|  |
| --- |
|  |
| Electronic star bucks’ system |
| [manages basic CRUD operations with data integration using excel] |
| [Mian Zain ul Tahir] |

**Table of Contents**

[**1.** **Project Introduction** 2](#_Toc135720706)

[**2.** **Problem Analysis** 2](#_Toc135720707)

[**3.** **Project Description** 2](#_Toc135720708)

[**4.** **Project Functionality** 3](#_Toc135720709)

[**4.1. Project Development** 3](#_Toc135720710)

[**4.1.1. Python Modules** 4](#_Toc135720711)

[**4.1.2. Custom Classes** 4](#_Toc135720712)

[**5.** **UML Class Diagram** 6](#_Toc135720713)

[**6.** **Logical Flow** 7](#_Toc135720714)

[**6.1. Flow Description** 7](#_Toc135720715)

[**6.2 Flow Pseudo-Code** 8](#_Toc135720716)

[**6.3. Flow Diagram** 9](#_Toc135720717)

[**7.** **Implementation & Test Cases** 11](#_Toc135720718)

[**7.1** **Run Case** 11](#_Toc135720719)

[**Case: 1** 11](#_Toc135720720)

[**Case-1 Implementation** 12](#_Toc135720721)

[**Case-1 Screens Flow** 19](#_Toc135720722)

[**Case-1 Validations** 20](#_Toc135720723)

[**Case-2** 20](#_Toc135720724)

[**Case-2 Implementation** 20](#_Toc135720725)

[**Case-2 Screens Flow** 24](#_Toc135720726)

[**Case-2 Validations** 24](#_Toc135720727)

[**Case-3** 24](#_Toc135720728)

[**Case-3 Implementation** 25](#_Toc135720729)

[**Case-3 Screens Flow** 27](#_Toc135720730)

[**Case-3 Validations** 28](#_Toc135720731)

# **Project Introduction**

The proposed project is the development of a computerized Starbucks administration system. The system is designed to run through a graphical user interface (GUI) and provide users with a safe and convenient platform for accessing their account information, records, and memberships. Additionally, users will be able to top-up their "Starbucks Accounts" to purchase coffee products or items from the Starbucks shop. The proposed system will have the capability to identify whether the user is an existing or a new user. For new users, there will be two registration options: “customer” or “employee”. New customers will be required to provide their username and password, full name, Emirate, phone number, and email IDs. A 5-digit membership ID, consisting of both letters and numbers, will be randomly assigned to the user. Once the user's account is set up, they will be able to add money to their Starbucks account via a credit card or cash in the Starbucks Smiles Card. Similarly, new employees will be able to register as employees by providing their employee ID, job title, username and password, full name, Emirate, phone number, and email IDs. The employee will also receive a 5-digit membership ID, consisting of both letters and numbers, randomly assigned to their account.

# **Problem Analysis**

The current problem is the lack of an automated system for managing Starbucks user accounts, records, and memberships. This results in inefficiencies and errors. This results in users to rely on manual processes and multiple systems to perform account-related tasks. So, Starbucks needs a computerized administration system that can provide users with a convenient platform for accessing and managing their accounts and memberships, while also ensuring that user data is secure and accurate.

# **Project Description**

The system will have two types of users: customers and employees. New customers will be able to register by providing their username and password, full name, Emirate, telephone number, and email address. A 5-digit membership ID will be randomly assigned to each user, which they can use to access their account. Upon registration, customers can add money to their account using a credit card or by paying cash at the Starbucks shop using the Starbucks Smiles Card. New employee-users can register as an employee by providing their employee ID, job title, username, and password, along with their full name, Emirate, telephone number, and email address. An employee's account will also be assigned a 5-digit membership ID, which they can use to access their account. Like customers, employees can add money to their account using a credit card or by paying cash at the Starbucks shop using the Starbucks Smiles Card.

When a user logs in to their account, they will be able to view their account information and update it if necessary. Users can add money to their account using a credit card or pay by cash in the shop. They can then buy coffee products or any three items listed in the system. All purchases will be converted into reward points based on the amount spent. For example, a purchase of a mug for Dhs 60 will earn the user 60 reward points.

Users can also create a family account, where each extra account under the family account is linked to the main account. Each extra account will have a unique account ID, but the total money for all accounts will be linked. Therefore, anyone can top-up the total account balance, and all family members can share the total.

The system will also have a reward program based on the total points earned. Users with 1000 or more points will receive a 20% discount, while users with up to 500 points will receive a 10% discount. Users with up to 300 points will receive a 5% discount. Managers and staff will also receive discounts of 25% and 15%, respectively.

# **Project Functionality**

## **4.1. Project Development**

The Project is developed in python using object-oriented programming (OOP) fundamentals. Several python modules are used to develop this project. Other than python modules different classes are used for data manipulation across the project. Modules and classes are listed and briefly explained as below:

### **4.1.1. Python Modules**

* **System**

Code: import sys

The **sys** module provides access to some system-specific parameters and functions, such as command-line arguments and the interpreter's version.

* **Operating System**

Code: import os

The **os** module provides a way to interact with the operating system, allowing developers to perform tasks like navigating directories, manipulating files, and executing system commands.

* **PyQt5**

Code: import pyqt5

**PyQt5** is one such popular GUI framework for Python that provides a set of tools for creating cross-platform graphical user interfaces. It allows developers to create custom widgets, dialogs, and other visual elements, and also includes support for event-driven programming, which enables developers to respond to user interactions with the interface.

