NP000573-CT108-3-1-Python assignment

by Birendra Dahal

Submission date: 17-Aug-2021 11:08PM (UTC+0800)
Submission ID: 1629428200
File name: python_final.docx (1.59M)
Word count: 3448
Character count: 28130

Introduction

The project's purpose of this assessment aims is to develop a single Python program in which bank administrators and purchasers can use to manage credit. Each part possesses its own set of modules and every segment is composed of order and clear strands. It's also possible to create a different collection book which is record from a different party. The chief relates to everyone's documents and has the right to authorize new hires, change information and other records, and issue clear credit identifications to buyers. On the other hand, the administrators will provide clients with access to a wide range of entry points and sections. Clients are also two-part division groups: "registered" and "unregistered". Each of these two types of purchasers has a smaller and different market share. Note buyers must use a unique certification method to access the purchaser's door, which requires them to click on their loading credentials. This task was developed to meet a large number of requirements.

Presupposition:

1. Create a loading framework for the administrators and client parts, and set up a separate loading procedure foreach different client title and password. Purchasers may be able to access their chosen entrance if their credentials are successfully provided.
2. Text documents should be created for a variety of applications, as needed. As a sample, a single item document is going to use to maintain a record of purchaser's individual documentation, as well as another used to maintain tabs on the situation purchaser's respect advice.
3. Manufacture multiple variables and each variable has a title that is distinct.
4. Each section has been archived for future reference in the course of the interaction of the encoding test records to address recurring difficulties in future development.
5. After entering the credit quantity, credit residence and financing costs, the prepaid credit calculator calculates the required monthly payments.

Greeting in the bank of Malaysia online credit and management order (MBOLMS) application form. All credit, portion, and payment information are here. You may use the website to check the payment era and follow your bank activities. Purchasers' information may also be found by administrators, and orders can be authorized or denied. Python scripts make it simple to carry out each of these tasks.

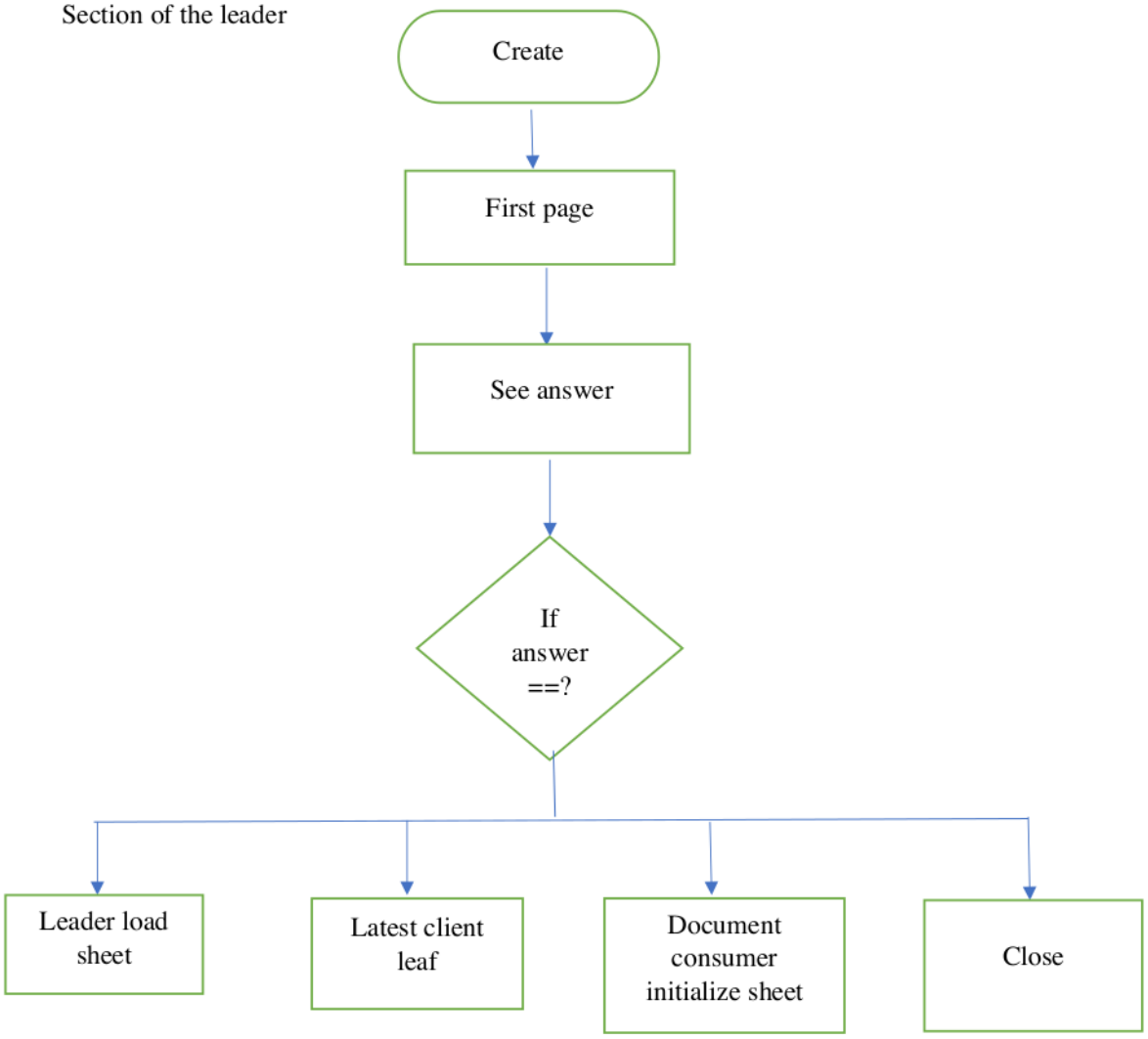
Assumption:

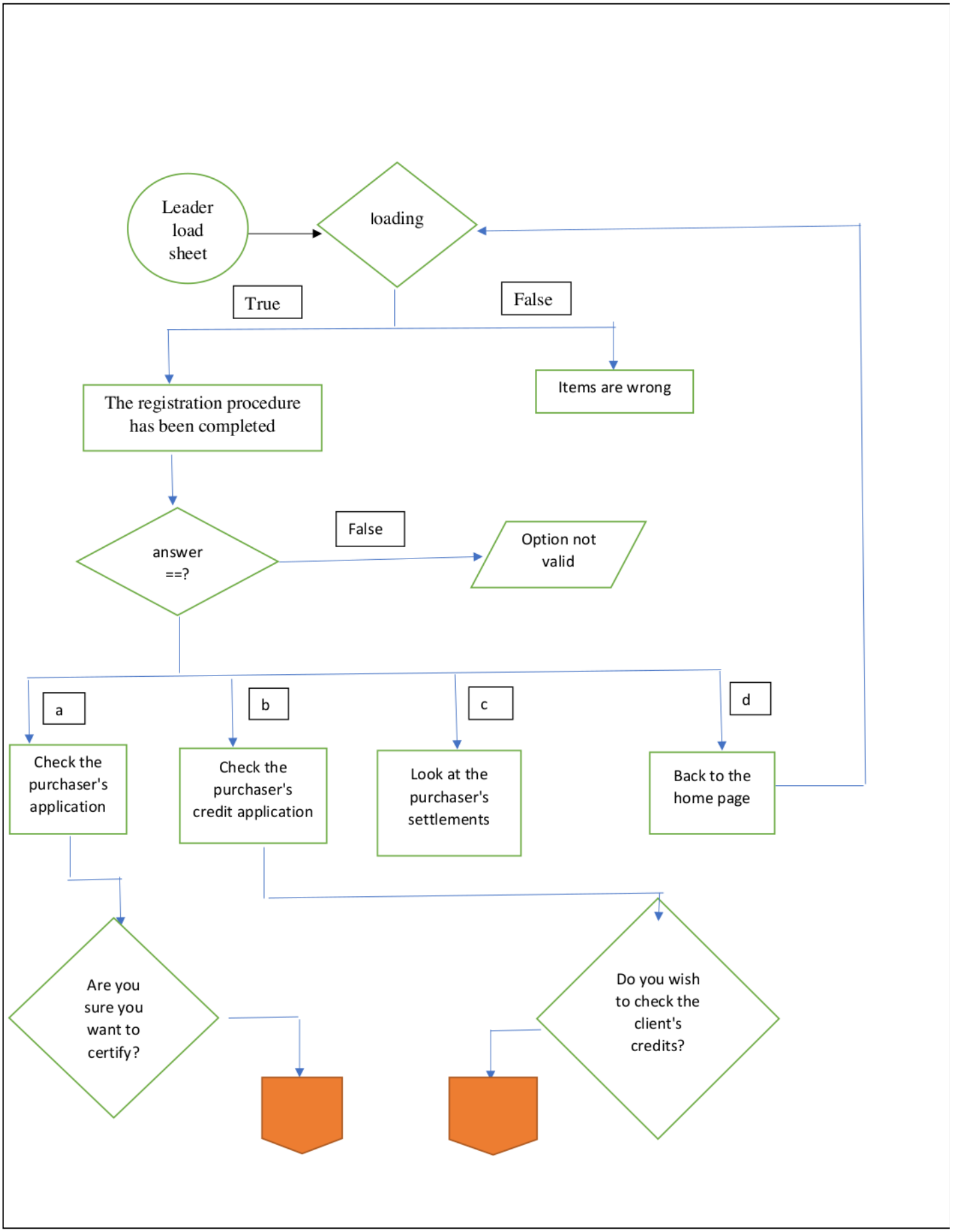
a) I guess the order can save administrators and purchaser information.
b) I assume that using their ID and password, new or current clients can log in or register.
c) administrators think they have complete control over all client documentation and applications, and that they have the authority to approve or refuse them.
d) I believe the credit application you provided can be evaluated by the client.
e) I accept that clients can track their settlements and credit approval by the servicer.
f) I believe that consumers can calculate credit cost and lead time.
g) Purchasers can enlist on the website before using/ establishing a new ID, password, email address, phone number, and birthdate, according to us birth date.
h) Client and leaders, I presume, may simply register or access the order from the main card to use it.

