MTAT.03.295 – Agile Software Development

Regular Exam – 12 January 2021

Part 2. Practice

Online Shopping

You have been hired to implement a system for online shopping. For the first release of the software, you will implement a Phoenix application with a basic set of features to allow users to choose and buy a certain amount of a given product.

The system must support users and products. For users, the system must store their emails, a 6-digit pin code, and their balance. Accordingly, emails must have a valid format and must be unique. The 6-digit pin code may not be unique, but it must be formed only by numbers, e.g., 546330 and 002561 are correct, but A67B90 and 4354 are invalid. Finally, the balance is a non-negative (i.e., >= 0) float number representing the amount of money the user owns to purchase products. For products, the system must store their names (i.e., a string which is unique), the cost of one unit of the product (i.e., a non-negative float number), and the number of items in stock (i.e., a non-negative integer number).

The application must satisfy the following requirements:

R1. For buying a product, the system must display a form in which the user will introduce the product's name (or a sub-string of it) and the maximum price the user is willing to pay (a positive float number). The system must check if the product's name introduced by the user corresponds to an existing product and if its price is at most the amount introduced by the user. Besides, the number of items of the requested product cannot be 0. Note that you should retrieve the matches of the full name of the product, but also a sub-string, e.g., if the user introduces "white", the system must find products named like "white snow", "just-white bread", or "white". If no product matches the searching criteria, the system must display a message notifying the user, including the reason, i.e., if the name is not matching or if the prices are higher than the amount the customer is willing to pay. On the contrary, if at least one product in the database matches the searching criteria, the system will choose the one with the lower cost and display a new form, including product name, cost, and the number of items. This form must not allow the user to change the number of items, name, or cost. The form must also include two inputs to introduce email and PIN-code to confirm the purchase. If the search retrieves multiple products with the same cost, you may choose one of them following the strategy you consider more convenient.

R2. On the confirmation of the purchase, the system must verify/perform the following actions. If the email introduced by the user does not exist in the database, the system will create a new user with the email and PIN provided and with balance 0. It must also notify the user that he needs to add money to his credit before purchasing any product. Note that the system must guarantee that the user's email and PIN codes are correct before updating the database. On the contrary, if the email exists in the database, the system must check the PIN code matches and that the user has enough balance to buy the product. If any of the validations above fails, the system must notify the user about the reason for the failure. Otherwise, the system must update the number of available products and the user's balance after the purchase accordingly. Subsequently, the system must notify the user about the successful purchase. The message must include the name of the product, the total amount of money paid by the user, and his remaining balance. Note that a user can buy only one product per purchase.

The task breakdown and corresponding marks are as follows:

- **Task 1 (4 points) [BDD]:** Specification of a Gherkin user story describing the scenario above as follows. Your user story must specify that at least 3 products and 1 user exist in the

database (providing tables with their data). That followed by how the user starts a search matching only 2 of the existing products (providing the corresponding valid information) and the actions triggered to display the product selected by the system. Next, you must specify how the user introduces valid credentials to confirms the purchase (i.e., the user must also have enough balance). Finally, as a result, the corresponding message to display to the user including, the name of the product, the total amount of money paid, and the remaining balance. Note that you must consider only a successful scenario in which a search happily ends with a purchase, and your feature must include the clauses Given, And, When, and Then. Also, note that generic statements like "the user introduces valid credentials" will not be accepted, as you are supposed to specify those valid credentials you want to check.

- **Task 2 (4 points) [BDD – steps implementation]:** Implementation of the white_bread steps described by the user story in Task 1.
- **Task 3 (2 points):** Setup of the application routes to support the search and purchase of products.
- **Task 4 (6 points):** Setup & implementation of models (via migrations) and seeding the database with some initial data based on your models. The seeding of the database must include both products and users.
- **Task 5 (12 points):** Controllers: you must implement the operations to render the templates to provide the input data for searching and purchasing confirmation of products. You must also implement the operations regarding the search/validation/purchasing confirmation of products (and validation/creation of a user as part of the purchasing confirmation if the user does not exist). Your implementation must fully comply with the specifications and validations described above in this examen, including the messages to notify users.
- **Task 6 (4 points):** Implement the views and templates to introduce the required input data to search for products and purchasing confirmation. Note that you do not need to provide a separate template for user creation as this operation occurs as part of a purchasing confirmation if the user does not exist. Similarly, you do not need to provide any template to create products. Also, note that you do not need to create/render a new template to display the messages. For that, you can just write in the flash as we did during the course practical sessions.
- **Task 7 (8 points):** (TDD) Unit tests for controllers checking the requirements R1 and R2. A single test may include several requirements. Specifically, your tests must consist of at least: R1 – A negative case in which the search matches only one product's name, but the cost is greater than the amount the customer is willing to pay. Thus, the system responds with a message error. One positive case in which at least two products match the search, i.e., including name and cost smaller than the amount introduced and with at least 1 item available, and thus, the system must select the one with the lowest price. The test must consider two products with different costs. R2 – A negative case in which the email introduced exists in the database, and the PIN is not empty, but its format is not correct, i.e., it includes characters that are not integer digits. Thus, the system displays an error. A positive case in which the user exists and the purchase confirmation is executed successfully. The test must check that both the number of products and the user's remaining balance are calculated and updated successfully in the database after the purchase.

Minimum Commits – Messages (NOT necessarily in the same order). Note that each message must correspond to a different commit.

- Initial Commit
- Task 1 Completed
- Task 2 Completed
- Task 3 Completed
- Task 4, Database setup for Products Completed.
- Task 4, Database setup for Users Completed.
- Task 5, R1 Completed
- Task 5, R2 Completed

- Task 6, Product searching template Completed
- Task 6, Purchasing completion template Completed
- Task 7, R1 Completed
- Task 7, R2 Completed