ATM System

Program:

Course Code: CSE 326

Course Name: Software Formal Specification

Examination Committee

Dr. Islam El Maddah

Ain Shams University
Faculty of Engineering
International Credit Hours Engineering Programs (I-CHEP)

Spring Semester – 2020

Student Personal Information for Group Work

Student Names:
Yara Hossam Mohamed
Ayman Hesham Mohamed

Student Codes:
16P3002
16P3037

Plagiarism Statement

I certify that this assignment / report is my own work, based on my personal study and/or research and that I have acknowledged all material and sources used in its preparation, whether they are books, articles, reports, lecture notes, and any other kind of document, electronic or personal communication. I also certify that this assignment / report has not been previously been submitted for assessment for another course. I certify that I have not copied in part or whole or otherwise plagiarized the work of other students and / or persons.

Signature/Student Name: **Yara Hossam Mohamed, Ayman Hesham Mohamed** **Date: 22/05/2020**

Submission Contents

01

First Topic

ATM Description

1. ATM description:

"ATM" or ***"automated teller machine"*** or ***"ABM" (automated bank machine)*** or ***"cash machine"*** is an electronic banking outlet distributed worldwide that offer and serve to the customer many options to do basic transactions without need of branch teller. ATM could be used by anyone having a credit or debit card

ATMs allow people to have quick service alone for example: bill payments, withdrawals, transfers between accounts … each ban has its own charges for cash withdrawals either by the operator of the ATM or the bank itself. We can avoid those fees by using the operated ATM for the holder's account.

Taking into consideration that ATMs must not make wrong operations because it must be confidential like the teller in the bank such as giving the right amount of money that the customer needs or show his right balance without having any error.

02

Second Topic

why to describe in formal way and not casually

- **Why to develop formally:**

We must describe our project specification in a formal way because of many reasons

First point, most of the system uses a software part that must be confidential, secured and made without any error that can cause the failure of the proceed of the function of the system because the safety isn't considered now to the hardware only but the software must be also secured.

Second point, due to the high technology most of the systems that serve a task for the humans are very complicated so the specifications must be clearly formally presented to the developers.

Third point, while using a formal description it is easier to detect any defects or errors or missing information made by the developers while making the frequently testing and it helps the testing team to detect this errors earlier as soon as possible.

Fourth point, the importance of using a formal description of the requirement to the system that it must be very understood to the developing team so they don't do any misconception mistakes or misunderstanding mistakes that make the system fail or crash.

Fifth point, formal specification helps the developing team to fast recognize the requirements needed for the system as they are described in a way that helps them to start directly the planning phase without the need to make meeting with the stakeholders for a missing information or misunderstanding information.

- **Weakness of formal approaches:**

1. **Contradictions:** a main problem in using the formal specification is when there is more than one statements that do not agree with each other
2. **Ambiguities:** is a problem that we find when there is more than one interpretation to the statement
3. **Vagueness:** in the large documents we can find many specifications that are not written in a precise way enough that could result “vague” specification
4. **Incompleteness:** when there is a failure in listing the limitations and error handling for every function
5. **Mixed levels of abstractions:** this problem could happen into a system with statements considered as abstract which are intermixed in a random manner with other statements which are written at a lower level of details

03

Third Topic

Informal Requirements

The ATM must have the ability to offer to the customer those services:

