GI2 2017-2018

WebProject: Highway Management

By:
Wejdene Haouari
Ilyes Ltifi
Insaf Khorchani
Salma Rais
Aya ourag
Chaker Iyadh

I. System Description:

There are two sections to this system: The agent section, and the admin section.

The agents connect to the system to do one of two things:

1. Create a ticket for a vehicle entering the highway.
2. Verify a ticket and collect a fee from a vehicle exiting the highway.

An agent can connect to the system with his CIN and a password provided by an administrator. An agent may only work from only one station. **The system insures that agents can only log in from their designated stations.**

The administrator is responsible for the management of the various components of the system. There is only **one admin account** on the server, it is by **default user=>“root” password=>”0000”.** For security reasons, the admin login cannot be modified. It has to be manually changed in the database.

NOTES:

- in order to properly test the system, you must open at least three sessions: one for admin, the rest for agents.
- **The agent sessions must have different cookies!** You can achieve this by connecting each agent from a different browsers. *See section* ***V. Operating the system*** *for more details.*
- If the server database doesn’t exist, it will be automatically generated upon any login attempt.
- **DO NOT CONNECT VIA PHPSTORM:** using phpstorm to open localhost may cause problems.

II. System Components:

- **Station:**
  A station has a name a distance and a price for each category. There are only three categories of vehicles.
  A station is associated with the highway it operates on. We will call the name of the highway Line.
  A station is identified by its name and its Line.

- **Agent:**
  An agent has a CIN (which is also his login) a name and a last name.
  Each agent **is associated with a station.**

- **Ticket:**
  A ticket has an id, date of create, category, and station of departure.
  There are only three categories.

- **Machine:**
  Each computer or machine has a unique id and is associated with a station. **These are the only computers that are allowed to connect to the website.**

- **Machine Requests:**
  Machine Requests represents a log of all computers that tried to connect to the system but **are not recognized in the database.**
  one machine request has computer id, the CIN of the agent that tried to connect from it and the exact time of the event.

III. Admin Section:

The admin can manage all the components above, except Ticket.

- **Adding Stations:**
  In order to **add a station**, the admin **must first select the line** on which to add the station, then input all the necessary information.

  A line is simply two stations, in order to add a line the admin must input two stations in the **add line** form.

- **Updating prices:**
  The admin can update the category prices for each station individually with **update price** or update the prices for all stations within a Line **by percentages** using **update General price**.

- **Deleting Stations:**
  the admin can delete a station individually. He can also delete an entire Line and all the stations within it. **Deleting a station will delete all the computers and agents associated with it.**

- **Agents:**
  An agent cannot create an account on the website. Only the admin can add or delete agents.

- **Add Machine:**
  And admin can review all the machine requests. He can select a machine and choose to accept or refuse it.When the admin accepts a machine request, that machine becomes associated with the station of the agent that made the request.
  The admin can also choose to block all unrecognized machines.

IV. Agent Section:

Station Entry:

The agent operates this button if he is working on vehicle coming into the highway. For each vehicle he must specify its category then print the ticket.

Get Ticket:

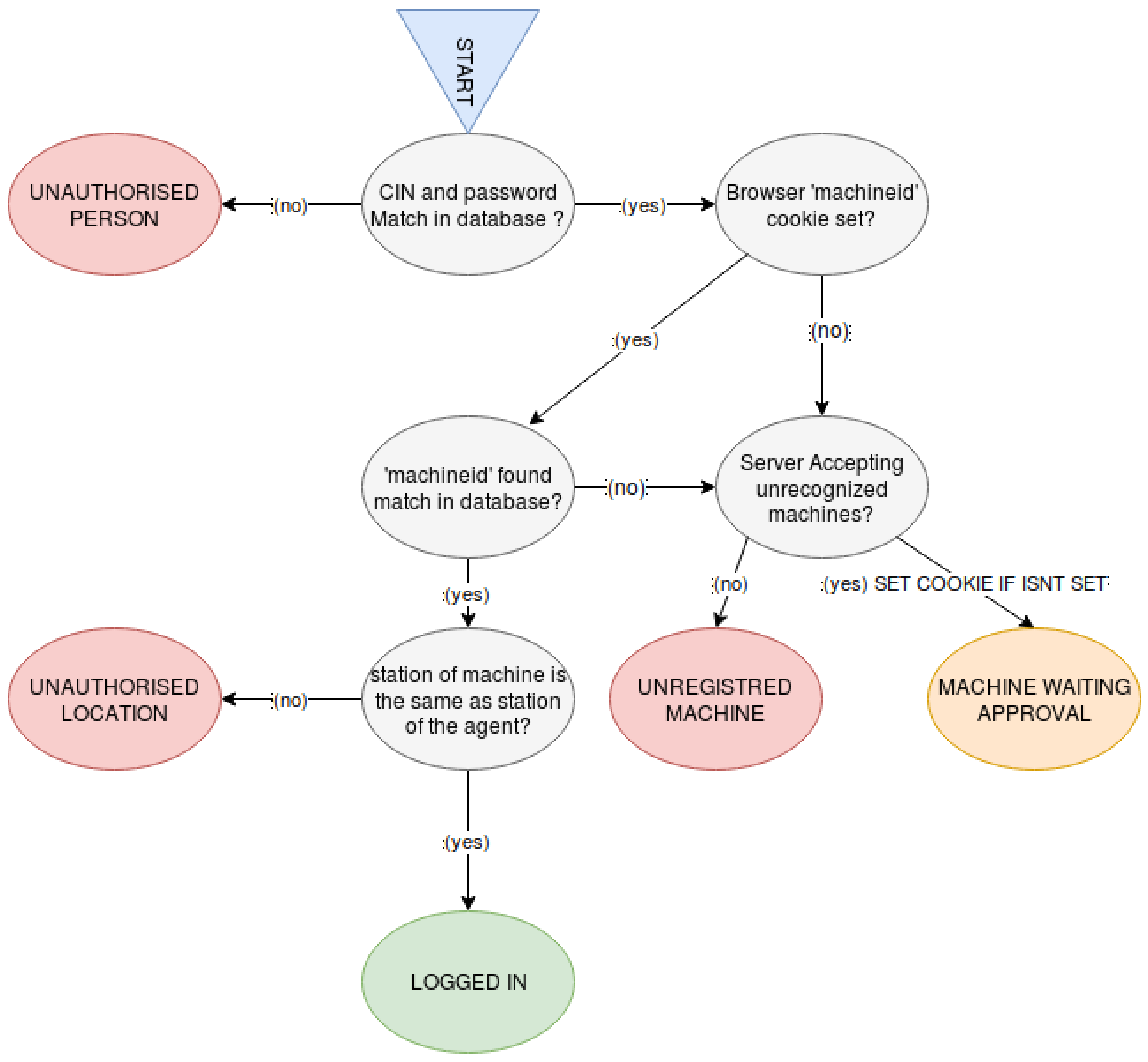
This agent operates on vehicles getting out of the highway.
For each vehicle he inputs the ticket number, and the system calculates the total distance traveled by that vehicle and the fee it must pay based on its category.