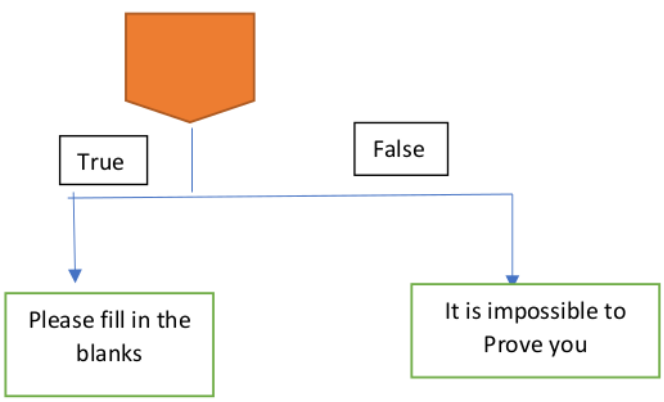
Flowchart

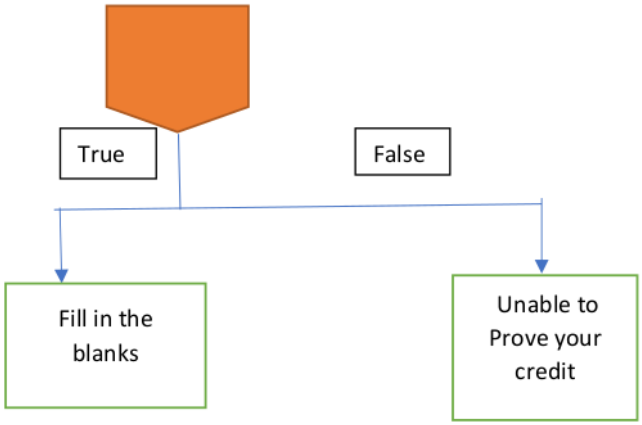
Flowchart overview

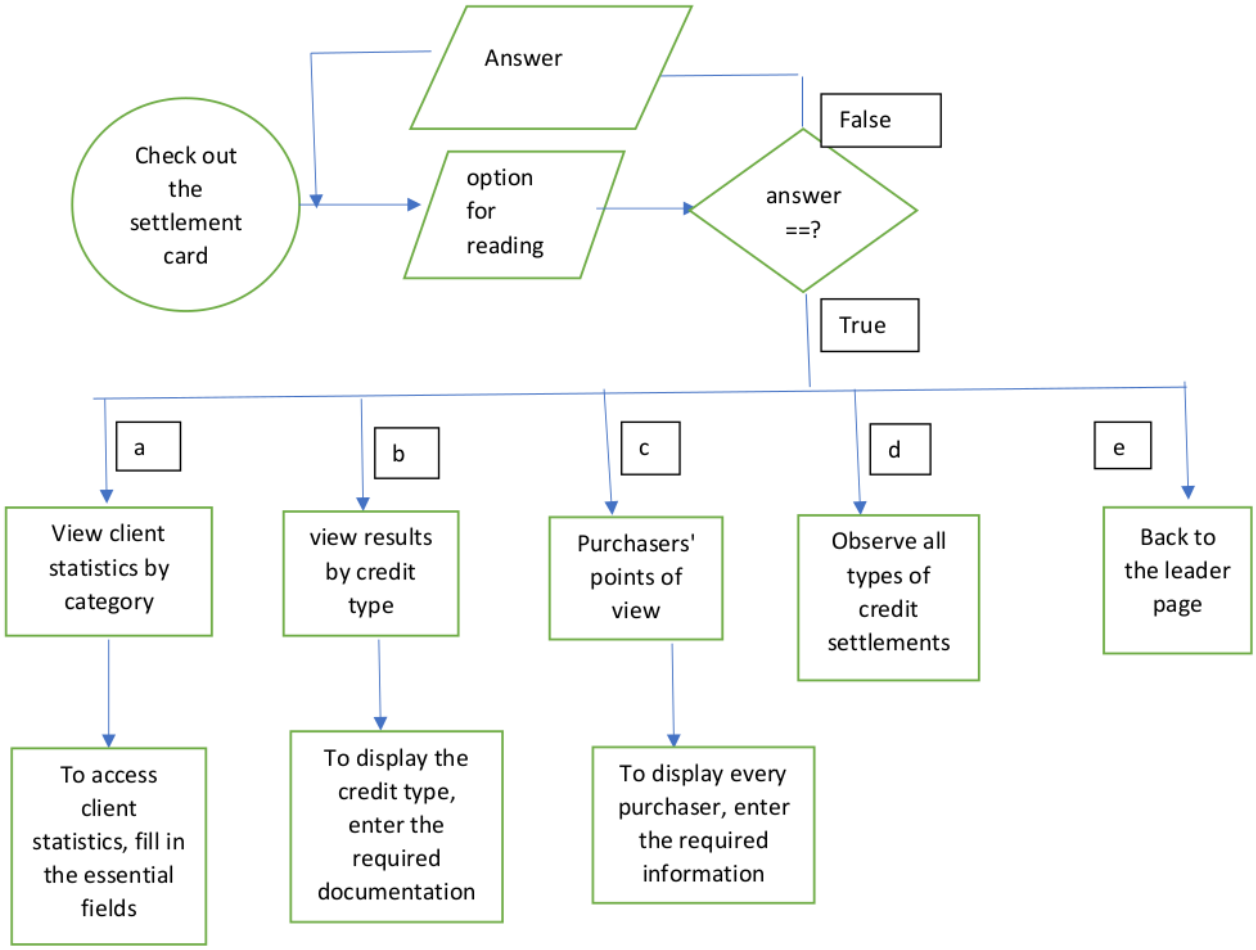
Section of the leader

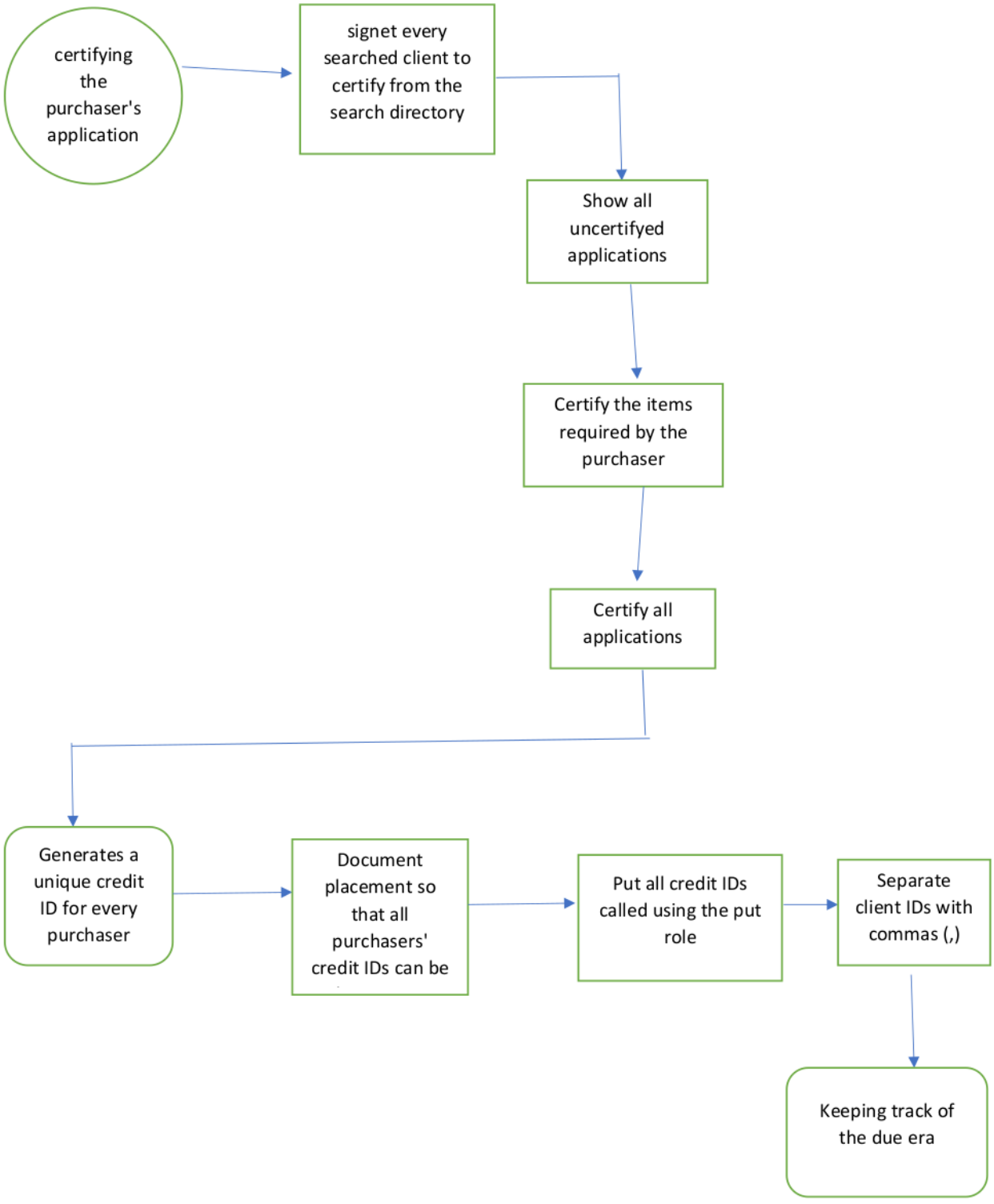


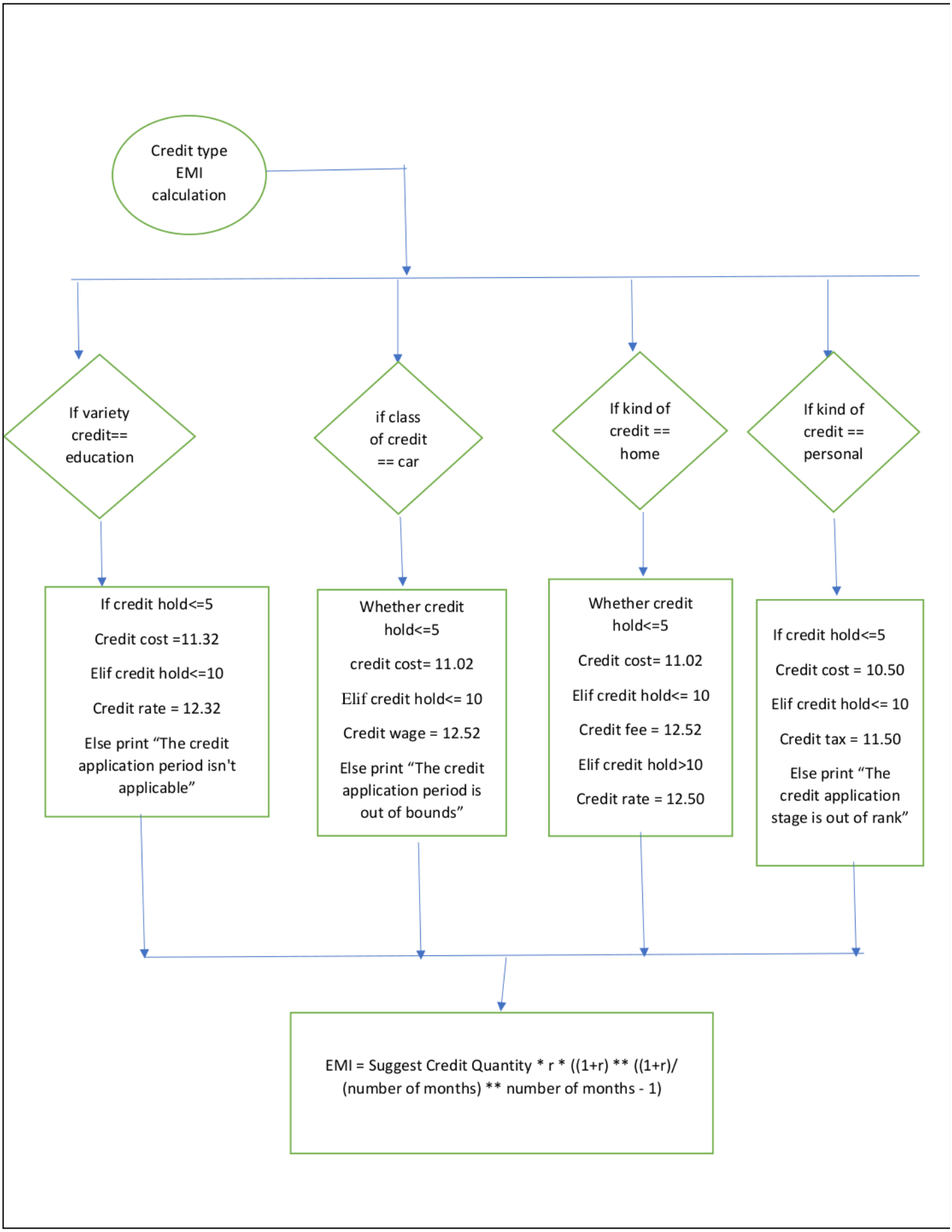

Leader load sheet
loading
True
False
Items are wrong
The registration procedure has been completed
answer ==?
False
Option not valid
a
b
c
d
Check the purchaser's application
Check the purchaser's credit application
Look at the purchaser's settlements
Back to the home page
Are you sure you want to certify?
Do you wish to check the client's credits?

