Project specifications

Car registration service

12.03.2023 година

Made by:

Росана Тодосиева, 181126

Никола Торбовски, 151200

Генц Ибраими, 181540

1. Introduction

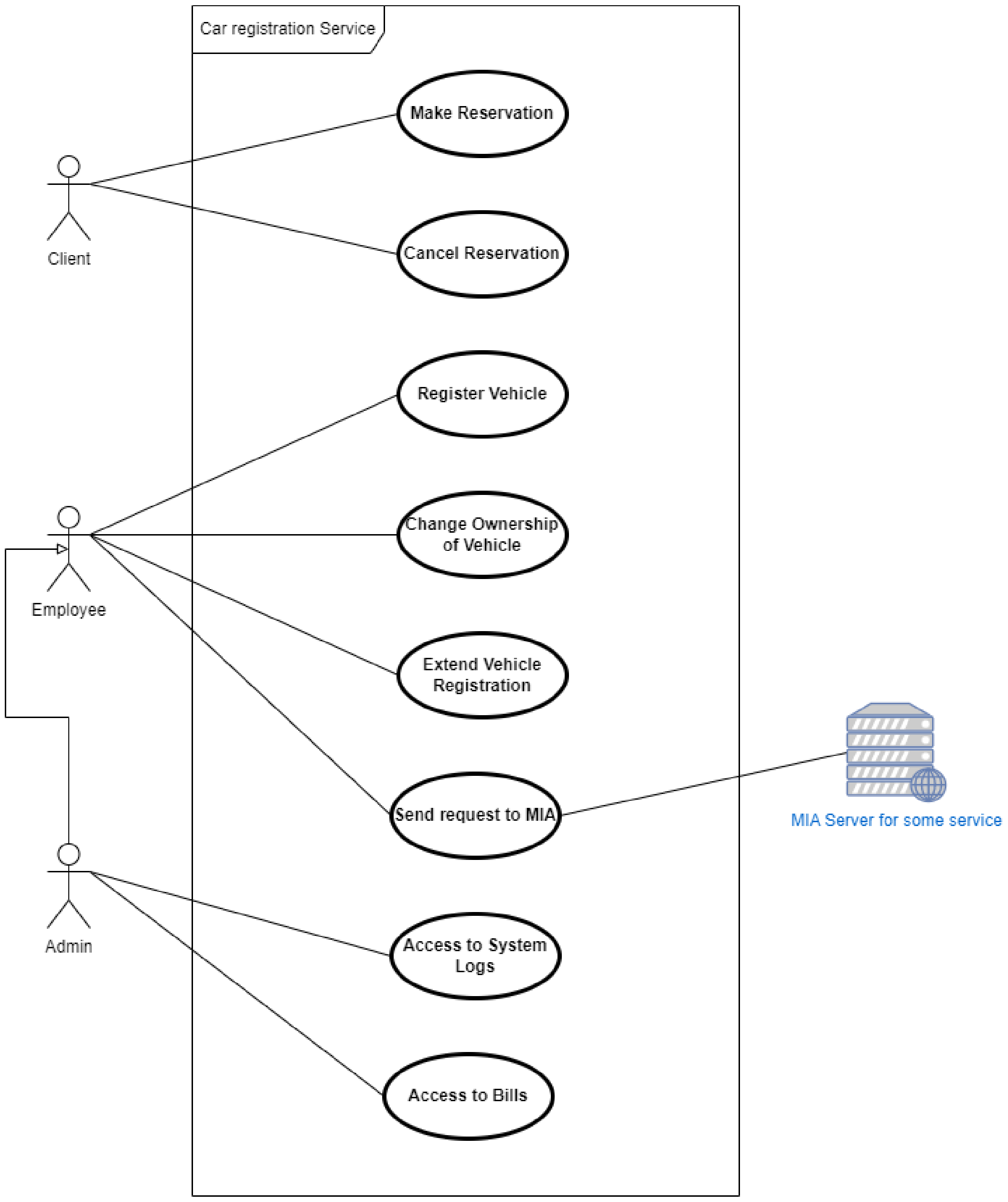
1.1 Purpose: An application in which employees can register vehicles, change ownership of cars or continue an already existing registration. This application will also have a client part where the client can choose a specific time in which he will go to register his vehicle and will avoid the crowd. If he's a new client, he can also enter his information in the system so when he goes there, the employee won't need to enter all his information manually and can take them from the database.

