Lab 2 : Service virtualisation

In this assignment you will develop a text based HMI for the credit card payment procedure (figure 1). First you will develop a component that will read the magnetic stripe of a credit card and validate it based on the Luhn algorithm. The component will then be used in developing the backend for the credit card payment system. A virtual service has been developed in place of the card reader and you will develop a driver that reads numbers from the service.

Use case:	**Credit card payment**
Actors:	Customer, bank
Purpose:	Make a credit card payment
Overview:	The customer swipes his card, inputs PIN and approves amount.
Scope:	Payment subsystem
Level:	Subfunction
Preconditions:	System is running. Amount received from check-out system.
Postconditions:	The amount has been paid
Special requirements:	If the transaction fails, the customer must be prompted to start over.
Actor action	**System response**
1. Customer swipes the credit card.	
	2. Card number is verified and buffered. Customer is prompted to input PIN
3. Customer inputs PIN	
	4. PIN is buffered. Customer is prompted to approve amount.
5. Customer approves amount	
	6. Card number, PIN and amount are encrypted and send to the bank
	7. Bank approves the transaction.
Alternative flow of events	
Line 2 : Invalid card number. Indicate error. Ask user to swipe again. Return to step 1.	
Line 5 : Customer does not approve the amount. Return to step 1.	
Line 6 : Wrong PIN returned from bank. Return to step 1.	
Line 7 : Bank rejects the transaction. Ask customer to pay with different method.	

Fig. 1 Use case narrative of the credit card payment

Service virtualisation

For easy testing without depending on expensive hardware, a virtual service has been developed in place of the card reader. The interface is the file /dev/swipe which will output a correct card number at the press of button 1 on the ZYBO and an incorrect card number at the press of button 2. The correct number is: 1234 0987 4321 7895, the incorrect number is 4321

5432 6543 7654. In the latter case, the last digit should have been a 6 instead of 4 to be correct. Use this calculator if you need more numbers for checking your implementation.

Driver

The driver reads from /dev/swipe and presents the data for other parts of the program. The driver must be reusable for later virtualisations following the same pattern for example for the barcode scanner. This means you must make sure the driver has low coupling and that it can be easily exchanged with another driver without requiring changes to the rest of the code (the backend).

Write a C/C++ program that uses the driver to read the number from the buffer and validate it based on the Luhn algorithm. If the number is valid, the program must write the card number on standard out. If the number is invalid, it must append timestamp and error message to a log file. The error message must contain the wrong card number.

Requirements specification for driver and validation

#	Requirement	Validation
R-1	Validation must be handled in software	Design inspection
R-2	Validation must be done according to the Luhn algorithm	Design inspection
R-3	Low coupling between the driver and verification	Design inspection
R-4	Driver must read from /dev/swipe	Design inspection
R-5	Must write error descriptions to a log	Design inspection
R-6	Must write validated numbers to standard out	Design inspection

HMI Backend

Write the control structure of a backend for the system following the state machine diagram in figure 2. The backend should be presented as a text interface in the terminal. When started the program must prompt the user to insert card and then wait for the card to be inserted. Upon insertion of a valid card, the user is prompted to input PIN and then to approve the amount from the check-out. Since receiving the amount is a prerequisite for the use case, you can use a static number. When the amount is approved the transaction should be written to the log as a service virtualisation of the actual transaction and omitting the encryption. The program must also write relevant debugging information to the log file to allow inspection of state changes and parameter passing while running.

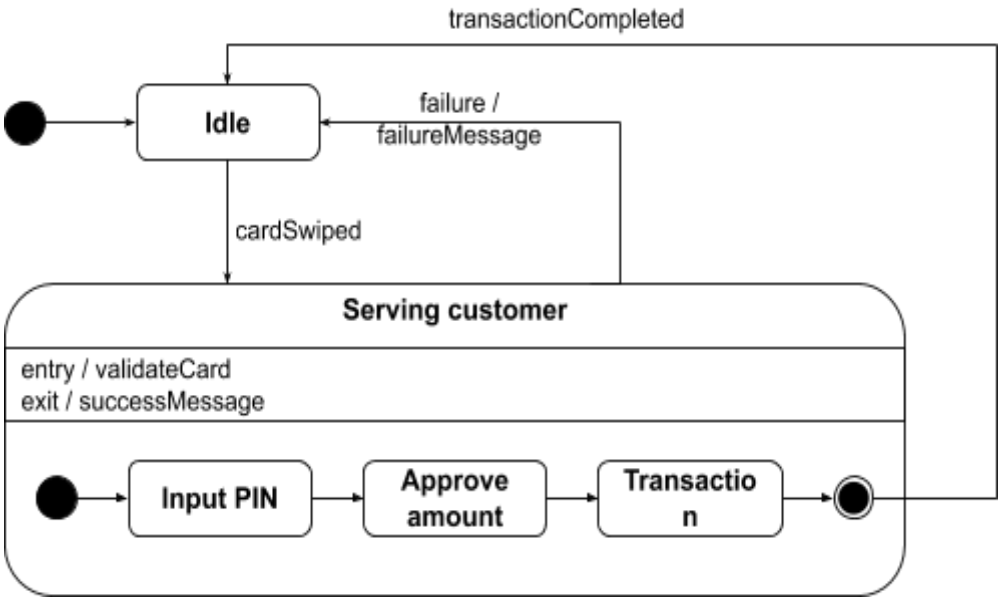

Fig 2. State diagram of the backend

Requirements specification for back-end

#	Requirement	Validation
R-7	Text based user interface	Design inspection
R-8	Validate card number based on Luhn	Design inspection
R-9	Tolerant to wrong inputs	User test
R-10	Times out if user inactive for 10 seconds	User test
R-11	Write debug information to log file	Design inspection
R-12	Write transaction data in log file	Design inspection