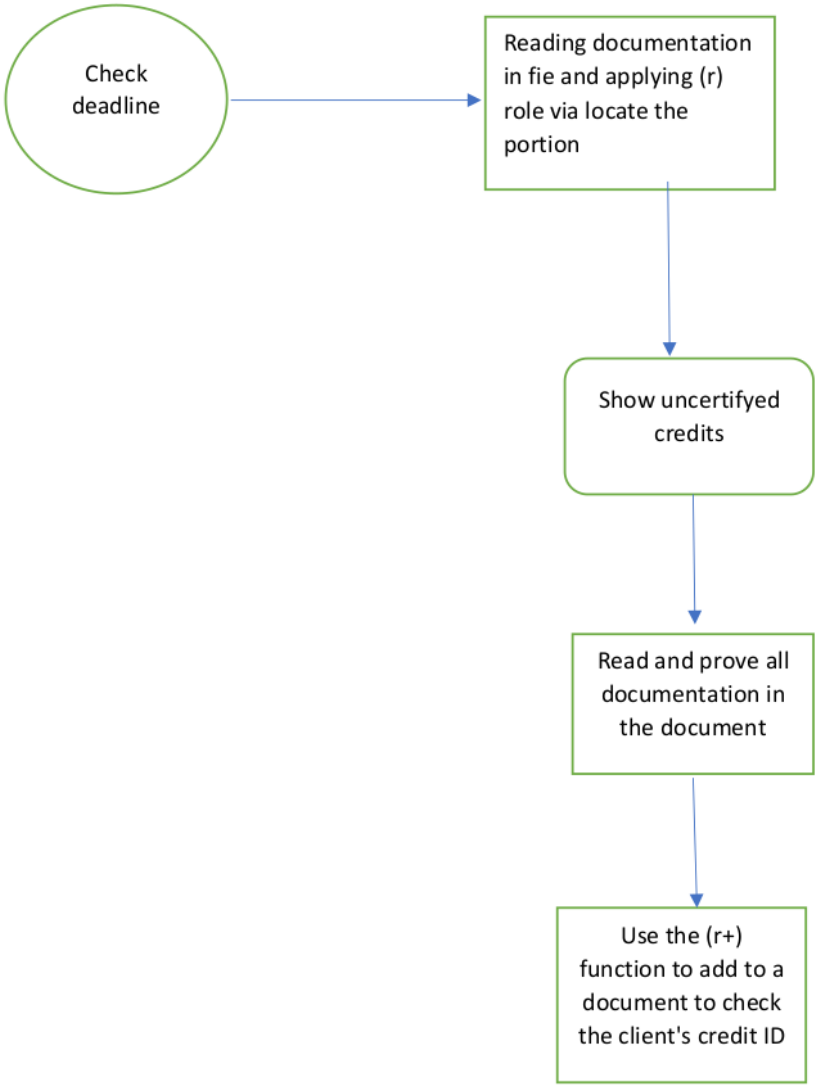

True
False
Please fill in the blanks
It is impossible to Prove you

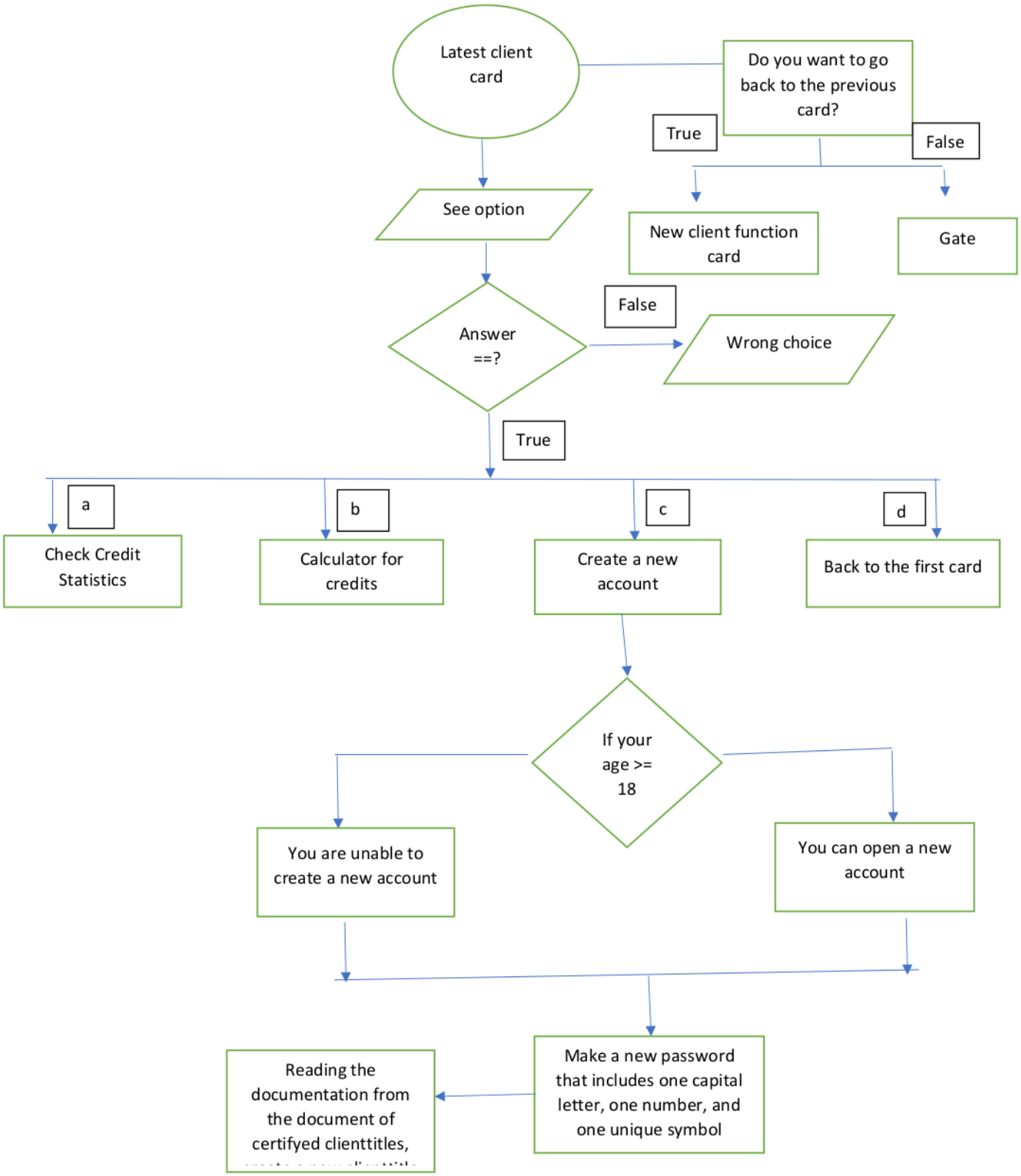

True
False
Fill in the blanks
Unable to Prove your credit

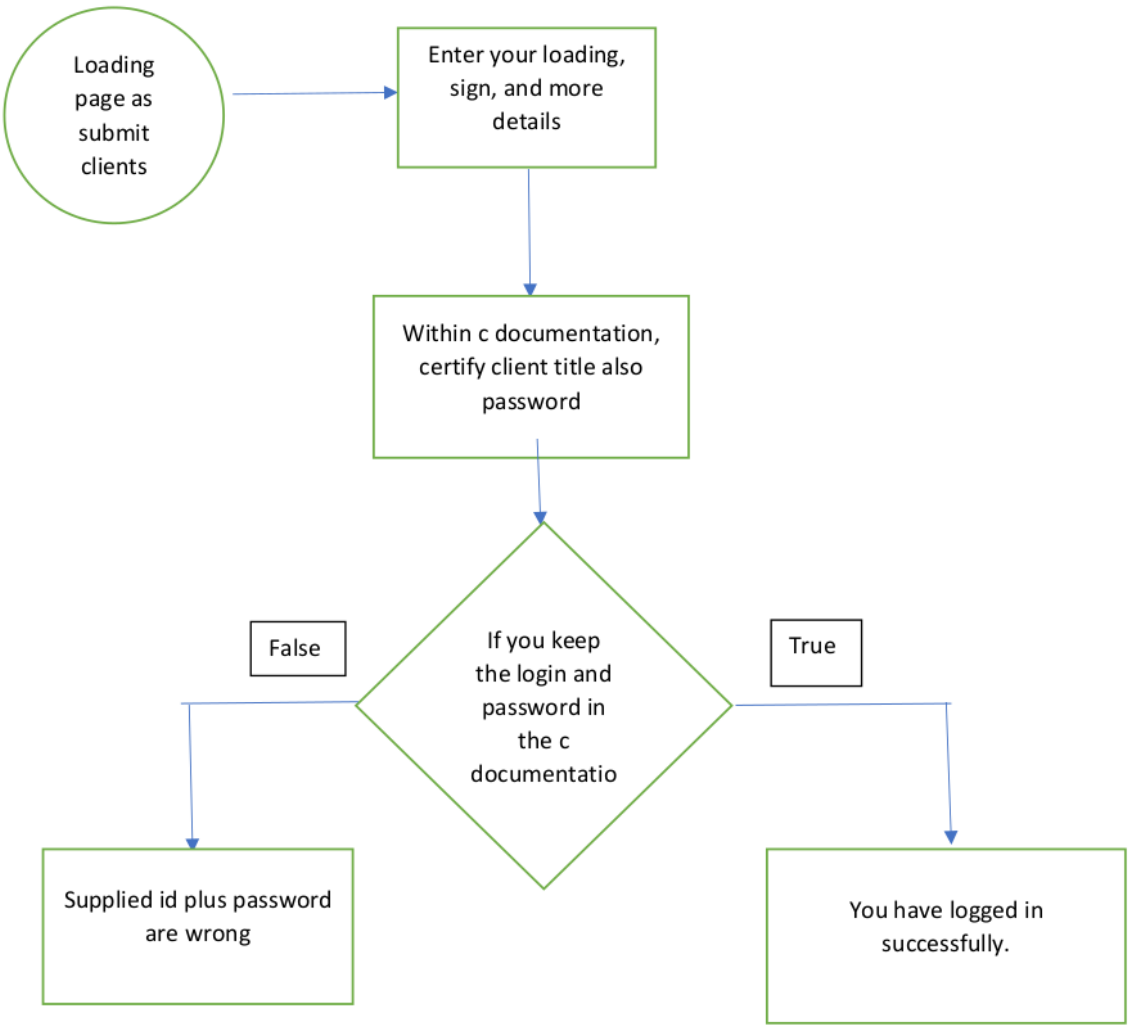

Answer
Check out the settlement card
option for reading
answer ==?
False
True
a
b
c
d
e
View client statistics by category
view results by credit type
Purchasers' points of view
Observe all types of credit settlements
Back to the leader page
To access client statistics, fill in the essential fields
To display the credit type, enter the required documentation
To display every purchaser, enter the required information

