MTAT.03.229 – Enterprise System Integration

Regular Exam, 3 June 2019

Notes:
- The total duration of the exam is 3 hours and 30 minutes.
- Parts 1 and 2 of are closed-book and closed-laptop. These two parts should be submitted on paper. Once you submit parts 1 and 2, you will be allowed to open your laptop to start part 3.
- Part 3 is open-laptop. Web browsing is allowed. This part should be submitted using the “Submit” button on the course web site.
- You are not allowed to communicate with anyone during the exam in any way (except with the lecturer).

Part 1. Foundations (10 points)

1) [4 points] A simulator of chemical reactions exposes a simple API for submitting simulation jobs. A simulation job is submitted via a POST on /api/simulation/. The request body contains all the input parameters required for the simulation. If the simulation is successful, a 200 code is returned together with the simulation results.

In the backend, the simulation job is handled by a Python script. The script checks the contents of the request, and if it is all correct, it invokes a simulation engine via command-line. When the engine completes, the Python script recovers it and sends it back via the same HTTP connection.

Each simulation job takes between 5 and 30 seconds of CPU time. In case of heavy workload, the backend usually crashes or it becomes too slow. How would you re-architect the backend to make the system more scalable? Do your changes have any impact on the REST API? If so, please explain? Note that the manager does not have a budget to buy more hardware. You should come up with a solution that works with the existing hardware.
2) **[2 points]** A supplier's information system allows customer to submit POs. If a PO is accepted by the supplier, the customer can later submit revised versions of the PO. Customers should be able to query all the revisions of a PO. A junior developer suggests that the operation to submit a revised version of a PO should be exposed as a PUT on /api/sales/pos/{po.id}/ and that the operation to retrieve a given PO revision should be exposed as a GET on /api/sales/pos/{po.id}/revisions/{rev.id} where rev.id is the identifier of the PO revision. Would you agree with this proposal? If not, what would you tell to the developer and what alternative would you propose?
3) [2 points] With reference to the previous question, the junior developer suggests that the operation to retrieve the details of the latest version of a PO should be exposed as a GET on /api/sales/pos/{po.id}/revisions/latest/. Would you agree with this proposal? If not, what would you tell to the developer and what (perhaps simpler) alternative would you propose?
4) **[2 points]** A supplier's information system allows customer to submit POs. If a PO is accepted by the supplier, the customer can later request the PO to be cancelled. The supplier can accept or reject the cancellation request. Sometimes, it takes 24 hours for the supplier to decide if it will accept a cancellation request or not. A junior developer suggests that the operation to submit a cancellation request should be exposed as a DELETE on /api/sales/pos/{po.id}/. Would you agree with this proposal? If not, what would you tell to the developer and what alternative would you propose?
Part 2. Optimum Design’s Sales Information System (16 points)

A Printed Circuit Board assembly (PCB assembly) is a core component in electronic devices. A PCB assembly consists of a circuit board with a number of electronic components (transistors, resistors, capacitors, etc.) connected via conductive tracks.

![Figure 1. Sample PCB Assembly](image)

Optimum Design is a manufacturer of PCB assemblies. Optimum Design manufactures “on demand” based on customer orders. A customer order includes an assembly drawing and a Bill of Materials (BOM). The BOM is a list of electronic components that are required to manufacture the PCB assembly. An assembly drawing is a binary file. We will assume that it is represented by a URL, where the file can be retrieved.

In general, Optimum Design does not keep any stock of electronic components. Instead, it orders these components from its suppliers when required.

We consider below the sales process of Optimum Design, which starts when Optimum Design receives a Request For Quote (RFQ) from a customer. The RFQ includes the customer ID, the assembly drawing, and the BOM.

When an RFQ is received, Optimum Design’s information system automatically requests quotes from Optimum’s suppliers for each component in the BOM. Suppliers take different amounts of time to reply to an RFQ. In some cases, the quotes are given immediately, but in more complicated cases, suppliers may take up to two days to respond to an RFQ. The quotes that suppliers send back to Optimum Design include a price and a delivery time.

Once all quotes for the electronic components of the BOM have arrived, Optimum Design’s system produces a quote based on the manufacturing cost, the cost of the materials, and a profit margin. The quote that Optimum Design sends to its customer includes the price, the expected delivery time and the period of validity of the quote.

Sometimes, customers ask for a revised quote due to minor adjustments in the assembly drawing and BOM. For this, the customer sends a revised RFQ that includes a revised drawing and a revised BOM. A revised RFQ is handled in the same way as an initial RFQ, except that Optimum Design only asks for quotes for items in the revised BOM that have changed with respect to the initial BOM. Optimum Design’s system allows every customer to retrieve the details of any RFQ they have submitted (including revised RFQs) as well as the quote sent back in response to each RFQ.
When a customer is satisfied with a quote, they may opt to submit a Purchase Order (PO). When Optimum Design receives a PO (which should refer to a valid quote) it sends POs to the corresponding suppliers in order to obtain all the required components to manufacture the assembly. To avoid confusion, the PO that Optimum Design receives from a customer is called a Customer PO, while a PO that Optimum Design sends to a supplier is hereby called a Material PO.

