VENDING MACHINE FSM

GROUP MEMBERS :

Vivek Amit Sheth - M00827627

Arron Suraj Singh Lohia - M0081447

Kabita Purja - M00664409

Shreyansh Narola - M00833534

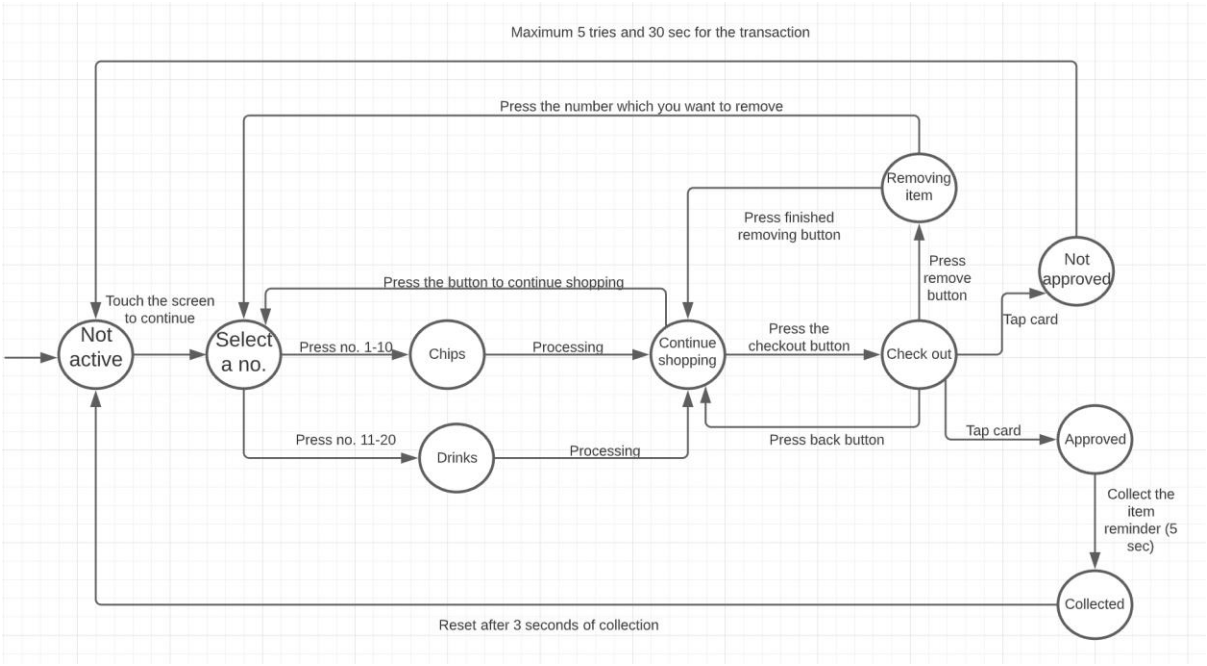
Introduction:

The objective of this project is to create a program that will stimulate the behavior of an FSM. We will implement this in RACKET and use lists and various functions in RACKET to represent the FMS. We have chosen a vending machine to display how an FSM will function in RACKET.

How the vending machine works?

At the beginning, the vending machine will be in the not-active state and when a customer wants to use the vending machine , then firstly the customer have to touch the screen to activate the machine and after the machine is active the machine asks the customer to enter a number which is dedicated to a chips packet or a drink and when the customer chooses any number then the machine processes the input and goes to the continue shopping state where the customer can add more items to their check-out bag or they can check-out as it is. If the customer wants to add or remove any item from the check-out bag then there is a dedicated remove button and add button in the screen at the checkout state after the customer is done with his/her shopping , the customer can checkout by tapping their bank card on the transaction machine. If the transaction is not approved then there will be maximum 5 tries and after the tries are over the machine will go back to the not active state and if the transaction is approved then the machine will go to the collected state where the machine will remind to collect the item for 5 sec and after the item(s) are collected then the machine will go back/reset itself after 3 sec and will go back to the not active state.

Diagram of our finite state machine:

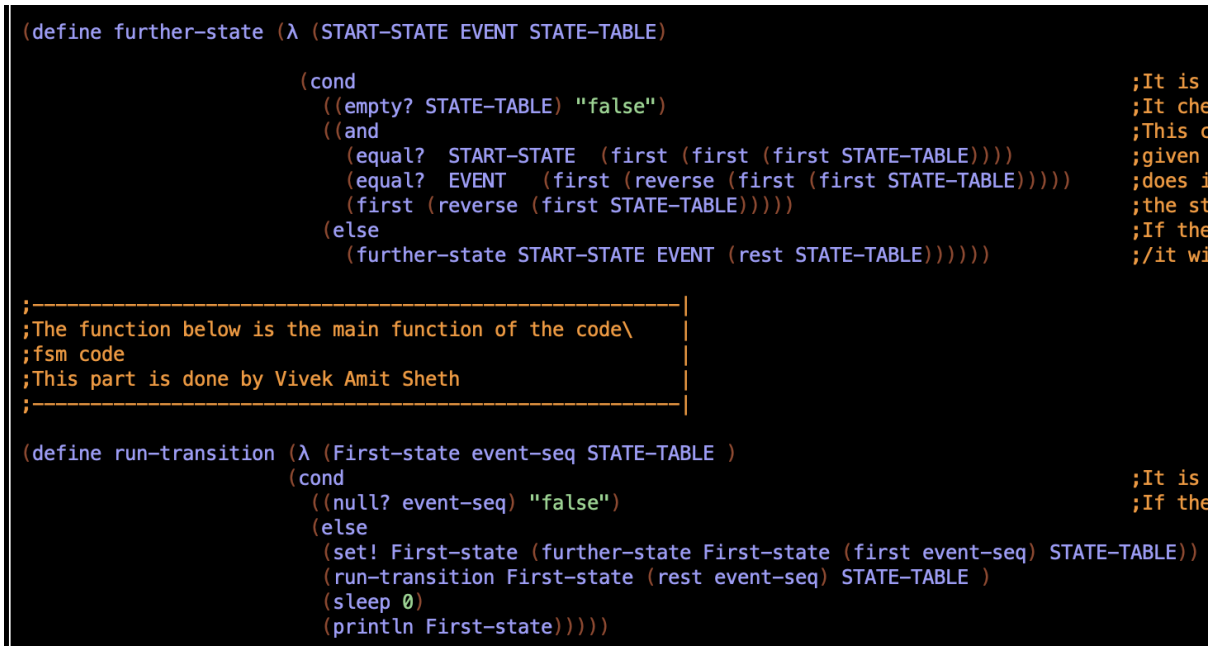


STATE TABLE FROM THE RACKET CODE:

```
(define STATE-TABLE
  '(
    (("NOT ACTIVE" "TOUCH THE SCREEN") "SELECT A NUMBER")

    (("SELECT A NUMBER" 1) "CHIPS - 1")
    (("SELECT A NUMBER" 2) "CHIPS - 2")
    (("SELECT A NUMBER" 3) "CHIPS - 3")
    (("SELECT A NUMBER" 4) "CHIPS - 4")
    (("SELECT A NUMBER" 5) "CHIPS - 5")
    (("SELECT A NUMBER" 6) "CHIPS - 6")
    (("SELECT A NUMBER" 7) "CHIPS - 7")
    (("SELECT A NUMBER" 8) "CHIPS - 8")
    (("SELECT A NUMBER" 9) "CHIPS - 9")
    (("SELECT A NUMBER" 10) "CHIPS - 10")


    (("SELECT A NUMBER" 11) "DRINK - 1")
    (("SELECT A NUMBER" 12) "DRINK - 2")
    (("SELECT A NUMBER" 13) "DRINK - 3")
    (("SELECT A NUMBER" 14) "DRINK - 4")
    (("SELECT A NUMBER" 15) "DRINK - 5")
    (("SELECT A NUMBER" 16) "DRINK - 6")
    (("SELECT A NUMBER" 17) "DRINK - 7")
    (("SELECT A NUMBER" 18) "DRINK - 8")
    (("SELECT A NUMBER" 19) "DRINK - 9")
    (("SELECT A NUMBER" 20) "DRINK - 10")

    (("CHIPS" "PROCESS") "CONTINUE SHOPPING")

    (("DRINKS" "PROCESS") "CONTINUE SHOPPING")


    (("CONTINUE SHOPPING" "PRESS CHECK OUT") "CHECK-OUT")
    (("CONTINUE SHOPPING" "PRESS CS") "SELECT A NUMBER")

    (("CHECKOUT" "TAP CARD") "NOT APPROVED")
    (("CHECKOUT" "TAP CARD") "APPROVED")
    (("CHECKOUT" "PRESS REMOVE") "REMOVING ITEM")
    (("CHECKOUT" "PRESS BACK") "CONTINUE SHOPPING")


    (("REMOVING ITEM" "PRESS FINISH REMOVE") "CONTINUE SHOPPING")
    (("REMOVING ITEM" "PRESS THE NUMBER") "SELECT A NUMBER")

    (("NOT APPROVED" "MAX 5 TRIES") "NOT ACTIVE")

    (("APPROVED" "COLLECT THE ITEM") "COLLECTED")

    (("COLLECTED" "RESET") "NOT ACTIVE")

    ))
```

The state table mentioned above is from the racket code which shows the states and events from the finite state machine (FSM) of the vending machine.

Following States :



```
(define further-state (λ (START-STATE EVENT STATE-TABLE)

                        (cond                                                          ;It is
                          ((empty? STATE-TABLE) "false")                               ;It che
                          ((and                                                        ;This c
                             (equal?  START-STATE  (first (first (first STATE-TABLE))))  ;given
                             (equal?  EVENT   (first (reverse (first (first STATE-TABLE)))))  ;does i
                             (first (reverse (first STATE-TABLE)))))                   ;the st
                          (else                                                        ;If the
                            (further-state START-STATE EVENT (rest STATE-TABLE))))))   ;/it wi

;-----------------------------------------------------|
;The function below is the main function of the code\    |
;fsm code                                                |
;This part is done by Vivek Amit Sheth                   |
;-----------------------------------------------------|

(define run-transition (λ (First-state event-seq STATE-TABLE )
                       (cond                                                         ;It is
                         ((null? event-seq) "false")                                 ;If the
                         (else
                          (set! First-state (further-state First-state (first event-seq) STATE-TABLE))
                          (run-transition First-state (rest event-seq) STATE-TABLE )
                          (sleep 0)
                          (println First-state)))))
```

This code mentioned above is the run-sequence.

TOTAL SOBS that were included in the project:

7,8,101,102,10,12,21,113,217.