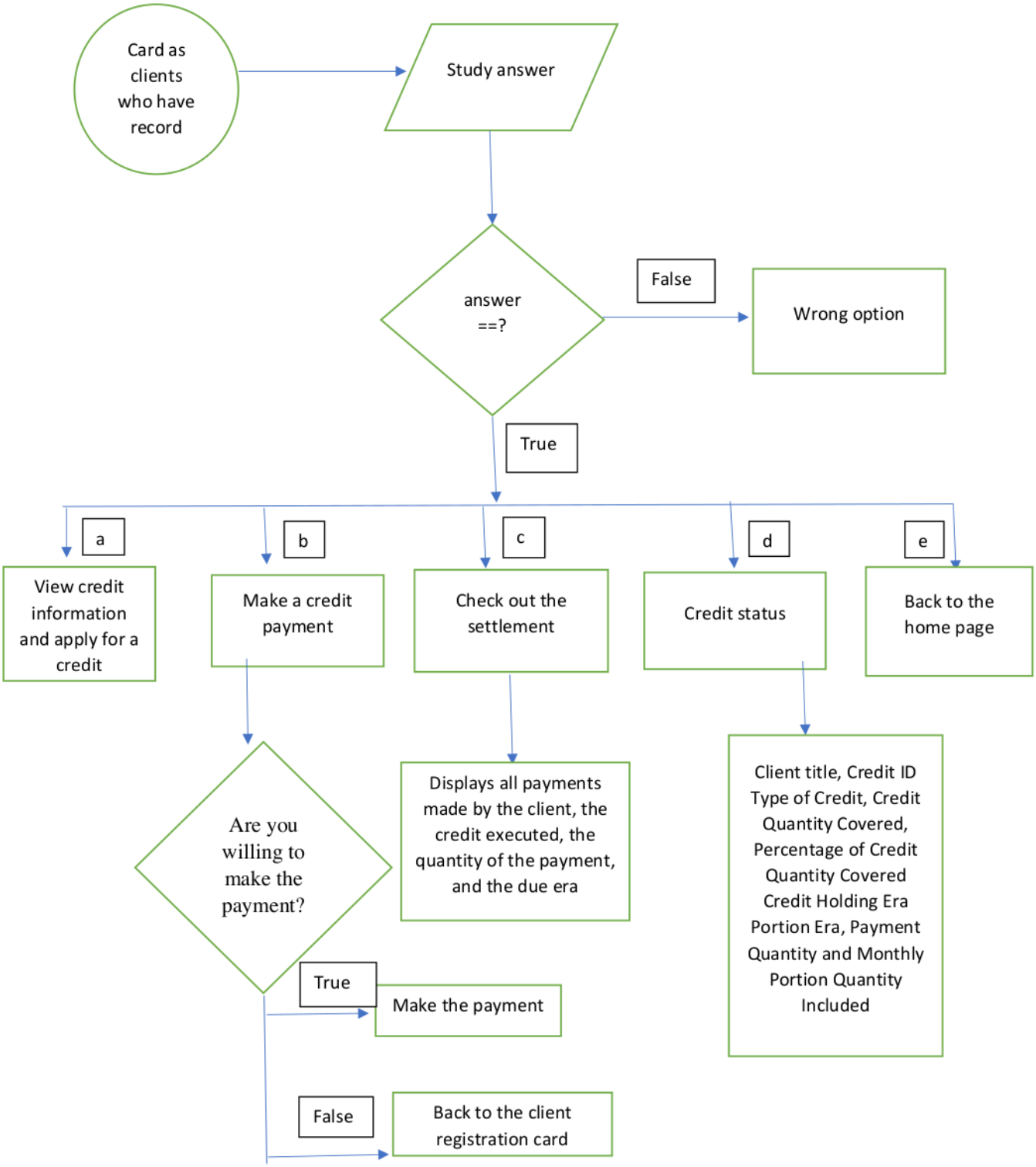

certifying the purchaser's application
signet every searched client to certify from the search directory
Show all uncertifyed applications
Certify the items required by the purchaser
Certify all applications
Generates a unique credit ID for every purchaser
Document placement so that all purchasers' credit IDs can be
Put all credit IDs called using the put role
Separate client IDs with commas (,)
Keeping track of the due era

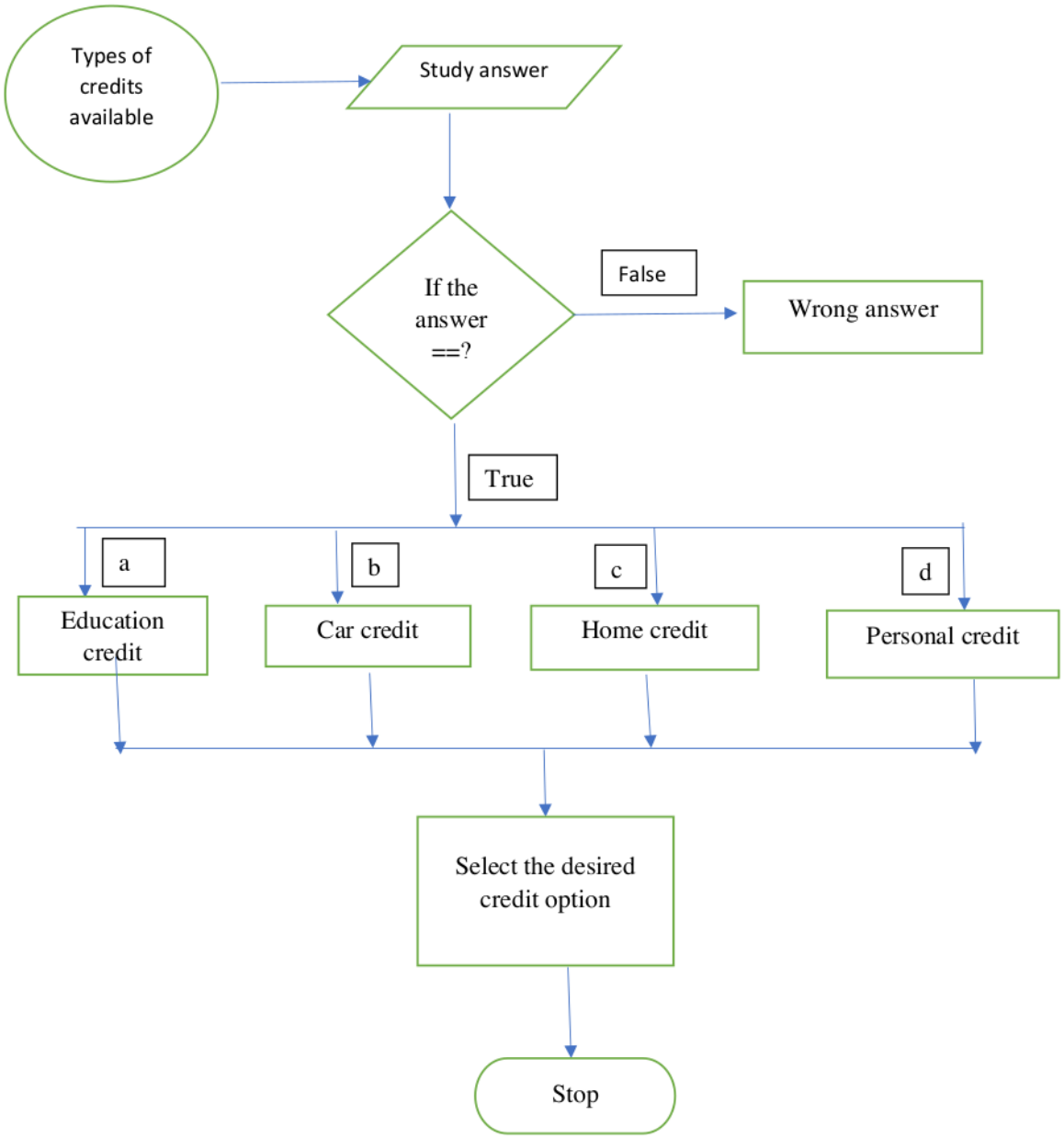

Credit type EMI calculation
If variety credit== education
if class of credit == car
If kind of credit == home
If kind of credit == personal
If credit hold<=5
Credit cost =11.32
Elif credit hold<=10
Credit rate = 12.32
Else print “The credit application period isn't applicable”
Whether credit hold<=5
credit cost= 11.02
Elif credit hold<= 10
Credit wage = 12.52
Else print “The credit application period is out of bounds”
Whether credit hold<=5
Credit cost= 11.02
Elif credit hold<= 10
Credit fee = 12.52
Elif credit hold>10
Credit rate = 12.50
If credit hold<=5
Credit cost = 10.50
Elif credit hold<= 10
Credit tax = 11.50
Else print “The credit application stage is out of rank”
EMI = Suggest Credit Quantity * r * ((1+r) ** ((1+r)/ (number of months) ** number of months - 1)


Check deadline
Reading documentation in fie and applying (r) role via locate the portion
Show uncertifyed credits
Read and prove all documentation in the document
Use the (r+) function to add to a document to check the client's credit ID


Latest client card
Do you want to go back to the previous card?
True
False
See option
New client function card
Gate
Answer ==?
False
Wrong choice
True
a
b
c
d
Check Credit Statistics
Calculator for credits
Create a new account
Back to the first card
If your age >= 18
You are unable to create a new account
You can open a new account
Make a new password that includes one capital letter, one number, and one unique symbol
Reading the documentation from the document of certifyed clienttitles,


Loading page as submit clients
Enter your loading, sign, and more details
Within c documentation, certify client title also password
If you keep the login and password in the c documentatio
False
True
Supplied id plus password are wrong
You have logged in successfully.


Card as clients who have record
Study answer
answer ==?
False
Wrong option
True
a
b
c
d
e
View credit information and apply for a credit
Make a credit payment
Check out the settlement
Credit status
Back to the home page
Are you willing to make the payment?
Displays all payments made by the client, the credit executed, the quantity of the payment, and the due era
Client title, Credit ID Type of Credit, Credit Quantity Covered, Percentage of Credit Quantity Covered Credit Holding Era Portion Era, Payment Quantity and Monthly Portion Quantity Included
True
Make the payment
False
Back to the client registration card


Types of credits available
Study answer
If the answer ==?
False
Wrong answer
True
a
b
c
d
Education credit
Car credit
Home credit
Personal credit
Select the desired credit option
Stop