A customer may request changes in their PO. To do so, they submit a PO Change Request. If the PCB assembly is already “in production”, the change request is rejected. Otherwise, Optimum Design determines which Material POs corresponding to the Customer PO are affected by the change. It then sends a material PO Change Request to each relevant supplier. A supplier may respond to a PO Change Request by either accepting it without any penalty, or by accepting it subject to a penalty. Once Optimum Design has received all responses to its Material PO Change Requests, it calculates the total cost of the Customer PO Change, and responds to the customer to indicate acceptance of the PO Change Request and the penalty if applicable.

Similarly, customers may request an order to be cancelled by sending a PO Cancellation Request. The procedure for handling PO Cancellation Requests is the same as for PO Change Requests. In particular, cancellations may incur a penalty.

Customers may consult at any given point in time the status of their PO (“awaiting materials”, “in production”, etc.).

**Tasks**

1. Draw a domain model of Optimum Design’s sales information system. [6 pts]

2. Specify a state machine capturing the lifecycle of a Customer PO. Each transition triggered by an external operation should be labelled with the name of the operation (the “relation”), the corresponding HTTP verb and the URL of the resource. [4 pts]

3. Specify the REST API (set of operations) that Optimum Design shall offer to its customers. The specification of a REST operation should include the (relative) URL of each resource, the HTTP verbs each resource accepts, and the relation (operation) that each verb implements over a resource. You do not need to describe the inputs nor the outputs. [6 pts]
Part 3. REST API Implementation and Testing (24 points)

In the practice session of 19 March, we implemented a version of Rentit’s information system that allows us to search plants, create POs, and accept or reject POs. Your task is to extend this implementation with the ability for customers to submit “tentative reservations” for plants.

A customer can send a tentative reservation request for a specific plant inventory item. A tentative reservation request can be accepted at the latest five calendar days before the start of the rental period. For example, on 20 May, a customer can send a tentative reservation request for a plant for the period [25 May..30 May]. If the same tentative reservation request were sent on 21 May, it would be rejected.

Rentit accepts a tentative reservation request (HTTP code 200) if the requested item is available for the full period indicated in the tentative reservation. Otherwise, the tentative reservation request is rejected (code 409). Tentative reservation requests do not need manual approval. They are either accepted or rejected right away, automatically.

Once a tentative reservation request is accepted, the request is valid for two days. During these two days, the customer can be assured that no other reservation for that item will be accepted by Rentit (neither via a tentative reservation nor by accepting a PO). After the two days have passed, the tentative reservation has no effect. One can say that after two days, the tentative reservation simply does not exist, regardless of whether the tentative reservation remains recorded in the database or not.

During the two days that a tentative reservation lasts, the customer can use this tentative reservation to place a PO for the plant inventory item. This is a special type of PO (called a pre-reserved PO). A pre-reserved PO is different from a regular PO because instead of referring to the name of a plant inventory entry (e.g. “bulldozer”), a pre-reserved PO refers to a tentative reservation, which itself refers to a specific plant inventory item (e.g. a bulldozer with serial number ABB20939I). If the customer places a pre-reserved PO and the corresponding tentative reservation is not valid, they get a 409 code back. If it is valid, they get a 200 code, even though the PO still needs to be internally approved or rejected by an employee at Rentit (this is also the case for regular POs).

Task. [Tests: 10 points + Business logic: 14 points]. You are required to implement the operation to create a tentative reservation and the operation to submit a pre-reserved PO. You can choose the URLs where you wish to expose these operations and the HTTP verbs (in line with the principles of REST). You should also write one or more test(s) to ensure that the above requirement is met. Your test(s) should cover the cases where: (i) a tentative reservation is rejected because it is submitted too late; (ii) a tentative reservation is rejected because the plant inventory item is not available for the whole period of the tentative reservation; (iii) a tentative reservation is accepted, and soon after, a second tentative reservation for the same plant inventory item is rejected because the first tentative reservation was accepted; and (iv) a tentative reservation is accepted, and later a pre-reserved PO referring to this tentative reservation is submitted (and code 200 is returned).

To get full points for testing, you should provide the Cucumber feature(s) for your test(s).
You can start by cloning the last commit of the following git repository:
https://bitbucket.org/orlenyslp/esi-2019/

You should keep the separation between the controller, the service and the repository, i.e. the business logic should be in the service and data access should be in the repository.

Your application does not need to have a frontend. It only needs to expose the required REST operations. The tests should be done on the API (i.e. you should call the API’s operations).

You should submit a TXT or PDF file with the link to your Bitbucket repository with the solution. Please leave your solution in the “master” branch of your repository and do not make any commits to “master” after the end of the exam. Please add also an explanation of the changes you made to the API, i.e. new REST operations you added with their URL template and HTTP verb.