1.2 Intended audience: This application is intended for a specific company, its employees and their clients.

2. General Description

2.1 Product perspective: Whenever we go to register our vehicles, we end up waiting in long lines and lose a lot of time. So we decided to create an application that will not only work for the employees and make their jobs easier, it will also make our (the clients) lives easier.

Description of the system:



2.1.1 Use-Case: Make Reservation

Description:	This use case describes how the Client uses the system to schedule appointment for Vehicle registration

Actors:	Client
Pre-conditions:	The Client must be logged with his personal account
Main Flow of Events:	1. The Client opens view Make Reservation 2. The system prefills his personal information in the form 3. The Client selects the type of service 4. The Client selects date and time for physical appointment 5. The Client sends a request 6. The Server creates new ticket 7. The Client gets status for the created ticket
Alternative Flows:	· **Cannot create new ticket** If in step 6 the Server cannot create new ticket, then 1. The system sends failure message to the Client 2. The use case resumes at 5 · **No Response from Server** If in step 7 the Client does not get response from the Server, then 1. The system sends failure message to the Client 2. The use case resumes at step 5 · **Quit** If at point prior to step 5 in the main flow the Client selects Cancel, then

	1. The use case ends
Post-conditions:	The Client gets status and ID for the newly created ticket

2.1.2 Use-Case: Cancel Reservation

Description:	This use case describes how the Client uses the system when he wants to cancel an already made reservation
Actors:	Client
Pre-conditions:	The Client must be logged with his personal account
Main Flow of Events:	1. The Client opens view Cancel reservation 2. The Client selects a reservation 3. The Client clicks the cancel selection feature 4. The System removes the reservation 5. The canceled reservation is removed from the list

Alternative Flows:	· **No reservations** If prior to step 1 no reservations were made, then 1. Show “You do not have any reservations” 2. The use case ends · **No Connection to Server** If at step 4 connection to Server is lost, then 1. Display failure message 2. The use case resumes at step 3
Post-conditions:	The Client has successfully canceled the reservation

2.1.3 Use-Case: Register Vehicle

Description:	This use case describes how the Employee uses the system to Register Vehicle using the provided documentation, by the Client, for the same Vehicle
Actors:	Employee
Pre-Conditions:	The Client must have a valid reservation
Main flow of Events:	1. The Employee creates new Entry for the clients’ ticket 2. The Employee inputs the documents handed by the Client

	3. The Employee saves the Entry in the system
Post-Conditions:	New Entry for the registration of the Vehicle has been saved in the system

2.1.4 Use-Case: Change Ownership of Vehicle

Description:	This use case describes how the Employee uses the system to Change Ownership of Vehicle from one Entity to another, using the documents provided by them
Actors:	Employee
Pre-Conditions:	One of the Entities must have a valid reservation
Main flow of Events:	1. The Employee creates new Entry for the clients' ticket 2. The Employee inputs the documents handed by both Entities 3. The Employee saves the Entry in the system
Post-Conditions:	The Vehicle owner has been updated in the system

2.1.5 Use-Case: Extend Vehicle Registration

Description:	This use case describes how the Employee uses the system to Extend Vehicle Registration for the Clients vehicle

Actors:	Employee
Pre-Conditions:	The Client must have a valid reservation
Main flow of Events:	1. The Employee creates new Entry for the clients' ticket 2. The Employee inputs the documents handed by the Client 3. The Employee saves the Entry in the system
Post-Condition:	The vehicle registration expiration date has been updated

2.1.6 Use-Case: Access to System Logs

Description:	This use case describes how the Admin uses the system to view System logs
Actors:	Admin
Pre-Conditions:	The Admin must be logged in
Main flow of Events:	1. The Admin opens the Admin Dashboard 2. The Admin opens the System Logs tab

2.1.7 Use-Case: Access to Bills

Description:	This use case describes how the Admin uses the system to Access information about all Bills for all Entries
Actors:	Admin
Pre-Conditions:	The Admin must be logged in
Main flow of Events:	1. The Admin opens the Admin Dashboard 2. The Admin opens the Bills tab

2.2 User class and characteristics:

Users of the system should be able to schedule a visit and fill their own information in the system, which will be saved in the database.