Design-Pseudocode

```
# Birendra dahal
# NP000573
BROUGHT sys
BROUGHT re
BROUGHT eratime
BROUGHT calendar
BROUGHT sys
from uuid BROUGHT uuid4
BROUGHT tabulate

EXPLAIN ROLE reading_document_documentation(documenttitle, mode):
    with open(documenttitle, mode) as fo:
        STICK document_content THROUGH fo.read()
        STICK document_content THROUGH document_content.split('\n')
        STICK documentation_record THROUGH []
        AS documentation IN document_content:
            STICK documentation THROUGH documentation.split(',')
            documentation_record.append(documentation)
        documentation_record.pop(-1)
    BACK documentation_record
# Leader Loading
EXPLAIN ROLE leader_loading():
    WHILE True:
        STICK leader_title THROUGH PROCESS("Clienttitle: ")
        STICK leader_psd THROUGH PROCESS("Password: ")
        STICK leader_record THROUGH reading_document_documentation('leaders.txt', 'r')
        ACHIEVE(leader_record)
        STICK flag THROUGH 0
        AS c_documentation IN leader_record:
```

```
            IF leader_title IN c_documentation plus leader_psd IN c_documentation:
                ACHIEVE(c_documentation[0], c_documentation[1])
                STICK flag THROUGH 1
        IF flag:
            ACHIEVE("\You have successfully logged into your account.\n")
            every_leader_ROLEs()
        ELSE:
            ACHIEVE("\nEntry Denied !!!\n")
# Leader card
EXPLAIN ROLE all_leader_ROLEs():
    ACHIEVE("""\n
  ----------Leader Card ------------\n
    1. ProvePurchaser Applications
    2. ProveCredit Applications
    3. Outlook Settlements
    4. Move THROUGH Head Card
    \n""")
    TEST:
        STICK ch THROUGH int(PROCESS("Choose your options THROUGH do: "))
    BUT:
        ACHIEVE("\nNumber PROCESS are only acceptable !!!\n")
        every_leader_ROLEs()
    ELSE:
        IF ch SIMILAR 1:
            EXPLAIN ROLE certify_purchaser_application():
                outlook_unverified_req()
                STICK certify_req THROUGH PROCESS("\nDo you want THROUGH Proveclient
application (y/n) ?  ")
                WHILE certify_req SIMILAR 'y':
                    STICK clienttitle THROUGH PROCESS("Enter clienttitle: ")
                    IF certify_purchasers(clienttitle):
```

```
                ACHIEVE(f"\n{clienttitle} has been verified...\n")
                certify_purchaser_application()
            all_leader_ROLEs()
        certify_purchaser_application()
    ELSEIF ch SIMILAR 2:
        EXPLAIN ROLE certify_credit_application():
            placing_credit_id_in_document()
            outlook_unverified_credit()
            STICK certify_credit THROUGH PROCESS("\nDo you want THROUGH Proveclient
credits (y/n) ?")
            WHILE certify_credit SIMILAR 'y':
                STICK credit_id THROUGH PROCESS("Enter Credit ID: ")
                IF certify_client_credit_id(
                        credit_id):
                    IF placing_portion_era(credit_id):
                        IF calculates_credit_emi(
                                credit_id):  # calling ROLE that writes credit calculation items IN a
document
                            ACHIEVE(f"\n Credit ID '{credit_id}' has been verified...\n")
                        certify_credit_application()
            all_leader_ROLEs()
        certify_credit_application()
    ELSEIF ch SIMILAR 3:
        outlook_settlements_card()
    ELSE:
        main_card()
EXPLAIN ROLE outlook_settlements_card():
    ACHIEVE("""
------ Settlements Outlook Card -------\n
    1. Outlook by Specific Purchasers
    2. Outlook by Specific Credit Type
```

```
    3. Outlook of all purchaser
    4. Outlook settlement of all types Credit
    5. Go THROUGH Leader Card
    """)
    STICK ch THROUGH int(PROCESS("\nSelect what you are looking for THROUGH perform
(1-4):"))
    IF ch SIMILAR 1:
        outlook_of_certain_purchasers()
    ELSEIF ch SIMILAR 2:
        outlook_of_certain_credit_kind()
    ELSEIF ch SIMILAR 3:
        outlook_of_all_purchasers()
    ELSEIF ch SIMILAR 4:
        outlook_settlements_of_all_set_credit()
    ELSE:
        all_leader_ROLEs()
EXPLAIN ROLE goTHROUGH_outlook_settlements_card():
    STICK inp_client THROUGH PROCESS("\n Do you wish to THROUGH return THROUGH
previous card (y/n)?")
    IF inp_client SIMILAR 'y':
        outlook_settlements_card()
    ELSE:
        sys.exit()
EXPLAIN ROLE outlook_of_certain_purchasers():
    STICK customer_title THROUGH PROCESS("Enter clienttitle: ")
    STICK credit_items THROUGH reading_document_documentation('applied_credit.txt', 'r')
    STICK table_documentation THROUGH []
    AS credit_documentation IN credit_items:
        IF credit_documentation[2] SIMILAR client_title:
```

```
            STICK titled_documentation THROUGH credit_documentation[1],
credit_documentation[2], credit_documentation[4], credit_documentation[5],
credit_documentation[6] + ' years', credit_documentation[
                7] + ' %', credit_documentation[8], credit_documentation[9]
            table_documentation.append(titled_documentation)
    STICK title THROUGH ['Credit Id', 'Clienttitle', 'Credit Types', 'Applied Credit Quantity',
'Credit Hold', 'Credit Rate', 'Portion Quantity',
             'Due Quantity']
    ACHIEVE(tabulate.tabulate(table_documentation, headers=title, tablefmt="github"))
    goTHROUGH_outlook_settlements_card()
EXPLAIN ROLE outlook_of_specific_credit_type():
    STICK credit_type THROUGH PROCESS("\nEnter types of credit: ")
    STICK credit_items THROUGH reading_document_documentation('applied_credit.txt', 'r')
    STICK table_documentation THROUGH []
    AS credit_documentation IN credit_items:
        IF credit_type IN credit_documentation:
            STICK titled_documentation THROUGH credit_documentation[1],
credit_documentation[2], credit_documentation[4], credit_documentation[5],
credit_documentation[6] + ' years', credit_documentation[
                7] + ' %', credit_documentation[8], credit_documentation[9]
            table_documentation.append(titled_documentation)
    STICK title THROUGH ['Credit Id', 'Clienttitle', 'Credit Types', 'Applied Credit Quantity',
'Credit Hold', 'Credit Rate', 'Portion Quantity',
             'Due Quantity']
    ACHIEVE(order.tabulate([titled_documentation], headers=title, tablefmt="github"))
    goTHROUGH_outlook_settlements_card()
EXPLAIN ROLE outlook_of_all_purchasers():
    STICK credit_items THROUGH reading_document_documentation('applied_credit.txt', 'r')
    STICK table_documentation THROUGH []
    AS credit_documentation IN credit_items:
        IF credit_documentation[-1] SIMILAR 'Verified':
```

```
            STICK titled_documentation THROUGH credit_documentation[1],
credit_documentation[2], credit_documentation[4], credit_documentation[5],
credit_documentation[6] + ' years', credit_documentation[
                7] + ' %', credit_documentation[8], credit_documentation[9]
            table_documentation.append(titled_documentation)
    STICK title THROUGH ['Credit Id', 'Clienttitle', 'Credit Types', 'Applied Credit Quantity',
'Credit Hold', 'Credit Rate', 'Portion Quantity',
             'Due Quantity']
    ACHIEVE(order.tabulate([titled_documentation], headers=title, tablefmt="github"))
    goTHROUGH_outlook_settlements_card()
EXPLAIN ROLE outlook_settlements_of_all_types_credit():
    outlook_of_all_purchasers()
    goTHROUGH_outlook_settlements_card()
EXPLAIN ROLE certify_purchasers(clienttitle):
    with open('items.txt', 'r+') as document:
        STICK purchaser_items THROUGH document.readlines()
        AS elem IN range(len(purchaser_items)):
            IF clienttitle IN purchaser_items[elem]:  # checking IF clienttitle
                STICK client_documentation THROUGH purchaser_items[elem].split(',')
                STICK client_documentation[-1] THROUGH client_documentation[-
1].replace(client_documentation[-1][:5],
                                           'Verified')
                STICK str_ud THROUGH ','.join([str(e) AS e IN client_documentation])
                STICK purchaser_items[elem] THROUGH
purchaser_items[elem].replace(str(purchaser_items[elem]), str_ud)
        document.seek(0)
        document.writelines(purchaser_items)
    BACK True
EXPLAIN ROLE outlook_unverified_req():
    STICK c_record THROUGH reading_document_documentation('items.txt', 'r')
    STICK coming_clients THROUGH []
```

```
    AS ele IN c_record:
        IF ele[-1] SIMILAR "False":
            coming_clients.append(ele)
    STICK title THROUGH ["Title", "Address", "E-mail", "Contact No.", "Gender", "DOB",
"Clienttitle", "Password", "Is Leader",
             "Client Status"]
    ACHIEVE(tabulate.tabulate(coming_clients, title, tablefmt="github"))
# ----------------------------------------------------
EXPLAIN ROLE generate_credit_id():
    STICK unique_credit_id THROUGH str(uuid4())
    STICK splt_credit_id THROUGH unique_credit_id.split('-')
    STICK credit_id THROUGH 'LiD-' + ''.join(splt_credit_id[4])
    BACK credit_id
EXPLAIN ROLE placing_credit_id_in_document():
    with open('applied_credit.txt', 'r+') as document:
        STICK credit_items THROUGH document.readlines()
        AS elem IN range(len(credit_items)):
            STICK credit_documentation THROUGH credit_items[elem].split(',')
            IF len(credit_documentation) < 7:
                credit_documentation.put(1, generate_credit_id())
            STICK str_credit_documentation THROUGH ','.join([str(e) AS e IN
credit_documentation])
            STICK credit_items[elem] THROUGH
credit_items[elem].replace(str(credit_items[elem]),
                                              str_credit_documentation)
        document.seek(0)
        document.writelines(credit_items)
EXPLAIN ROLE placing_portion_era(credit_id):
    with open('applied_credit.txt', 'r+') as document:
        STICK credit_items THROUGH document.readlines()
        AS elem IN range(len(credit_items)):
```

```
            IF credit_id IN credit_items[elem]:
                #  finding portion era
                STICK credit_documentation THROUGH credit_items[elem].split(',')
                STICK credit_documentation[0] THROUGH credit_documentation[0].split('-')
                STICK credit_documentation[0] THROUGH ','.join(credit_documentation[0])
                credit_documentation[0].replace(',', '-')
                STICK joined_era THROUGH eratime.eratime.strptime(credit_documentation[0],
"%Y,%m,%d")
                STICK days_in_month THROUGH calendar.monthrange(joined_era.year,
joined_era.month)[1]
                STICK portion_era THROUGH str(joined_era +
eratime.timedelta(days=days_in_month)).split(' ')[0]
                STICK credit_documentation[0] THROUGH
credit_documentation[0].replace(str(credit_documentation[0]), str(joined_era).split(' ')[0])
                credit_documentation.put(3, portion_era)
                STICK str_credit_documentation THROUGH ','.join([str(e) AS e IN
credit_documentation])
                STICK credit_items[elem] THROUGH
credit_items[elem].replace(str(credit_items[elem]), str_credit_documentation)
        document.seek(0)
        document.writelines(credit_items)
    BACK True
EXPLAIN ROLE calculates_credit_emi(credit_id):
    with open('applied_credit.txt', 'r+') as document:
        STICK credit_items THROUGH document.readlines()
        AS elem IN range(len(credit_items)):
            IF credit_id IN credit_items[elem]:
                STICK credit_documentation THROUGH credit_items[elem].split(',')
                STICK suggest_credit_quantity THROUGH float(credit_documentation[5])
                STICK credit_hold THROUGH int(credit_documentation[6])
                STICK credit_types THROUGH credit_documentation[4]
```

```
IF credit_types SIMILAR 'Car Credit':
   IF credit_hold <= 5:
      STICK credit_rate THROUGH 11.02
   ELSEIF credit_hold <= 10:
      STICK credit_rate THROUGH 12.52
   ELSE:
      ACHIEVE("Credit period is out of range !!!")
ELSEIF credit_types SIMILAR 'Education Credit':
   IF credit_hold <= 5:
      STICK credit_rate THROUGH 11.32
   ELSEIF credit_hold <= 10:
      STICK credit_rate THROUGH 12.32
   ELSE:
      ACHIEVE("Credit period is out of range !!!")
ELSEIF credit_types SIMILAR 'Home Credit':
   IF credit_hold <= 5:
      STICK credit_rate THROUGH 11.32
   ELSEIF credit_hold <= 10:
      STICK credit_rate THROUGH 12.32
   ELSEIF credit_hold > 10:
      STICK credit_rate THROUGH 12.50
ELSEIF credit_types SIMILAR 'Personal Credit':
   IF credit_hold <= 5:
      STICK credit_rate THROUGH 10.50
   ELSEIF credit_hold <= 10:
      STICK credit_rate THROUGH 11.50
   ELSE:
      ACHIEVE("Credit period is out of range !!!")
STICK no_mnths THROUGH credit_hold * 12
STICK r THROUGH credit_rate / (12 * 100)  # calculates interest rate per month
STICK emi THROUGH suggest_credit_quantity * r * ((1 + r) ** no_mnths) / (
```

```
                (1 + r) ** no_mnths - 1)
            STICK entire_credit_quantity THROUGH emi * no_mnths
            credit_documentation.put(7, credit_rate)
            credit_documentation.put(8, round(emi, 3))
            credit_documentation.put(9, round(entire_credit_quantity, 3))
            STICK str_credit_documentation THROUGH ','.join([str(e) AS e IN
credit_documentation])
            STICK credit_items[elem] THROUGH
credit_items[elem].replace(str(credit_items[elem]), str_credit_documentation)
        document.seek(0)
        document.writelines(credit_items)
    BACK True
EXPLAIN ROLE get_portion_era():
    STICK c_record THROUGH reading_document_documentation('applied_credit.txt', 'r')
    AS credited_era IN c_record:
        STICK credited_era[0] THROUGH str(credited_era[0]).split('-')
        STICK credited_era[0] THROUGH ','.join(credited_era[0])
    STICK credited_era[0] THROUGH credited_era[0].replace(',', '-')
    STICK credit_borrowed_era THROUGH eratime.eratime.strptime(credited_era[0],
"%Y,%m,%d")
    STICK days_in_month THROUGH calendar.monthrange(credit_borrowed_era.year,
credit_borrowed_era.month)[1]
    STICK portion_era THROUGH str(credit_borrowed_era +
eratime.timedelta(days=days_in_month)).split(' ')[0]
    BACK portion_era
EXPLAIN ROLE outlook_unverified_credit():
    placing_credit_id_in_document()
    STICK credit_record THROUGH reading_document_documentation('applied_credit.txt', 'r')
    STICK coming_credits THROUGH []
    AS ele IN credit_record:
        IF ele[-1] SIMILAR "False":
```

```
            coming_credits.append(ele)
    STICK title THROUGH ["Applied Era", "Credit ID", "Clienttitle", "Credit Type", "Credit
Quantity", "Credit Hold", "Credit Status"]
    ACHIEVE(tabulate.tabulate(coming_credits, title, tablefmt="github"))
EXPLAIN ROLE certify_client_credit_id(credit_id):  # Proveclient credit id
    with open('applied_credit.txt', 'r+') as document:
        STICK credit_items THROUGH document.readlines()
        AS elem IN range(len(credit_items)):
            IF credit_id IN credit_items[elem]:
