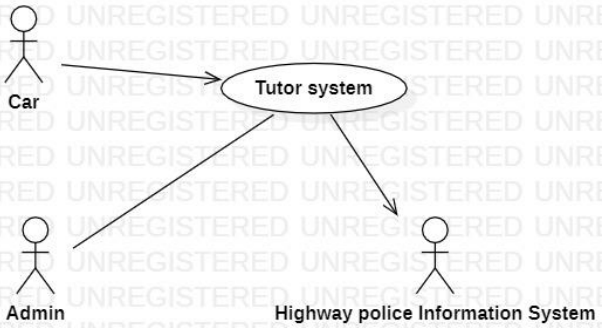
In point A and B on the highway a gate is built with one or more cameras that take pictures of cars passing, triggered by a motion sensor. The motion sensor is connected to the camera, the camera is connected to the Internet. Via internet the pictures, along with the time when they were shot, are sent to a central computer. Here some post processing is made automatically: image analysis (to recognize the tag of the car), image storage, computation of the average speed of a car, using the identified tag, For cars that have a speed above a defined limit, a report is produced and sent (as a json file with a defined structure) to the information system of the highway police. The information system of the highway police is outside the tutor system.

The Tutor system is designed to handle many hundreds of pairs of gates, throughout the highway network.

The Tutor system also provides an API for external systems. Through this API it is possible to configure gates, set parameters, download and delete pictures, and in general do maintenance operations.

In the following you should analyze and model the Tutor system.

