CLOUD-BASED ON-LINE CONCERT TICKETING SYSTEM

(A SOFTWARE ARCHITECTURE CAPABLE OF HANDLING PEAK LOADS)

Assignment for Capita Selecta in Software Engineering — October 2019

1. CONTEXT

Tomorrowland —a very successful music festival— is experiencing problems with handling peak loads for their on-line ticketing system. When tickets go on sale, pre-registered users spend hours queuing to purchase tickets to likely get a response that "tickets are sold out".

This is a very frustrating experience. Therefore, the People of Tomorrow are investigating whether a cloud solution (especially the *elasticity* of cloud solutions) would be able to reduce the queuing times. They cannot avoid that many users will get the message "tickets sold out" but this response should come after 10 minutes maximum.

2. ASSIGNMENT

You are asked to provide a software architecture for cloud based on-line ticketing system capable of dealing with peak loads. You must model the system using the "Abstract Behavioural Specification Language" [http://abs-models.org] and run a series of simulations to (a) demonstrate that your architecture indeed is capable of dealing with peak loads while addressing the functional and non-functional requirements listed below; (b) assess the effect of each non-functional requirement on the capacity of the system.

You must present your initial version of your solution 4 weeks from here to a review team. Next you need to validate the architecture by means of a proof-of-concept implementation in 2 sprints of 3 weeks. Last but not least, explain your architecture in a detailed "lessons learned report" (10 pages).

3. FUNCTIONAL REQUIREMENTS

The functional requirements are specified by means of two uses cases.

USE CASE 1: PRE-REGISTRATION

Customers who want to purchase a ticket should pre-register, at least two weeks before the date that tickets go on sale. After providing a unique name and a password, the system will ask for a series of details (name, address, gender, … and credit card details). These are all stored in an internal database. After successful pre-registration, the customer receives a single token (a unique number associated with a customer and encoded as a QRCode) which is required for ordering tickets. The customer can change the registered details.

Success end:

- Tomorrowland has customer details (incl. credit card) stored inside a database.
  + Customer received token.

Use Case 2: Order Tickets

At the time when tickets go on sale, a special purpose web-site opens up where customers can order their tickets. When opening a sale, a given number of tickets are made available and the system is not allowed to sell more tickets than this maximum number.

To purchase tickets, a customer must log-in with the pre-registration token and authenticate by means of the unique name and password associated with that customer. Thus, the combination of unique name - password and token can only make one purchase. In one purchase the maximum number of tickets that may be ordered is four.

Tickets are sold on a first come-first served basis. As long as there are tickets available, the system charges the credit card account with the proper amount and generates a PDF with the tickets ordered. Once the total number of tickets is sold out, the customer receives a message "Tickets are sold out".

Success end:

- *Tickets available:* Tomorrowland has charged credit-card with the appropriate amount
  + Customer received PDF with the ordered tickets.
- *Sold out:* No credit-card transactions have been filed
  + Customer received a message that the tickets are sold out.

4. Non-functional Requirements

The non-functional requirements are listed below. They have an explicit name and a unique number for easy reference.

Create a new iteration of the ABS model for each of the non-functional requirements and assess the impact they have on one another.

- *[NF1] Peak Loads.* The system should scale dynamically to ensure that customers get their tickets or the "sold out message" in less than 10 minutes.
- *[NF2] Encrypted.* All communication between the components in the system runs over an encrypted line. The keys to encrypt messages are stored in a secure key vault. Access to this key vault is a potential bottleneck in the system, thus the performance impact of this key vault should be part of your architecture assessment. You should start from the assumption that the key vault will supply it's answer in a constant amount of time.
- *[NF3] Credit Card Balance.* When a purchase is made and there are still tickets available, the system must verify whether the credit card has sufficient balance to cover the purchase. There are three brands of credit-cards, and each of them has a different service to check the balance. This service is external to the on-line ticketing system, hence the performance impact of these separate balance checks should be part of your architecture assessment. Each of these services is bound by a service-level agreement to respond within a certain time range (best case - average case - worst case); your assessment most cover the whole spectrum.

5. Other Constraints

- You use the *ABS* modelling language and environment at http://abs-models.org.
- This is an *individual assignment*. You are encouraged to discuss solutions among each other but the solution you propose must be yours.

- *Plagiarism* is explicitly forbidden. We will collect the different ABS models and we will run checks for plagiarism on them.
- Your *report* should be 10 pages maximum in the IEEE 2 column conference proceedings style. Templates in word and latex can be found at https://www.ieee.org/conferences_events/conferences/publishing/templates.html.