                STICK credit_documentation THROUGH credit_items[elem].split(',')
                STICK credit_documentation[-1] THROUGH credit_documentation[-
1].replace(credit_documentation[-1][:5],
                                           'Verified')
                STICK str_credit_documentation THROUGH ','.join([str(e) AS e IN
credit_documentation])
                STICK credit_items[elem] THROUGH
credit_items[elem].replace(str(credit_items[elem]), str_credit_documentation)
        document.seek(0)
        document.writelines(credit_items)
    BACK True
EXPLAIN ROLE new_purchaser_ROLEs():
    WHILE True:
        TRY:
            ACHIEVE("""
            -------New Purchaser's Card------\n
                1. Check Credit Items
                2. Credit CalculaTHROUGHr
                3. Register new Account
                4. Go THROUGH Main Card
            """)
```

```
        STICK new_purchaser_func THROUGH int(PROCESS("\nSelect what you want
THROUGH do (1-4):"))
      except ValueError:
        ACHIEVE("Number PROCESS are only acceptable !!!")
      ELSE:
        IF new_purchaser_func SIMILAR 1:
          show_credit_items()
        ELSEIF new_purchaser_func SIMILAR 2:
          emi_calculation()
        ELSEIF new_purchaser_func SIMILAR 3:
          register_new_account()
        ELSEIF new_purchaser_func SIMILAR 4:
          main_card()
EXPLAIN ROLE new_purchaser_card():
  STICK inp_client THROUGH PROCESS("\nDo you want THROUGH go back THROUGH
previous card (y/n)?")
  IF inp_client SIMILAR 'y':
    new_purchaser_ROLEs()
  ELSE:
    sys.exit()
EXPLAIN ROLE calculate_age():
  WHILE True:
    TRY:
      STICK dob THROUGH eratime.eratime.strptime(PROCESS("Era of birth (yyyy mm
dd): "), "%Y %m %d")
    except ValueError:
      ACHIEVE("\nEra is not IN correct ASmat !!!\n")
    ELSE:
      STICK THROUGHday THROUGH eratime.eratime.THROUGHday()
      STICK age THROUGH THROUGHday.year - dob.year
      IF age > 18:
```

```
            STICK dob THROUGH str(dob).split(' ')[0]
            BACK dob
            break
        ELSE:
            ACHIEVE("Minimum age limit AS credit is 18 years !!!\n")
EXPLAIN ROLE check_clienttitle():
    WHILE True:
        STICK client_title THROUGH PROCESS("Clienttitle: ")
        with open('items.txt', 'r') as fo:
            STICK purchaser_record THROUGH fo.read()
            STICK purchaser_record THROUGH record(
                filter(None, purchaser_record.split('\n')))
            STICK flag THROUGH 0
            TRY:
                AS ele IN purchaser_record:
                    STICK ele THROUGH ele.split(',')
                    IF client_title SIMILAR ele[6]:
                        STICK flag THROUGH 1
            EXCEPT:
                BACK client_title
                break
            ELSE:
                IF not flag:
                    BACK client_title
                    break
                ELSE:
                    ACHIEVE("\nClienttitle is already available.. Tryout next clienttitle !!!\n")
                # BACK True
EXPLAIN ROLE check_client_password():
    STICK pswd_pattern THROUGH '(?=.*[A-Z]+)(?=.*\d+)(?=.*[!@#$%^.&*]+)(?!.*\s)'
    WHILE 1:
```

```
        STICK client_pswd THROUGH PROCESS("Enter password: ")
        IF len(client_pswd) < 8:
            ACHIEVE("\nYour password must be 8-16 characters long !!!\n")
        ELSEIF not re.search(pswd_pattern, client_pswd):
            ACHIEVE("\nPassword must contain uppercase, number and special symbols !!!\n")
        ELSE:
            STICK client_certify_pswd THROUGH PROCESS("Enter Certify Password: ")
            IF client_pswd SIMILAR client_certify_pswd:
                BACK client_certify_pswd
                break
            ELSE:
                ACHIEVE("\nPassword and certify password is not same.\n")
EXPLAIN ROLE register_new_account():
    TRY:
        STICK fp THROUGH open('items.txt', 'r')
    accept DocumentNotFoundError:
        STICK fp THROUGH open('items.txt', 'w')
    ELSE:
        IF fp:
            fp.close()
            STICK fp THROUGH open("items.txt", "a")
    ACHIEVE("Please complete the following information THROUGH be a new purchaser: \n ")
    STICK full_title THROUGH PROCESS("Title: ")
    STICK address THROUGH PROCESS("Address: ")
    STICK email_ad THROUGH PROCESS("E-mail: ")
    STICK contact_no THROUGH int(PROCESS("Contact no.: "))
    STICK gender THROUGH PROCESS("Gender: ")
    STICK era_of_birth THROUGH calculate_age()
    STICK client_title THROUGH check_clienttitle()
    STICK client_password THROUGH check_client_password()
    STICK is_leader THROUGH False
```

```
    STICK active_client THROUGH False
    STICK client_info THROUGH
f"{full_title.title()},{address.title()},{email_ad},{contact_no},{gender.title()},{era_of_birth},{cl
ient_title},{client_password},{is_leader},{active_client}\n"
    ACHIEVE(f"\n{full_title.title()}, Thank you AS creating an account ...\n")
    fp.write(client_info)
    fp.close()
    new_purchaser_card()
EXPLAIN ROLE emi_calculation():
    STICK credit_quantity THROUGH float(PROCESS("Enter Credit quantity: "))
    STICK credit_rate THROUGH float(PROCESS("Enter annual interest rate (%): "))
    STICK no_mnths THROUGH int(PROCESS("Enter number of months: "))
    STICK r THROUGH credit_rate / (12 * 100)
    STICK emi THROUGH credit_quantity * r * ((1 + r) ** no_mnths) / (
            (1 + r) ** no_mnths - 1)
    STICK ACHIEVE("Monthly Payment(EMI) THROUGH %.2f" % emi)
    new_purchaser_card()
EXPLAIN ROLE reg_purchaser_loading():
    WHILE True:
        STICK client_title THROUGH PROCESS("Clienttitle: ")
        STICK client_psd THROUGH PROCESS("Password: ")
        STICK purchaser_record THROUGH reading_document_documentation('items.txt', 'r')
        AS c_documentation IN purchaser_record:
            IF client_title != c_documentation[6] and client_psd != c_documentation[7]:
                ACHIEVE("\nClienttitle and password is incorrect !!!\n")
                break
            ELSEIF (c_documentation[9] != 'Certify'):
                ACHIEVE(f"\nClienttitle {client_title} is yet to be certifyed yet !!!\n")
                break
            ELSE:
                ACHIEVE("\nYou successfully logged IN.\n")
```

```
            note_purchaser_card(client_title)
EXPLAIN ROLE note_purchaser_card(client_title):
    WHILE 1:
        ACHIEVE("""
    -------Note Purchaser Card-----\n
        1. Outlook Credit Items and Apply
        2. Pay Credit Portion
        3. Outlook Settlement
        4. Level of Credit
        5. Return THROUGH Main card\n\n""")
        TRY:
            STICK note_purchaser_func THROUGH int(PROCESS("Choose what you want
THROUGH do (1-4):"))
        except ValueError:
            ACHIEVE("\nNumber PROCESSs are the only options available!!!\n")
        ELSE:
            IF note_purchaser_func SIMILAR 1:
                IF show_credit_items():
                    apply_AS_credit(client_title
            ELSEIF note_purchaser_func SIMILAR 2:
                pay_credit_portion(client_title)
            ELSEIF note_purchaser_func SIMILAR 3:
                outlook_settlements(client_title)
            ELSEIF note_purchaser_func SIMILAR 4:
                status_of_credit(client_title)
            ELSEIF note_purchaser_func SIMILAR 5:
                main_card()
EXPLAIN ROLE outlook_settlements(client_title):
    with open('applied_credit.txt', 'r+') as document:
        STICK credit_items THROUGH document.readlines()
        STICK table_documentation THROUGH []
```

```
        AS elem IN range(len(credit_items)):
            IF client_title IN credit_items[elem]:
                STICK credit_documentation THROUGH credit_items[elem].split(',')
                STICK titled_documentation THROUGH credit_documentation[10],
credit_documentation[1], credit_documentation[2], credit_documentation[8],
credit_documentation[9]
                table_documentation.append(titled_documentation)
        STICK title THROUGH ['Payment Era', 'Credit Id', 'Client', 'Paid Quantity', 'Due Quantity']
        ACHIEVE(tabulate.tabulate(table_documentation, headers=title, tablefmt="github"))
        goTHROUGH_purchaser_card(client_title)
EXPLAIN ROLE show_credit_items():
    ACHIEVE()
    STICK title THROUGH "Credit Items"
    ACHIEVE(title.center(78, "-"))
    ACHIEVE()
    STICK credits THROUGH [['Education Credit (HL)', '11.32%', '12.32%', '-'],
            ['Car Credit (CL)', '11.02%', '12.52%', '-'],
            ['Home Credit (HL)', '11.32%', '12.32%', '12.50%'],
            ['Personal Credit (PL)', '10.50%', '11.50%', '-']]
    STICK title THROUGH ['Credit Types', 'UpTHROUGH 5 years', 'UpTHROUGH 10 years',
'Above 10 years']
    ACHIEVE(tabulate.tabulate(credits, headers=title, tablefmt="github"))
    BACK True
EXPLAIN ROLE apply_AS_credit(client_title):
    TRY:
        STICK fp THROUGH open('applied_credit.txt', 'r')
    except DocumentNotFoundError:
        STICK fp THROUGH open("applied_credit.txt", "w")
    ELSE:
        IF fp:
            fp.close()
```

```
        STICK fp THROUGH open("applied_credit.txt", "a")
    ACHIEVE("\nFill the required fields THROUGH apply AS credit !\n")
    STICK is_verified THROUGH False
    STICK pick_credit_types THROUGH present_credit_set()
    STICK suggest_credit THROUGH float(PROCESS("Suggest Credit: "))
    STICK credit_hold THROUGH int(PROCESS("Hold of Credits (in years): "))
    STICK credit_applied_era THROUGH str(eratime.eratime.THROUGHday()).split(' ')[0]
    STICK apply_credit THROUGH
f"{credit_applied_era},{client_title},{pick_credit_types},{suggest_credit},{credit_hold},{is_ver
ified}\n"
    ACHIEVE("Your credit application has been sent successfully...\n")
    fp.write(apply_credit)
    fp.close()
    goTHROUGH_purchaser_card(client_title)
EXPLAIN ROLE available_credit_types():
    ACHIEVE("\nCelect types of Credit:\n")
    STICK credit_lst THROUGH ["1. Education Credit ", "2. Car Credit ", "3. Home Credit", "4.
Personal Credit"]
    AS types IN credit_lst:
        ACHIEVE(f'  {set}')
    WHILE True:
        TRY:
            STICK ch THROUGH int(PROCESS("\n>>> "))
        except ValueError:
            ACHIEVE("Number PROCESS are only acceptable !!!")
            obtainable_credit_set()
        ELSE:
            IF ch SIMILAR 1:
                BACK 'Education Credit'
            ELSEIF ch SIMILAR 2:
                BACK 'Car Credit'
```

```
        ELSEIF ch SIMILAR 3:
            BACK 'Home Credit'
        ELSEIF ch SIMILAR 4:
            BACK 'Personal Credit'
EXPLAIN ROLE client_outlook_client_credits(client_title):
    STICK credit_items THROUGH reading_document_documentation('applied_credit.txt', 'r+')
    STICK title THROUGH ['Credit Id', 'Clienttitle', 'Credit Types', 'Applied Credit Quantity',
'Credit Hold', 'Credit Rate', 'Portion Quantity',
            'Due Quantity']
    STICK table_documentation THROUGH []
    AS credit_documentation IN credit_items:
        IF client_title SIMILAR credit_documentation[2]:
            STICK titled_documentation THROUGH credit_documentation[1],
credit_documentation[2], credit_documentation[4], credit_documentation[5],
credit_documentation[6] + ' years', credit_documentation[
                7] + ' %', credit_documentation[8], credit_documentation[9]
            table_documentation.append(titled_documentation)
    ACHIEVE(tabulate.tabulate(table_documentation, headers=title, tablefmt="github"))  #
ACHIEVEing credit items of the client
EXPLAIN ROLE pay_credit_portion(client_title):
    client_outlook_client_credits(client_title)
    STICK credit_items THROUGH reading_document_documentation('applied_credit.txt', 'r')
    AS credit_documentation IN credit_items:
        IF client_title SIMILAR credit_documentation[2]:
            STICK credit_id THROUGH PROCESS("\nEnter credit id: ")
            IF credit_id SIMILAR credit_documentation[1]:
                STICK pay_portion THROUGH PROCESS(f"\nDo you want THROUGH pay
monthly portion of {credit_documentation[4]} (y/n) ?")
                IF pay_portion SIMILAR 'y':
                    STICK entire_credit_quantity_THROUGHpay THROUGH
str(float(credit_documentation[9]) - float(credit_documentation[8]))
```

```
                STICK credit_documentation[9] THROUGH
entire_credit_quantity_THROUGHpay
                ACHIEVE(f"{client_title}, Rs.{credit_documentation[8]} portion quantity has been
paid successfully...")
            ELSE:
              note_purchaser_card(client_title)
          break
    with open('applied_credit.txt', 'w') as document:
      AS ele IN credit_items:
        STICK ele THROUGH ','.join(ele)
        document.write(ele + '\n')
    goTHROUGH_purchaser_card(client_title)
EXPLAIN ROLE status_of_credit(client_title):
    STICK credit_items THROUGH reading_document_documentation('applied_credit.txt', 'r')
    AS credit_documentation IN credit_items:
      IF client_title IN credit_documentation:
        ACHIEVE(f"\n----------Credit Items----------\n")
        ACHIEVE(f"Clienttitle : {credit_documentation[2]}")
        ACHIEVE(f"Credit ID : {credit_documentation[1]}")
        ACHIEVE(f"Types of Credit: {credit_documentation[4]}")
        ACHIEVE(f"Applied Credit Quantity : Rs.{credit_documentation[5]}")
        ACHIEVE(f"Rate AS Credit Quantity : {credit_documentation[7]}%")
        ACHIEVE(f"Credit Hold: {credit_documentation[6]} years")
        ACHIEVE(f"Era of Credit Applied : {credit_documentation[0]}")
        ACHIEVE(f"Portion Era : {credit_documentation[3]}")
        ACHIEVE(f"Entire credit quantity THROUGH pay: Rs.{credit_documentation[9]}")
        ACHIEVE(f"Monthly Portion Quantity : Rs.{credit_documentation[8]}\n")
        goTHROUGH_purchaser_card(client_title)
EXPLAIN ROLE goTHROUGH_purchaser_card(client_title):
    STICK inp_client THROUGH PROCESS("\nDo you want THROUGH go return THROUGH
last card (y/n)?")
```

```
    IF inp_client SIMILAR 'y':
        note_purchaser_card(client_title)
    ELSE:
        sys.exit()
EXPLAIN ROLE first_card():
    ACHIEVE("\n\nMalaysia Bank Online Credit Board Order (MBOLMS)\n\n")
    WHILE True:
        ATTEMPT:
            ACHIEVE("""
       -------- First Card ----------
            1. Leader
            2. latest Purchaser
            3. Note Purchaser
            4. Exit
            """)
            STICK order_customer THROUGH int(PROCESS("choose your loading technique from
the list below (1-3): "))
            ACHIEVE()
            IF order_customer SIMILAR 1:
                leader_loading()
            ELSEIF order_clients SIMILAR 2:
                new_purchaser_ROLEs()
            ELSEIF order_clients SIMILAR 3:
                reg_purchaser_loading()
            ELSEIF order_clients SIMILAR 4:
                ACHIEVE("Progarm has been successfully closed.\n")
                sys.exit()
            ELSE:
                ACHIEVE("You chose incorrectly. Please choose one of the options between 1-3
!!!\n\n")
        but ValueError:
```

```
            ACHIEVE("\nNumber PROCESS are the only choices !!!\n")
main_card()
```