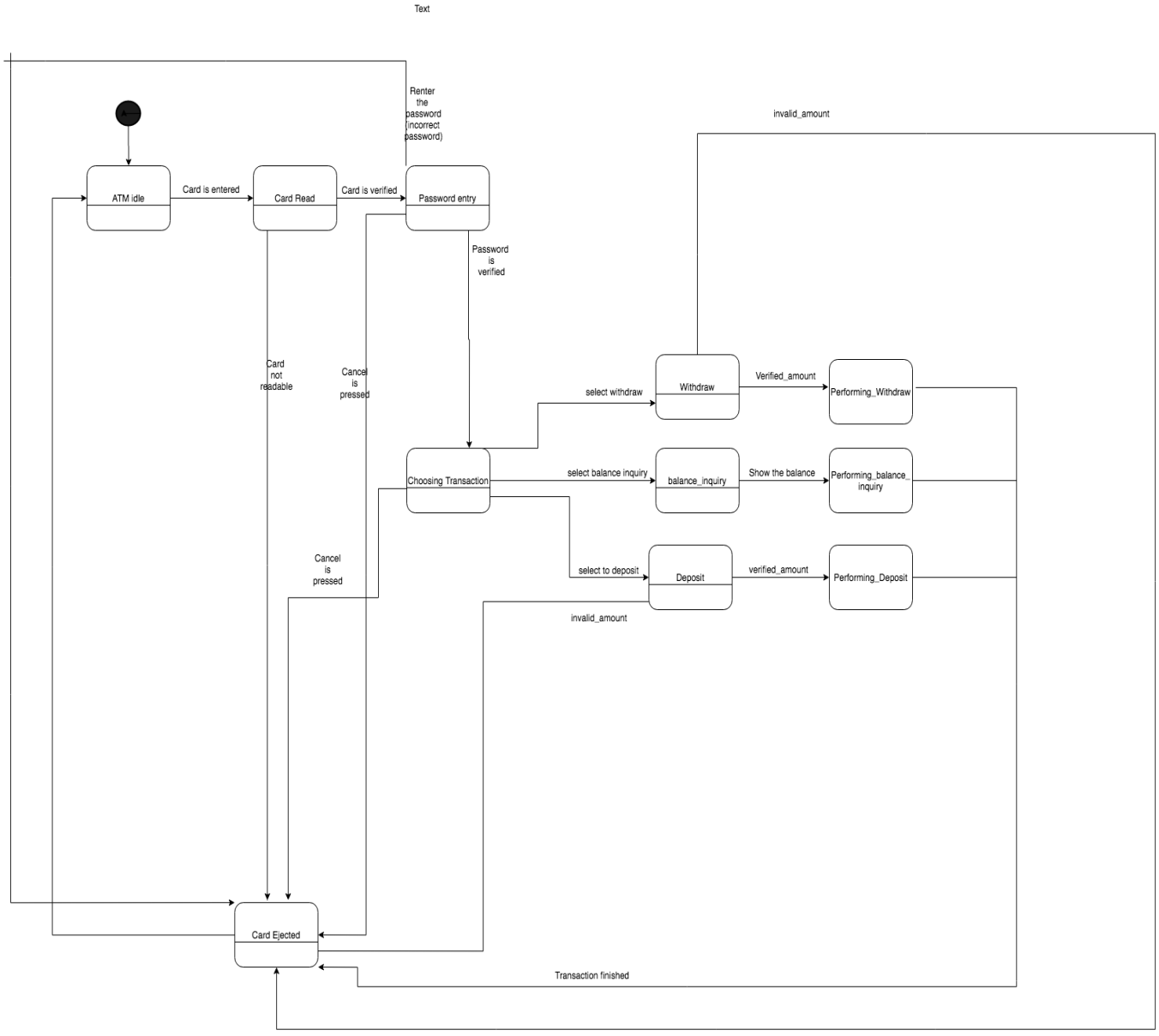
- It must allow the customer to make a cash withdrawal from any linked account to his card taking into consideration that his current balance has to be more or equal the required amount of withdraw and with a maximum amount limit (5000LE)
- It must allow the customer to deposit an amount of cash to any linked account to the card by entering the amount in the ATM and it must verify that the money is not forged money.
- It must allow customer to transfer amounts of money between any two linked accounts to the card and with a maximum amount limit(5000LE).
- It must allow customer to make a balance inquiry.