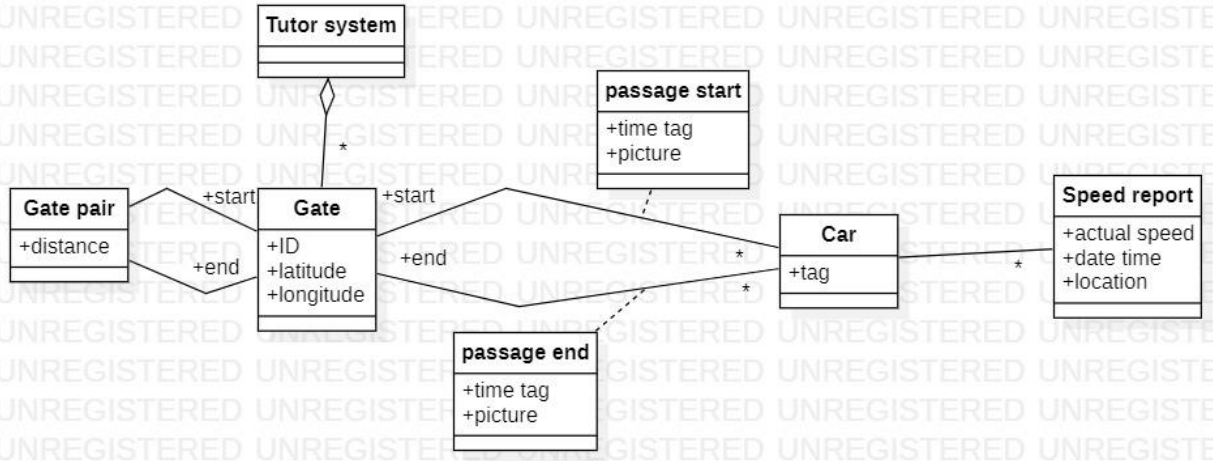
1 Define the context diagram



2 Define interfaces

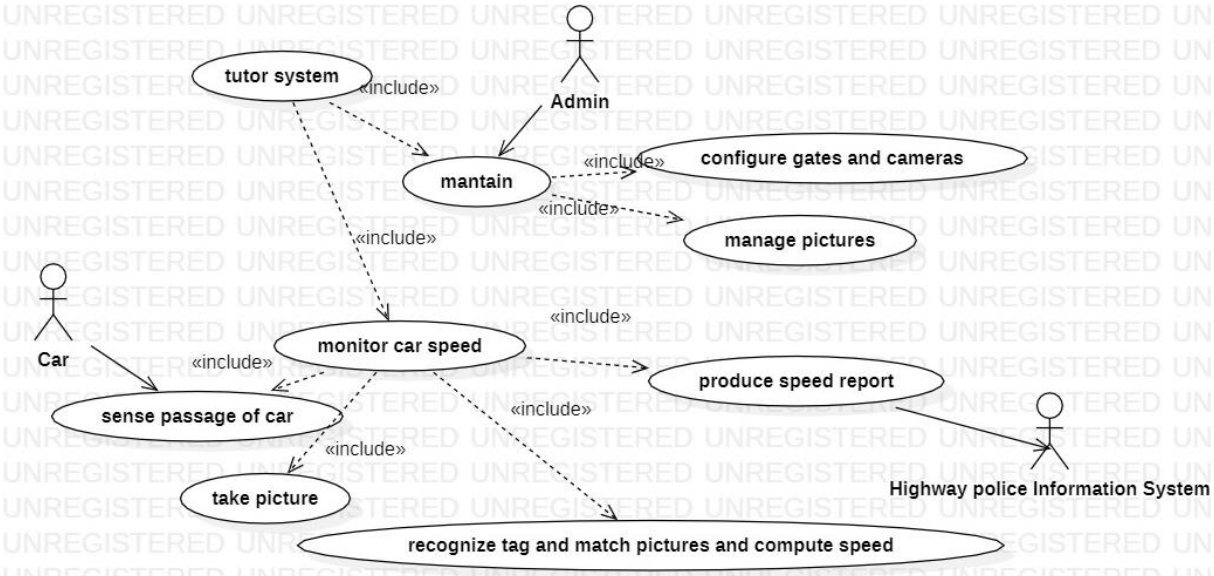
Actor	Logical interface	Physical interface
Car	Car-is-passing event	Motion sensor
Admin	API (configure system, set parameters, manage pictures..)	PC, screen / keyboard
Highway police information system	API (send report) + car speed report (Json file)	Internet link

3 Define the glossary (key concepts and relationships), using a UML class diagram



Gate pairs must be represented, to store their distance, and be able to match pictures correctly. Many pictures of cars are taken at gate ‘start’, same at gate ‘end’, then they must be matched using the tag of the car (collected via image processing). After the match the time tags can be compared and the speed computed as distance / (time tag end – time tag start). The system does not need to retrieve other car information (owner, driver), this can be done by the Highway police.

4 Draw the use case diagram. For each use case give self explicable long names



5 Define the Non Functional requirements

Privacy: non authorized users must not be able to access pictures and reports

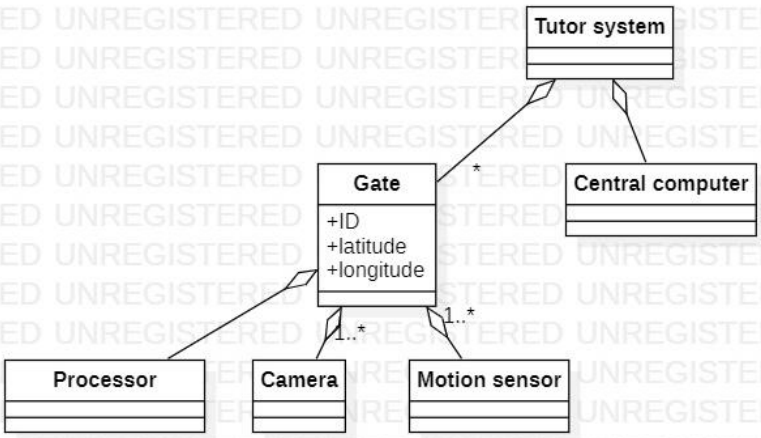
Domain: the system should be able to visually recognize cars' tags from all over Europe

Reliability: the system should be able to capture pictures, analyze them, and produce reports even in case of heavy rain / snow

