Programming II

Project Description

Caico: Car Insurance Company

Deadline **Sunday 11.04.2021, 23:59 CET.**

Let us imagine, that we are building up a new software for a car insurance company. The software shall be used by dealers of the insurance company to add new customers and by the customers to file up claims in the case of accidents, thefts etc. The insurance company has additional functionality to get overview of all customers, their payment status and the claims etc.

This programming task is about implementing a web API, so that the same software can be used by different insurance companies to build their own applications using the provided API. At the moment, we do not care about the front-end application, which could be any browser or a mobile app. And we also don't care about a database that stores the information. In this exercise, we deal with the API implementation only. You may use a REST client like Postman to quickly see the results of the API call.

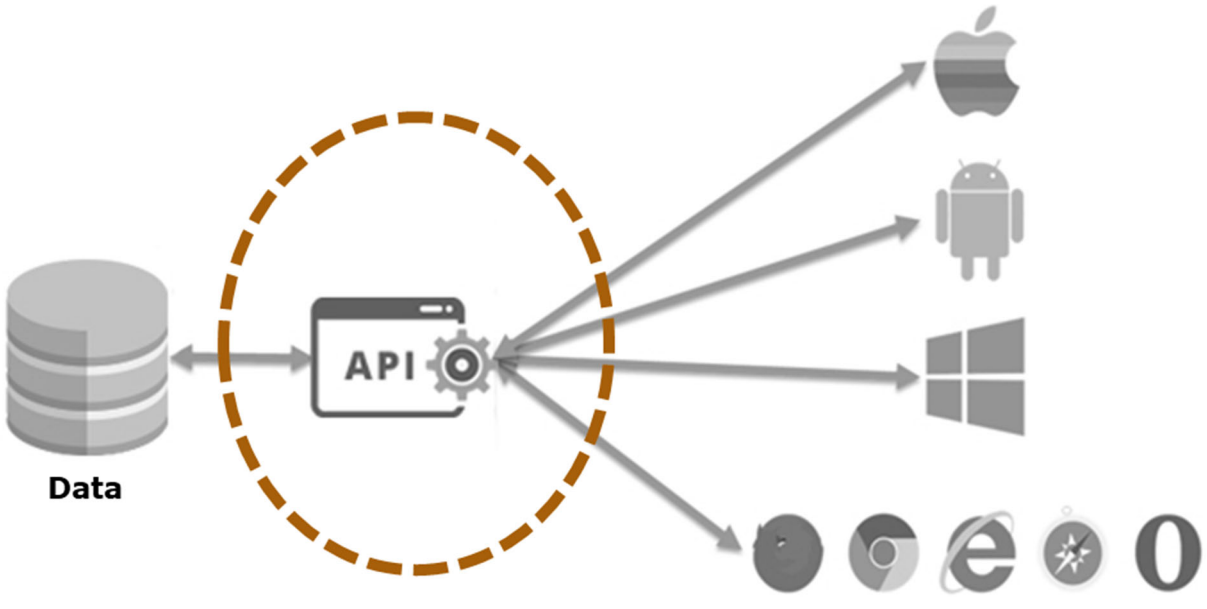
The API consists of the following functionality:

- ➔ Management of customers
  - o Add/remove customers to/from the system. Each customer has a customer-id, name, address and at least one car associated with the customer number.
  - o Add/remove cars to/from customers. Each car has at least the model name, number plate, motor power and the year it was manufactured in.
  - o Each customer has an insurance agent, who is responsible for him/her.
- ➔ Management of insurance agents
  - o Add/remove agents to/from the system. Each agent has an agent-id, name, address.
  - o When an agent is removed, transfer all customers in his/her supervision to another agent first.
- ➔ Management of insurance claims
  - o Customers can file up insurance claims. Such claims are first reviewed by the responsible agent and then passed on to the insurance company. Each claim is assigned a unique claim-id.
  - o Claims are either rejected, partly covered, or fully covered by the insurance policy.
- ➔ Management of financials
  - o The system keeps track of the payments made by the customer.
  - o Based on the number of customers, and their claims, the agents are paid a monthly revenue.
- ➔ Management of general statistics
  - o Display total revenue and profits of the insurance company
  - o Display claim statistics per customer
  - o Display the best agent (customers, claims, …)

The API is implemented in Python using a package called Flask, which allows you to define HTML methods GET, POST, PUT etc. Each method returns a JSON object, which can be used by the front-end application in adequate ways. The summary of HTML methods to be implemented as part of the homework is listed in the following table. The first few methods have already been implemented as examples.

Hints:

- Design your objects and classes allowing for easy future extensions
- Write Test cases to thoroughly test your program – manual testing with Postman is not enough.



Summary of HTTP Methods:

	URL	Type	Description
✓	/customer	POST	Add a new customer (parameters: **name, address**). *An example is given.*
✓	/customer/<customer_id>	GET	Return the details of a customer of the given **customer_id**. *An example is given.*
✓	/customer/<customer_id>/car	POST	Add a new car (parameters: **model, number_plate, motor_power**). *An example is given.*
✓	/customer/<customer_id>	DELETE	Delete the customer with the given **customer_id**. *An example is given.*
	/customers	GET	Return a list of all customers. *Thee given example is not complete. It does not display the cars belonging to the customer.*
	/agent	POST	Add a new insurance agent (parameters: **name, address**).
	/agent/<agent_id>	GET	Return the details of the agent with the given **agent_id**.
	/agent/<agent_id>/<customer_id>	POST	Assign a new customer with the provided **customer_id** to the agent with **agent_id**.
	/agent/<agent_id>	DELETE	Delete the agent with the given agent_**id**. If the agent has customers, move the customers to other agent first.
	/agents	GET	Return a list of all agents.
	/claims/<customer_id>/file	POST	Add a new insurance claim (parameters: **date, incident_description, claim_amount**).
	/claims/<claim_id>	GET	Return details about the claim with the given claim_id.
	/claims/<claim_id>/status	PUT	Change the status of a claim to REJECTED, PARTLY COVERED or FULLY COVERED. Parameters: approved_amount.
	/claims	GET	Return a list of all claims.
	/payment/in/	POST	Add a new payment received from a customer. (parameters: **date, customer_id, amount_received**).
	/payment/out/	POST	Add a new payment transferred to an agent. (parameters: **date, agent_id, amount_sent**).
	/payments/	GET	Return a list of all incoming and outgoing payments.
	/stats/claims	GET	Return a list of all claims, grouped by responsible agents
	/stats/revenues	GET	Return a list of all revenues, grouped by responsible agents
	/stats/agents	GET	Return a sorted list of agents based on their performance. Be creative and come up with a calculation scheme as performance indicator.

Example code implementing the first four methods is provided in a Github repository.

https://github.com/deepak-dhungana/INF-SS20-ProgrammingII

In order to create a new Github account, visit https://github.fhkre.ms and login with your IMC FH Krems account.

Submission of the project must be done through a Github Repository. Create your own repository and upload your code to your private repository before the deadline. Submit the URL of your repository via email.