04

Fourth Topic

Formal Requirements Using SMV

- **State Diagram:**



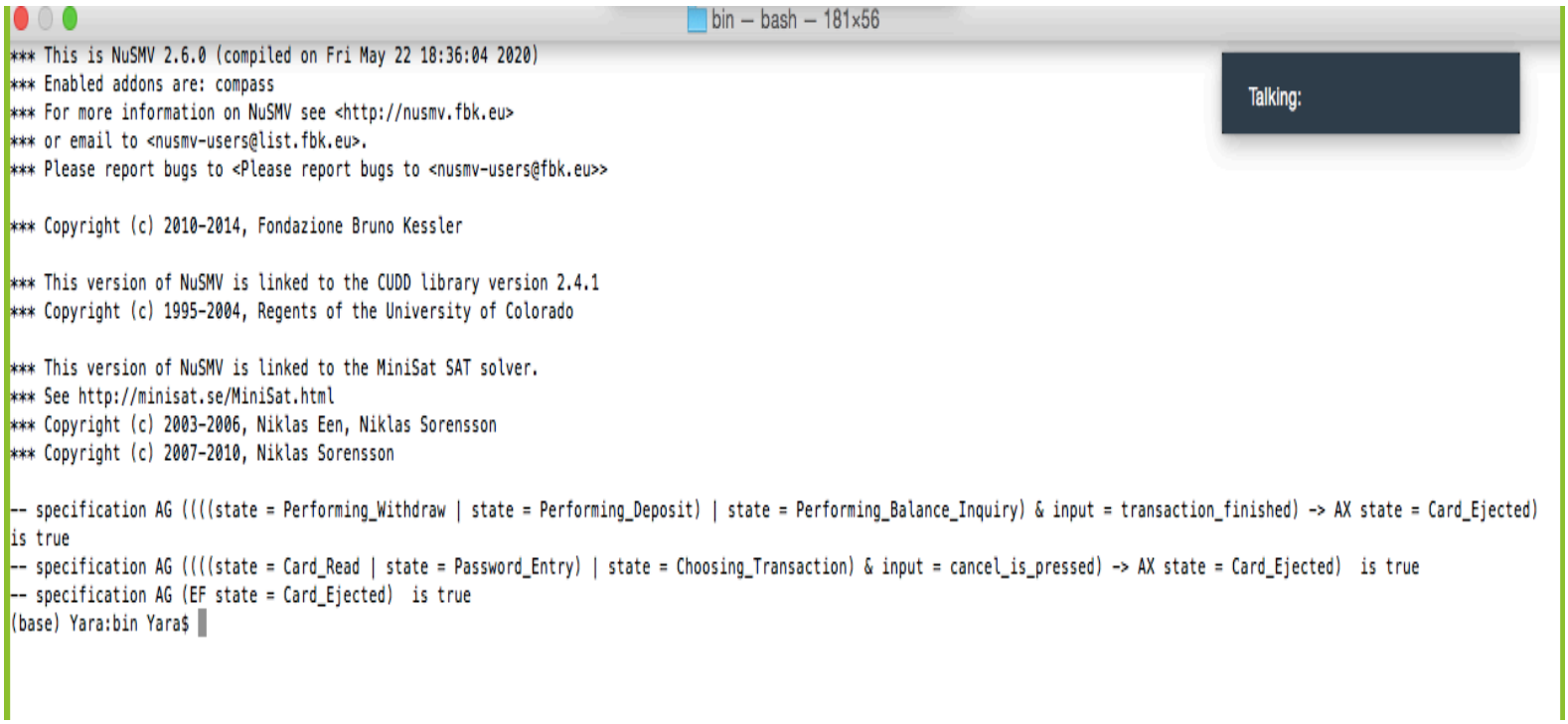
```
MODULE main

VAR

  state:  {ATM_Idle, Card_Read , Password_Entry  , Choosing_Transaction,
Withdraw, Deposit , Balance_Inquiry , Performing_Withdraw , Performing_Deposit ,
Performing_Balance_Inquiry ,Card_Ejected};

  input: {Card_is_entered, Card_is_verified , Password_is_verified , Select_withdraw
, select_deposit , select_balance_inquiry , transaction_finished , cancel_is_pressed ,
wrong_pass  ,invalid_card , amount_is_verified , show_balance,invalid_amount};

ASSIGN

  init(state) := ATM_Idle;


  next(state) := case

    state = ATM_Idle   & input = Card_is_entered     : Card_Read;

    state = Card_Read  & input = Card_is_verified : Password_Entry ;

    state = Card_Read  & input = invalid_card   : Card_Ejected;

    state = Password_Entry  & input = Password_is_verified       :
Choosing_Transaction;

    state = Password_Entry & input = wrong_pass    : Card_Ejected;


    state = Choosing_Transaction & input = Select_withdraw    : Withdraw;

    state = Choosing_Transaction & input = select_deposit    : Deposit;

    state = Choosing_Transaction & input = select_balance_inquiry    :
Balance_Inquiry;
```

```
    state = Withdraw & input =  amount_is_verified   : Performing_Withdraw;
    state = Deposit & input =  amount_is_verified    : Performing_Deposit;
    state = Balance_Inquiry   & input = show_balance   : Performing_Balance_Inquiry;

    state = Performing_Withdraw  & input =  transaction_finished : Card_Ejected;
    state = Performing_Deposit & input =  transaction_finished   : Card_Ejected;
    state = Performing_Balance_Inquiry   & input =  transaction_finished :
Card_Ejected;

    state = Withdraw & input =  invalid_amount  : Card_Ejected;
    state = Deposit & input =  invalid_amount  : Card_Ejected;

    state = Card_Read  & input = cancel_is_pressed  : Card_Ejected;
    state = Password_Entry & input = cancel_is_pressed  : Card_Ejected;
    state = Choosing_Transaction & input = cancel_is_pressed  : Card_Ejected;
    TRUE : state;
  esac;

SPEC AG ( ((state = Performing_Withdraw) | (state = Performing_Deposit) | (state =
Performing_Balance_Inquiry)) & (input = transaction_finished  ) -> AX state =
Card_Ejected );
SPEC AG ( ((state = Card_Read) | (state = Password_Entry) | (state =
Choosing_Transaction)) & input  = cancel_is_pressed  -> AX state = Card_Ejected );
SPEC AG ( EF (state = Card_Ejected) );
```