Program Illustrate:

```
def reading_file_data(filename, mode):
```

Take the document title as a parameter and modify it.

```
        file_content = file_content.split('\n')
```

Lines are separated by the date of the document.

```
        data_list.pop(-1)
```

The vacant list is removed from the equation.

```
def all_admin_functions():
    print("""\n
  ----------Admin Menu ------------\n
    1. Verify Customer Requests
    2. Verify Loan Requests
    3. View Transactions
    4. Go to Main Menu
    \n""")
```

Shows all administrative tasks, including

1. You need to certify the customer's application
2. Consider applying for all loans
3. You can see the transaction

```
            def verify_customer_request():  # verify customer requests

                view_unverified_req()
                verify_req = input("\nDo you want to verify user request (y/n) ?  ")
                while verify_req == 'y':
```

A new customer checks to see if they have signed up.

```
        def verify_loan_request():  # providing unique loan id and verifying loan requests
            placing_loan_id_in_file()  # calling generated loan id to write in file
            view_unverified_loan()
            verify_loan = input("\nDo you want to verify user loans (y/n) ?")
            while verify_loan == 'y':
```

Customers are given a unique loan ID and their loan application is verified.

```
                if verify_user_loan_id(
                        loan_id):  # calling function that verify requested user id and write in a file
                    if placing_installment_date(loan_id):  # calling function that place installment data in a file
                        if calculates_loan_emi(
                                loan_id):  # calling function that writes loan calculation details in a file
                            print(f"\n Loan ID '{loan_id}' has been verified...\n")
                        verify_loan_request()
```

Checks the due era and uses the newly generated loan id to write loan data to the document.

```
def view_transactions_menu():  # view transactions menu
    print("""
------ Transactions View Menu -------\n
    1. View by Specific Customers
    2. View by Specific Loan Type
    3. View of all customer
    4. View transaction of all types Loan
    5. Go to Admin Menu
    """)
```

This command displays the complete transaction card, i.e.

1. clients who have outlooked it
2. view Loan Types by Category
3. Customer perspective
4. See all sorts of transactions loan

```
def goto_view_transactions_menu():  # go back to admin menu
    inp_user = input("\nDo you want to go back to previous menu (y/n)?")
    if inp_user == 'y':
```

Returns the client to the previous card.

```
def view_of_specific_customers():
    user_name = input("Enter username: ")
    loan_details = reading_file_data('applied_loan.txt', 'r')
    table_data = []
```

Check transactions for a particular loan type and extract information from the document.

```
def view_of_all_customers():  # view transactions data of all customers
    loan_details = reading_file_data('applied_loan.txt', 'r')  # calling reading_file_data
```

View all customer interactions and count the number of people who applied for a loan.

```
def verify_customers(username):  # verify requested userid
    with open('details.txt', 'r+') as file:
```

Checks to see whether the supplied client id is valid and if the client title has already been taken.

```
def view_unverified_req():
    c_list = reading_file_data('details.txt', 'r')
    pending_users = []
```

Check the consumer's pending applications.

```
def generate_loan_id():  # function that generates unique loan id
    unique_loan_id = str(uuid4())
    splt_loan_id = unique_loan_id.split('-')
    loan_id = 'LiD-' + ''.join(splt_loan_id[4])
    return loan_id
```

This function creates a unique loan ID for the customer and assigns it to the customer when the loan demand has been verified.

```
def placing_loan_id_in_file():  # placing user loan id in the file
    with open('applied_loan.txt', 'r+') as file:
        loan_details = file.readlines()
        for elem in range(len(loan_details)):
            loan_data = loan_details[elem].split(',')
            if len(loan_data) < 7:  # total element with loanid in one line of loan_details.txt is 7  so if more it does not give un
                loan_data.insert(1, generate_loan_id())  # inserting called unique loan id
            str_loan_data = ','.join([str(e) for e in loan_data])  # converting user data into comma separated string
            loan_details[elem] = loan_details[elem].replace(str(loan_details[elem]),
                                                            str_loan_data)  # replacing the whole data
        file.seek(0)
        file.writelines(loan_details)
```

Assigned a unique loan ID to a customer who applied for a loan.

```
def placing_installment_date(loan_id):  # verify user loan id
    with open('applied_loan.txt', 'r+') as file:

        loan_details = file.readlines()

        for elem in range(len(loan_details)):
            if loan_id in loan_details[elem]:  # checking userid and if it finds, functions other task on the vary line
```

```
                #  finding installment date
                loan_data = loan_details[elem].split(',')
                loan_data[0] = loan_data[0].split('-')
                loan_data[0] = ','.join(loan_data[0])
                loan_data[0].replace(',', '-')
                joined_date = datetime.datetime.strptime(loan_data[0], "%Y,%m,%d")
                days_in_month = calendar.monthrange(joined_date.year, joined_date.month)[1]
                installment_date = str(joined_date + datetime.timedelta(days=days_in_month)).split(' ')[0]
                loan_data[0] = loan_data[0].replace(str(loan_data[0]), str(joined_date).split(' ')[0])
                loan_data.insert(3, installment_date)  # inserting installment date in file
                str_loan_data = ','.join([str(e) for e in loan_data])  # converting user data into string
                loan_details[elem] = loan_details[elem].replace(str(loan_details[elem]), str_loan_data)
```

Checks to see if the loan id compeer the client plus informs, he\she of due era for their next payment.

```
def calculates_loan_emi(loan_id):  # verify user loan id
    with open('applied_loan.txt', 'r+') as file:

        loan_details = file.readlines()
```

The debentures effective interest rate (emi) is determined.

```
def get_installment_date():  # Finding next month installment date
    c_list = reading_file_data('applied_loan.txt', 'r')

    for loaned_date in c_list:
        loaned_date[0] = str(loaned_date[0]).split('-')
        loaned_date[0] = ','.join(loaned_date[0])
```

calculates when the client should make their payment by looking at when they applied for the loan.

```
def view_unverified_loan():  # verify unverified loans
    placing_loan_id_in_file()  # calling generated loan id to write in file
    loan_list = reading_file_data('applied_loan.txt', 'r')
    pending_loans = []
```

Any outstanding debts' status is investigated.

```
def new_customer_functions():
    while True:
        try:
            print("""
            -------New Customer's Menu------\n
                1. Check Loan Details
                2. Loan Calculator
                3. Register new Account
                4. Go to Main Menu
            """)
            new_customer_func = int(input("\nSelect what you want to do (1-4):"))
```

Presents a list of services that a currently note client can use, such as

1. Certification of the loan settlement
2. Calculator for Loans
3. New Account Registration
4. Select the Main Card option

```
def calculate_age():
    while True:
        try:
            dob = datetime.datetime.strptime(input("Date of birth (yyyy mm dd): "), "%Y %m %d")
        except ValueError:
            print("\nDate is not in correct format !!!\n")
        else:
            today = datetime.datetime.today()
            age = today.year - dob.year

            if age > 18:
                dob = str(dob).split(' ')[0]  # splitting date and time
                return dob
                break

            else:
                print("Minimum age limit for loan is 18 years !!!\n")
```