The system will support three types of user privileges, Customer (normal user), Employee and Admin. Customers will have access to customer functions, the employees will have access to the employees functions and the admin will have access to both user and employees actions, plus check when the employees worked, check all documents and check all payments.

The customer should be able to do the following functions:

1. Make a reservation:
    a. For changing ownership
    b. For continuing registration
    c. For removing a vehicle from its ownership
2. Cancel an existing reservation

The employee should be able to do the following functions:

1. Add vehicles and owners
2. Register vehicles
3. Change car ownership
4. Continue existing registration
5. Give requests to the MVR for removing the car, registrations and for driving in other countries

The admin should be able to do the following functions:

1. Everything that user and employee can do
2. Check all documents
3. Check system log times
4. Check bills

2.3 Operating environment:

1. The data of the system will be stored in a PostgreSQL database.
2. The system will operate on every operating system.
3. The system will require an internet connection.
4. The system must operate within common web browsers.

2.4 Constraints:

1. The system shall require a maximum of 2 minutes for loading and logging into the system.
2. The system should have a free flowing interface to keep the usability of it simple and easy. Complex actions shall not be included to keep the system complexity minimized.
3. The system shall handle failures successfully.

2.5 Assumptions and dependencies:

1. The user and employee will have decent internet connectivity.
2. The user and employee will be familiar with using an internet browser.
3. The system will be very rarely overloaded.
4. In one year we assume the system will be used by at least 1000 users.
5. The system can be used by multiple users who will be connected at the same time.
6. Users' personal data will be protected.
7. Certain functionalities of the system will require the privileges of an admin or other user role.
8. The time between the data being stored in the database and the client reacting in real time will be almost unnoticeable.
9. Testing must be carried out to ensure that the application is fully functional.

3. System Requirements

3.1 Functional requirements

3.1.1 System requirements

Level of priority	Description
First level	Necessary requirements
Second level	Desirable requirements
Third level	Additional requirements

R.1. The application shall inform the user how to perform the services offered on an automated basis. (First level)

R.2. The application shall inform and guide the employed in how to perform the services offered on an automated basis. (First level)

R.3. The application should allow admin login. (First level)

R.4. The application should allow login to the user. (First level)

R.5. The application should allow the admin to log out. (First level)

R.6. The application should allow a user to log out. (First level)

R.7. The application should be available from all current browsers: Google Chrome, Firefox, Opera, Safari.(First level)

R.8. The application should allow the system to announce/ show only the time slots that are available. (First level)

R.9. The application shall contain updated information regarding the last time the vehicle was registered. (Second level)

R.10. The application shall contain updated information regarding the last time the ownership was changed. (Second level)

R.11. The application shall not allow the user or the employee to detect application failures, especially ones that won’t be specific. (Second level)

4. External Interface Requirements

There are many types of interfaces as such supported by the PetFriend software system namely: User Interface, Software Interface and Communication Interface. The protocol used shall be HTTP.

4.1 Communications Interfaces

The system shall use the HTTP protocol for communication over the internet.

4.2 Software Interfaces

Front-end software: Angular or simple html/css
Back-end software: Python
Database: PostgreSQL

5. Non-Functional Requirements

5.1 Performance requirements

1. The CPU usage must not be above 30% while interacting with the application.
2. The web application must fully load under 2.5 seconds.
3. The API response time must not exceed 1 seconds.
4. The application must support at least 500 simultaneous users.
5. The web application's RAM usage on a user system must not be above 500MB.
6. The landing page supporting at least 500 concurrent users must provide 5 seconds or less response time, including the rendering of text and images.

5.2 Security requirements

The car registration service is an application where the users will not have to worry about their private data. There will be a username and password authentication. Passwords will be encrypted and with that, we will ensure that no one except the user itself can have the access to their account. If they forget their password, we will put up an option to change it with their login email address.