Agent Login:

Because this is website, if a registered agent can connect from anywhere he wants, he can create fake tickets and compromise the system.
So how do we require that the agent can only log in from his work station?

Our solution is to create a database of machines that can connect to the system. Having each machine associated with a station and each agent is associated with a station, we can check if the agent's station is the same as the machine's station.
We use cookies to store the machine id on the machine.

Here is a flowchart explaining the login verification process.


START
CIN and password Match in database ?
(no)
UNAUTHORISED PERSON
(yes)
Browser 'machineid' cookie set?
(yes)
(no)
'machineid' found match in database?
(no)
Server Accepting unrecognized machines?
(no)
(yes) SET COOKIE IF ISNT SET
UNREGISTRED MACHINE
MACHINE WAITING APPROVAL
(yes)
station of machine is the same as station of the agent?
(no)
UNAUTHORISED LOCATION
(yes)
LOGGED IN

V. Operating the system:

In order for the system to function there must be two stations (that define a line), two agents and two machines.

1. Add the line.
2. Add the two agents.
3. Have the two agents try to connect to your system.
4. Accept the machine requests.
5. The agents can now connect and operate the system.

VI. System Architecture:

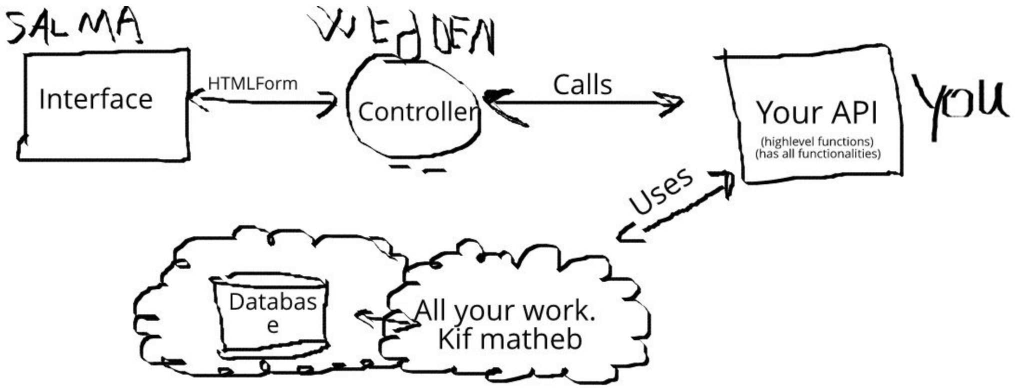
The system has four layers, each layer only communicates with the one directly above or below it:

- **<u>User Interface Layer:</u>** handles the website appearance and enforces session control (an agent cannot enter an admin page, he will automatically be redirected, and vise versa) . Associated folders : interface, includes/interface, res.

- **<u>Controller Layer:</u>** handles redirection, verifies user input and handles error displays. This layer insures that the correct arguments are passed to the layer below it. Associated folder: includes/controller

- **<u>Component API layer:</u>** every system component has its own API which is a PHP file containing only the functions that the rest of the system needs. The functions require minimal arguments and return error messages if the call is impossible. Associated folder: includes/api

- **<u>src Layer:</u>** In order to encapsulate complexity in the API layer, some APIs use functions/services defined in the src. These are generally DataBase Queries, or functions not needed by the rest of the system. **src/DBConnection.php** provides the DB connection to all the src files, and **builds the database if it doesn't exist.** Here is a sketch from the early days of the system conception:


SALMA
WEJDEN
YOU
Interface
HTMLForm
Controller
Calls
Your API
(highlevel functions)
(has all functionalities)
Uses
Databas
e
All your work.
Kif matheb