Architectural Thinking for Intelligent Systems

Assignment for Lecture A5

Team Members:
1. Ankit Agrawal, 2581532
2. Anika Fuchs, 2580781

Part 1: Formulate 3 high-level user stories

1) **User story:** As a shopper, I can login to my profile, so that I am able to access my profile on the robot.
   Acceptance criterion: A notification showing 'login successful' can be identified by the customer after logging in with the correct credentials.
2) **User story:** As a local store manager, I can see the notification about any errors that occur directly on the application, so that I am able to provide immediate recovery maintenance.
   Acceptance criterion: The application notifies about errors by sending an alert to the local store manager.
3) **User story:** As a shopper, I can pay for my purchase with PayPal, so that I don't need to wait in a queue to pay manually.
   Acceptance criterion: The customer is forwarded to the PayPal interface after clicking on 'Pay with PayPal'.

Part 2: Check on INVEST (Independent, Negotiable, Valuable, Estimable, Small, Testable)

1) I: Yes, it is independent
   N: Yes, it is negotiable
   V: Yes
   E: Yes
   S: Yes
   T: Yes

2) I: No, dependent on the error handling of the robot.
   N: Yes
   V: Yes, maintenance
   E: Yes
   S: Yes
   T: Yes

3) I: No, dependent on the purchase itself
   N: No, we specified it to be PayPal
   V: Yes

E: Yes
S: Yes
T: Yes

Part 3: Measurability of acceptance criterion

We can ensure the measurability of acceptance tests by following these steps:

1. Acceptance tests should demonstrate a single goal.
2. Acceptance tests should be independent of other tasks or results of other tasks.
3. Broad acceptance test/criteria make a user story vague. Hence, we can write effective acceptance tests that outline the scope of work so that the developers can plan and estimate their effort properly.
4. Create a set of acceptance tests for each user story covering all positive/negative scenarios (error handling) and also consider the range of input/output bounds.
5. We can include the 'rule' and a set of related acceptance tests for additional clarity (This shows the acceptance tests which illustrates that particular rule).
6. Satisfies the definition of done.

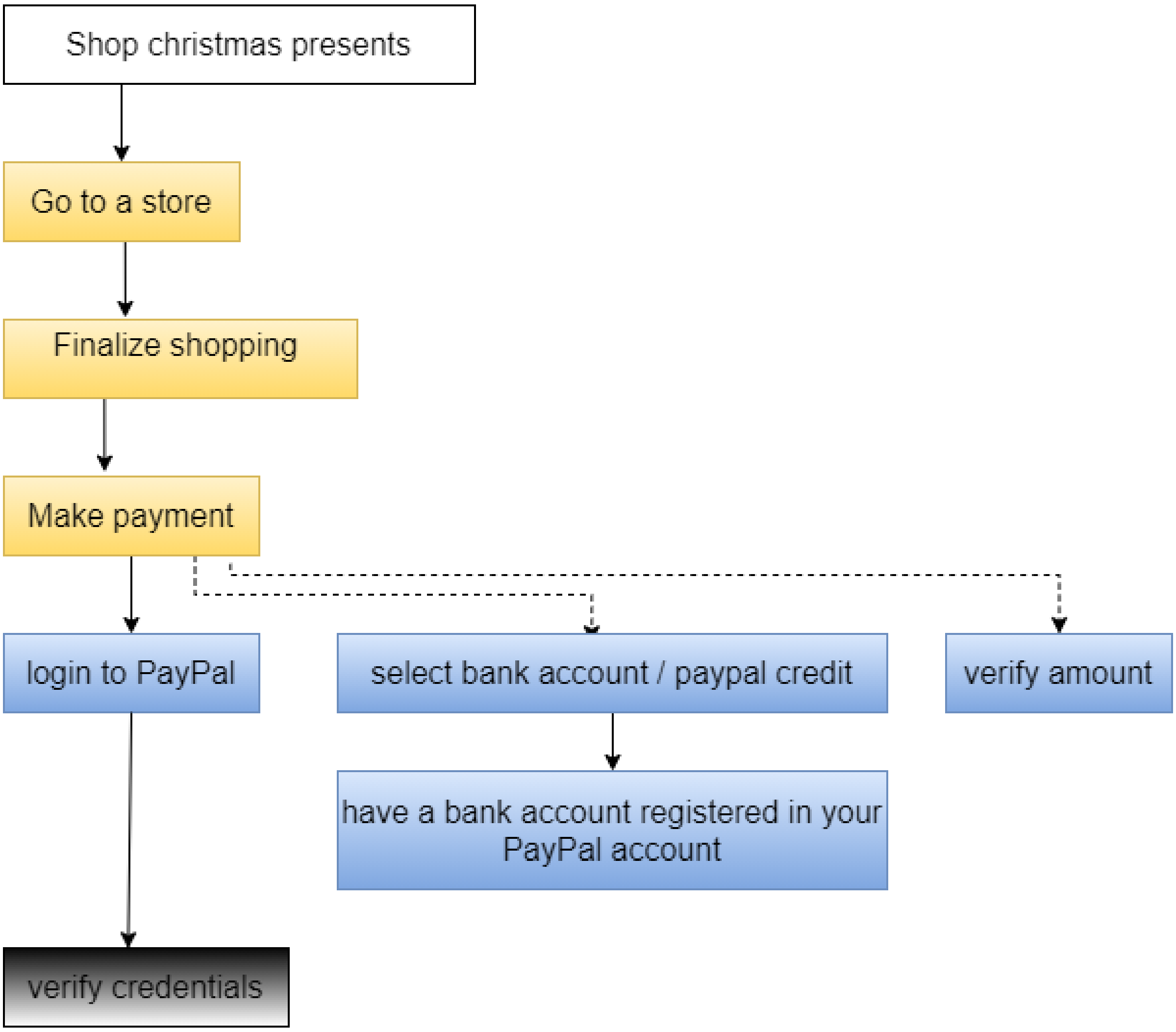
Part 4: Use case to one of the user stories

Use Case for Payment User Story (3)

Short description / goal of the Use Case	Customer pays his/her christmas shopping with PayPal
Actor(s)	Customer / shopper
Preconditions	Shopping / decision process is completed, customer selected 'Go to payment on the UI'
Basic Flow	1. Customer revises the payment amount 2. Customer selects 'Pay with PayPal' in the menu 3. Customer enters his/her PayPal credentials 4. Customer selects a bank account to pay with or PayPal credit. 5. Customer completes payment 6.Customer receives a 'payment successful' notification on the robot 7. Customers can select if he/she wants to receive the invoice and how to receive it. 9. User logout is done automatically 8. Customer can leave the store
Alternative flows	Payment can be invalid, customer could choose

	other payment options; purchase is checked by an employee before being able to pay; payment does not work with several options -> customer needs to pay manually, ...
Postconditions	Debit of payment on customer side, store receives payment
NFR	network connection of the robot -> time, clarity of payment UI.
Special considerations	Security
Author/Date/Revision History	AA, AF, 07/12/2020

Part 5: Goal hierarchy for this use case



Was it high- level enough?
Yes, it describes all the high-level requirements and the flow of the system and dependencies of a step.

Did your use case description show enough detail?
No, it shows the high-level details and the basic flow, but we don't have much technical knowledge, also we don't have the conditional cases in the diagram.