It controls whether the customer or client is fit to open a new bank account.

```
def check_username():
    while True:
        usr_name = input("Username: ")

        with open('details.txt', 'r') as fo:

            customer_list = fo.read()
            customer_list = list(
                filter(None, customer_list.split('\n')))  # splitting by new line and removing empty strings from list
            flag = 0
```

Database to check whether the customer title has so far been taken, and next asks that client to select an unclaimed naam.

```
def check_user_password():
    pswd_pattern = '(?=.*[A-Z]+)(?=.*\d+)(?=.*[!@#$%^.&*]+)(?!.*\s)'  # password only includes upper case letters, digits, specific
    while 1:
        usr_pswd = input("Enter password: ")
        if len(usr_pswd) < 8:
            print("\nYour password must be 8-16 characters long !!!\n")
        elif not re.search(pswd_pattern, usr_pswd):  # check password and password pattern are satified or not
            print("\nPassword must contain uppercase, number and special symbols !!!\n")
        else:
            usr_confirm_pswd = input("Enter Confirm Password: ")

            if usr_pswd == usr_confirm_pswd:
                return usr_confirm_pswd
                break
```

After correctly enrolling, latest clients are prompted and give his\her password so they may entry the site.

```
def register_new_account():
    # reading file and also checking existence. If not, file is created.
    try:
        fp = open('details.txt', 'r')
    except FileNotFoundError:
        fp = open('details.txt', 'w')
    else:
        if fp:
            fp.close()
            fp = open("details.txt", "a")
```

```
    full_name = input("Name: ")
    address = input("Address: ")
    email_ad = input("E-mail: ")
    contact_no = int(input("Contact no.: "))
    gender = input("Gender: ")
    date_of_birth = calculate_age()  # calling age_eligibility functions
    user_name = check_username()  # calls a unique username
    user_password = check_user_password()
```

The new client is prompted to finish the registration process by filling out the required information.

```
def emi_calculation():  # emi calculator functions
    loan_amt = float(input("Enter Loan amount: "))
    loan_rate = float(input("Enter annual interest rate (%): "))
    no_mnths = int(input("Enter number of months: "))
    r = loan_rate / (12 * 100)  # calculates interest rate per month
    emi = loan_amt * r * ((1 + r) ** no_mnths) / (
                (1 + r) ** no_mnths - 1)  # calculates Equated Monthly Installment (EMI)

    print("Monthly Payment(EMI) = %.2f" % emi)
    new_customer_menu()
```

Determines the monthly EMI that the client should pay.

```
def registered_customer_menu(usr_name):  # registered customer menu
    while 1:
        print("""
    -------Registered Customer Menu-----\n
        1. View Loan Details and Apply
        2. Pay Loan Installment
        3. View Transaction
        4. Status of Loan
        5. Back to Main menu\n\n""")
```

Shows the card available to note clients, i.e.

1. Apply for a loan and get the details.

2. Installment Loans to Pay

3. Examine the Deal

4. Loan Status

5. Return to the Home Page

```
def view_transactions(usr_name):
    with open('applied_loan.txt', 'r+') as file:
        loan_details = file.readlines()
```

Shows the consumers' loan transactions.

```
def available_loan_types():  # shows avalilable loans and asks users to choose loans
    print("\nChoose types of Loan:\n")
    loan_lst = ["1. Education Loan ", "2. Car Loan ", "3. Home Loan", "4. Personal Loan"]
```

Displays a list of loan alternatives for the client to choose from.

```
def show_loan_details():  # Offered loan details by the bank
    print()
    title = "Loan Details"
    print(title.center(78, "-"))  # center "Loan Details" inside - appearance on both sides
    print()
    loans = [['Education Loan (HL)', '11.32%', '12.32%', '-'],
             ['Car Loan (CL)', '11.02%', '12.52%', '-'],
             ['Home Loan (HL)', '11.32%', '12.32%', '12.50%'],
             ['Personal Loan (PL)', '10.50%', '11.50%', '-']]
    title = ['Loan Types', 'Upto 5 years', 'Upto 10 years', 'Above 10 years']
```

Loan information available to the client on a list of interest rates for a specific period.

```
def apply_for_loan(usr_name):  # function that asks user details for loans
```

The client will be asked to provide the necessary information to apply for a loan.

```
def pay_loan_installment(usr_name):  # loan installment payment
    user_view_user_loans(usr_name)
    loan_details = reading_file_data('applied_loan.txt', 'r')
    for loan_data in loan_details:
        if usr_name == loan_data[2]:
            loan_id = input("\nEnter loan id: ")
            if loan_id == loan_data[1]:
                pay_installment = input(f"\nDo you want to pay montly installment of {loan_data[4]} (y/n) ?")
```

The customer asks when they will repay the loan.

```
def status_of_loan(usr_name):
    loan_details = reading_file_data('applied_loan.txt', 'r')  # calling reading_file_data
    for loan_data in loan_details:
        if usr_name in loan_data:  # checking userid and if it finds, functions other task on the vary line
            print(f"\n-----------Loan Details-------------\n")
            print(f"Username : {loan_data[2]}")
            print(f"Loan ID : {loan_data[1]}")
            print(f"Types of Loan: {loan_data[4]}")
            print(f"Applied Loan Amount : Rs.{loan_data[5]}")
            print(f"Rate for Loan Amount : {loan_data[7]}%")
            print(f"Loan Tenure: {loan_data[6]} years")
            print(f"Date of Loan Applied : {loan_data[0]}")
            print(f"Installment Date : {loan_data[3]}")
            print(f"Total loan amount to pay: Rs.{loan_data[9]}")
            print(f"Monthly Installment Amount : Rs.{loan_data[8]}\n")
            goto_customer_menu(usr_name)
```

This field shows the consumer's loan status.

```
def goto_customer_menu(usr_name):
    inp_user = input("\nDo you want to go back to previous menu (y/n)?")
    if inp_user == 'y':
```

If the client wishes to return to the main menu, they will be prompted to do so.

Selected output

```
Malaysia Bank Online Loan Management System (MBOLMS)



     -------- Main Menu ----------
           1. Admin
           2. New Customer
           3. Registered Customer
           4. Exit
```

```
Select your login method from the following (1-3): 2



            -------New Customer's Menu------

                1. Check Loan Details
                2. Loan Calculator
                3. Register new Account
                4. Go to Main Menu
```

```
Select what you want to do (1-4):3
Please fill out the following requirements to be a new customer:

Name: hari
Address: kadhagari
E-mail: harinepal@gmail.com
Contact no.: 9816340582
Gender: male
Date of birth (yyyy mm dd): 2000 12 29
```

```
-------- Main Menu ----------
        1. Admin
        2. New Customer
        3. Registered Customer
        4. Exit

Select your login method from the following (1-3): 1
```

```
        1. Check Loan Details
        2. Loan Calculator
        3. Register new Account
        4. Go to Main Menu


Select what you want to do (1-4):4


Malaysia Bank Online Loan Management System (MBOLMS)
```

```
-------- Main Menu ----------
        1. Admin
        2. New Customer
        3. Registered Customer
        4. Exit

Select your login method from the following (1-3): 1

Username: admin
Password: admin
[['admin', 'admin']]
```

```
admin admin

You have logged in successfully.



  ----------Admin Menu ------------

    1. Verify Customer Requests
    2. Verify Loan Requests
    3. View Transactions
```

```
    4. Go to Main Menu


Select what you want to do: 1
| Name   | Address   | E-mail              |   Contact No. | Gender   | DOB        | Username   | Password   | Is Admin   | User Status   |
|--------|-----------|---------------------|---------------|----------|------------|------------|------------|------------|---------------|
| Hari   | Kadhagari | harinepal@gmail.com |    9816340582 | Male     | 2000-12-29 | hari       | Hari@321   | False      | False         |

Do you want to verify user request (y/n) ?  y
Enter username: hari
```

```
hari has been verified...

| Name   | Address   | E-mail   | Contact No.   | Gender   | DOB   | Username   | Password   | Is Admin   | User Status   |
|--------|-----------|----------|---------------|----------|-------|------------|------------|------------|---------------|

Do you want to verify user request (y/n) ?  n


  ----------Admin Menu ------------

    1. Verify Customer Requests
```

```
    2. Verify Loan Requests
    3. View Transactions
    4. Go to Main Menu


Select what you want to do: 4


Malaysia Bank Online Loan Management System (MBOLMS)
```

```
      -------- Main Menu ----------
            1. Admin
            2. New Customer
            3. Registered Customer
            4. Exit

Select your login method from the following (1-3): 3

Username: hari
Password: Hari@321
```

```
You have logged in successfully.



    -------Registered Customer Menu-----

        1. View Loan Details and Apply
        2. Pay Loan Installment
        3. View Transaction
        4. Status of Loan
        5. Back to Main menu
```

```
Select what you want to do (1-4):1

---------------------------------Loan Details---------------------------------

| Loan Types          | Upto 5 years   | Upto 10 years   | Above 10 years   |
|---------------------|----------------|-----------------|------------------|
| Education Loan (HL) | 11.32%         | 12.32%          | -                |
| Car Loan (CL)       | 11.02%         | 12.52%          | -                |
| Home Loan (HL)      | 11.32%         | 12.32%          | 12.50%           |
| Personal Loan (PL)  | 10.50%         | 11.50%          | -                |
```

```
Fill the required fields to apply for loan !


Choose types of Loan:

  1. Education Loan
  2. Car Loan
  3. Home Loan
  4. Personal Loan

>>> 1
```

```
Proposed Loan: 800000
Tenure of Loans (in years): 4
Your loan request has been sent successfully...


Do you want to go back to previous menu (y/n)?y

    -------Registered Customer Menu-----

        1. View Loan Details and Apply
        2. Pay Loan Installment
```

```
        3. View Transaction
        4. Status of Loan
        5. Back to Main menu


Select what you want to do (1-4):5


Malaysia Bank Online Loan Management System (MBOLMS)
```

```
       -------- Main Menu ----------
              1. Admin
              2. New Customer
              3. Registered Customer
              4. Exit

Select your login method from the following (1-3): 1

Username: admin
Password: admin
[['admin', 'admin']]
```

```
admin admin

You have logged in successfully.



  ----------Admin Menu ------------

    1. Verify Customer Requests
    2. Verify Loan Requests
    3. View Transactions
```

```
    4. Go to Main Menu


Select what you want to do: 2
| Applied Date   | Loan ID          | Username   | Loan Type      |   Loan Amount |   Loan Tenure | Loan Status   |
|----------------|------------------|------------|----------------|---------------|---------------|---------------|
| 2021-08-09     | LiD-7165ad19df22 | hari       | Education Loan |        800000 |             4 | False         |

Do you want to verify user loans (y/n) ?y
Enter Loan ID: LiD-7165ad19df22
```

```
 Loan ID 'LiD-7165ad19df22' has been verified...

| Applied Date   | Loan ID   | Username   | Loan Type   | Loan Amount   | Loan Tenure   | Loan Status   |
|----------------|-----------|------------|-------------|---------------|---------------|---------------|

Do you want to verify user loans (y/n) ?n


  ----------Admin Menu -----------

    1. Verify Customer Requests
```

```
    2. Verify Loan Requests
    3. View Transactions
    4. Go to Main Menu


Select what you want to do: 4


Malaysia Bank Online Loan Management System (MBOLMS)
```

```
       -------- Main Menu ----------
             1. Admin
             2. New Customer
             3. Registered Customer
             4. Exit

 Select your login method from the following (1-3): 4

 Progarm has been closed successfully.
```

Conclusion

Finally, it is a web-based banking board structure that agree controllers and clients that perform multiple tasks a similar time. After signing up as a latest member of the financial institution, customers can apply for monthly installment loans and monitor the status of their loans, whether they are school loans, car loans, home loans, or personal loans. The administrator can review all the customer's transactions, check the loan status of the new client, and check the loan status of the new client.

NP000573-CT108-3-1-Python assignment

ORIGINALITY REPORT

9%	0%	0%	9%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Asia Pacific University College of Technology and Innovation (UCTI) Student Paper	9%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		