Correctness (precision): speed should be computed with max error 1%

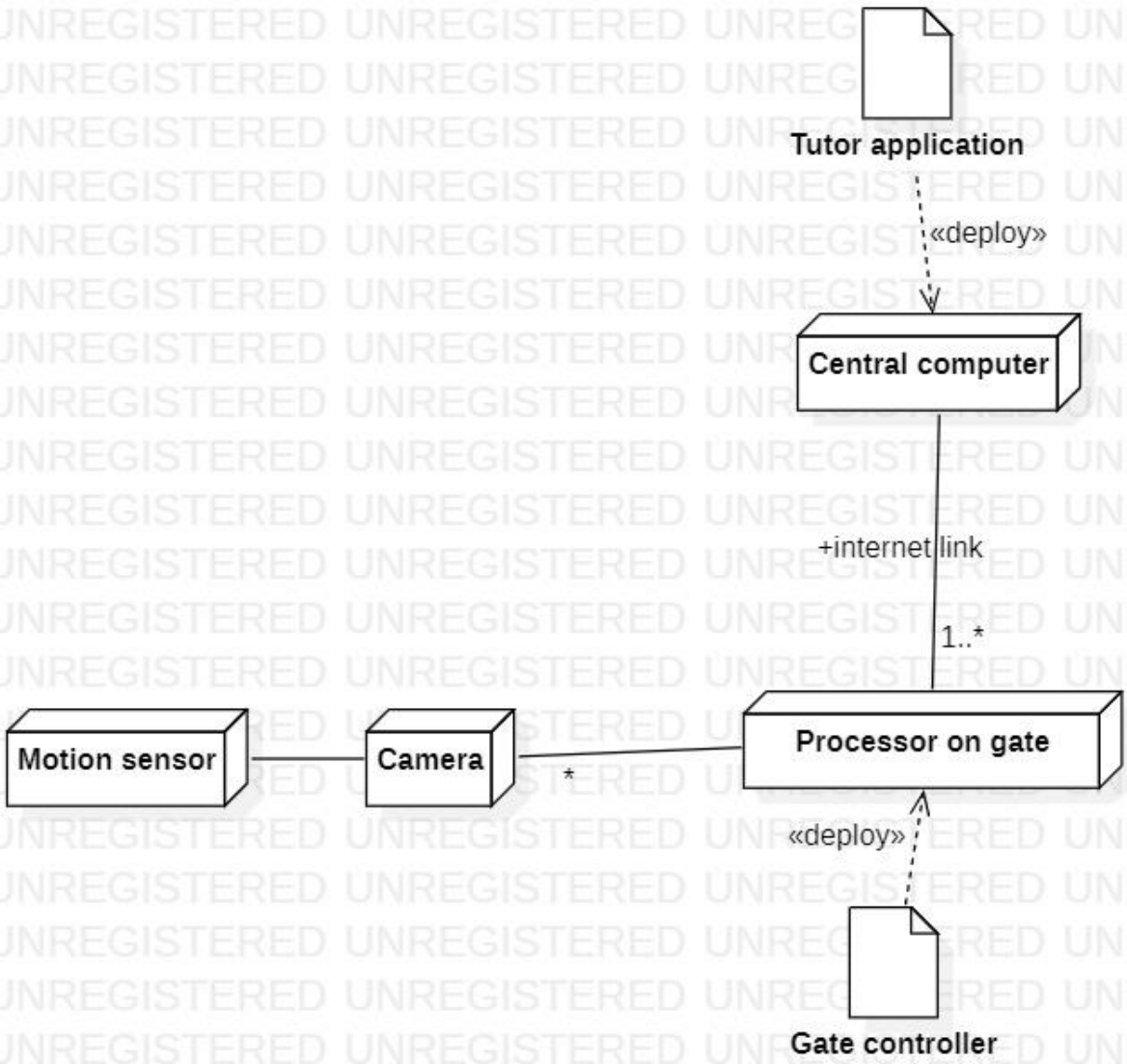
Performance: the system should be able to produce and send a report at most 10 seconds after a picture has been analyzed and a car is travelling at an average speed higher than the maximum

6 Define the system design (using UML class diagram)



Although not considered in the textual description, some kind of processor is needed in each gate, both to monitor the state of the gate (errors and state of sensors and cameras), to oversee the communication with the central computer, to store and possibly preprocess pictures.

7 Draw the deployment diagram



8 For the following function 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; 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	Feasibility (Y/N)	Coverage obtained (%)	Test cases
node	Y	100	T1
edge	Y	100	T1
multiple condition (line 8)	-	There is no multiple condition at line 8, t1 achieves 100%	T1
loop	Not controllable	33	T1
path	Y	There are 4^6 possible paths, so in principle 100% coverage is feasible	

T1 (2, 30, 25, 26, 18, 23)