05

Fifth Topic

Running of SMV



Output:

```
Last login: Fri May 22 23:31:44 on ttys000
(base) Yara:~ Yara$ cd /Users/Yara/Downloads/NuSMV-2.6.0-Darwin/bin
(base) Yara:bin Yara$ nusmv atm.smv
*** This is NuSMV 2.6.0 (compiled on Fri May 22 18:36:04 2020)
*** Enabled addons are: compass
*** For more information on NuSMV see <http://nusmv.fbk.eu>
*** or email to <nusmv-users@list.fbk.eu>.
*** Please report bugs to <Please report bugs to <nusmv-users@fbk.eu>>

*** Copyright (c) 2010-2014, Fondazione Bruno Kessler

*** This version of NuSMV is linked to the CUDD library version 2.4.1
*** Copyright (c) 1995-2004, Regents of the University of Colorado
```

```
*** This version of NuSMV is linked to the MiniSat SAT solver.
*** See http://minisat.se/MiniSat.html
*** Copyright (c) 2003-2006, Niklas Een, Niklas Sorensson
*** Copyright (c) 2007-2010, Niklas Sorensson

-- specification AG ((((state = Performing_Withdraw | state =
Performing_Deposit) | state = Performing_Balance_Inquiry) & input =
transaction_finished) -> AX state = Card_Ejected)  is true
-- specification AG ((((state = Card_Read | state = Password_Entry) |
state = Choosing_Transaction) & input = cancel_is_pressed) -> AX state
= Card_Ejected)  is true
-- specification AG (EF state = Card_Ejected)  is true
     (base) Yara:bin Yara$
```