### **4.1.2. Custom Classes**

* **User\_Record**

This class contains all the basic attributes of the users that will be registered on the platform. A list of type <User\_Records> will contain objects of the users that will be used and will be manipulated in the project. The list of declared attributes are:

1. Emirate
2. Phone Number
3. Account Balance
4. Membership ID
5. Is Family Account
6. Family ID
7. Account Type

* **Users**

This class is inherited from the class ‘User\_Records’ and it contains the following attributes that will help in **System** **Login.**

1. Username
2. Password
3. User ID

This class has a one-to-one relation with ‘User\_Record’.

* **Employee**

Another custom written class that is inherited from ‘User\_Record’. All the attributes of this class are the same as ‘User\_Record’ and only one additional attribute is named as **Job Title.**

* **Customer**

This class is also inherited from ‘User\_Record’ and the attributes of this class are the same as ‘User\_Record’.

* **Driver**

This is the main class of the project and this class holds methods for all the data manipulation. All the CRUD (Create, Read, Write, Delete) methods are written in this class. This class is basically responsible for the managing all the other classes

* **FM class**

This class represents the file manager as per requirements this file manages all the file system. i.e., reading and writing data from and into the files respectively. All the four files are being managed by this class.

* **Purchase**

This class is to manage the data of sales. This class contain the following attributes:

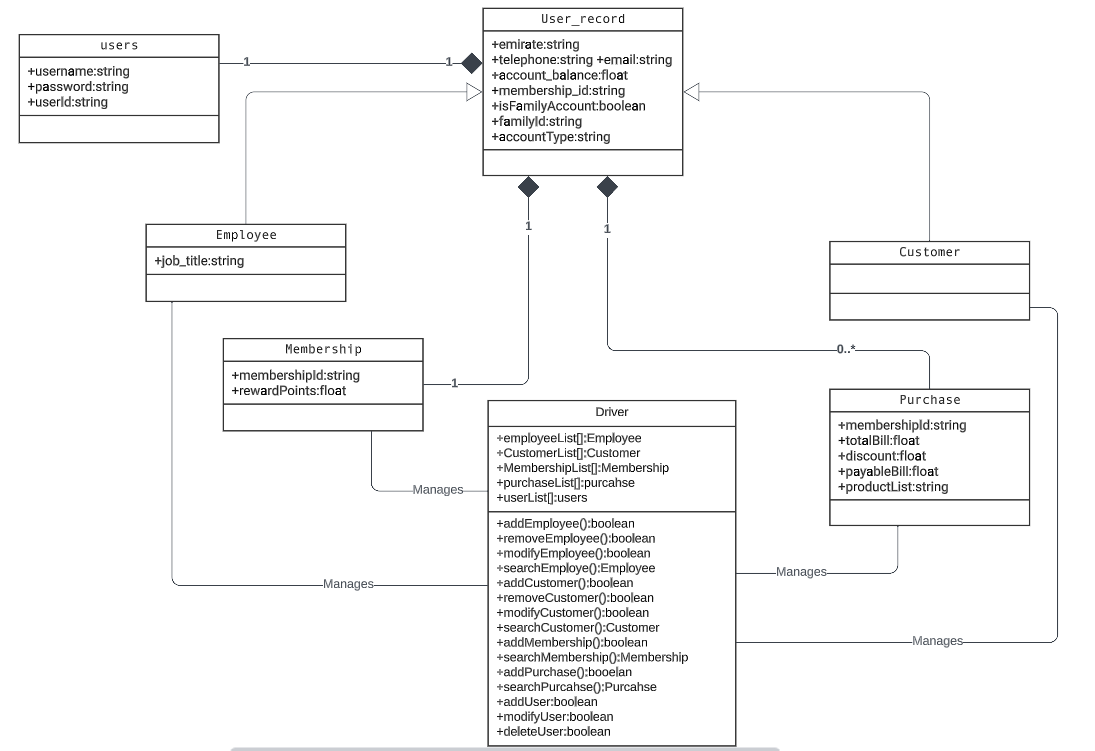
1. Membership ID
2. Total Bill
3. Discount
4. Payable
5. Product List

* **Membership**

This class has a one-to-one relation with ‘User\_Record’, this means if the user does not exist neither will the membership. This class contains the following attributes:

1. Membership ID
2. Reward Point

# **5. UML Class Diagram**

****

# **6. Logical Flow**

The basic logical flow of the project is given below. This will give you an idea of how the project will work and how the data will be managed.

## **6.1. Flow Description**

* Start the Starbucks administration system and display the login screen. By running the “homescreen.py”
* If the user is an existing user, ask for their username and password to log in. Otherwise, ask if they want to register as a customer or an employee.
* If the user wants to register as a customer, ask for their full name, Emirate, telephone, email, username, and password. Generate a 5-digit membership ID for the user and save their details in the system.
* If the user wants to register as an employee, ask for their employee ID, job title, full name, Emirate, telephone, email, username, and password. Generate a 5-digit membership ID for the employee and save their details in the system.
* After registration or login, display the user's account info and options to add money to their account using a credit card or cash in the Starbucks Smiles Card.
* Display options for the user to buy items or coffee products. After the user makes a purchase, calculate the corresponding reward points and add them to the user's account.
* If the user creates a family account, display options for them to add extra accounts linked to the main account. Each extra account should have a different account ID, but the total money for all accounts should be linked.
* Display options for anyone to top-up the total account balance and share it among all family members.
* Calculate the user's reward program based on their total points and job title. Apply the corresponding discount to the user's next purchase.
* Save all users and purchase details in the txt file for future reference.
* End the program.

## **6.2 Flow Pseudo-Code**

**Login/SignUp**

**Case:1**

If (user is an existing user):

Ask for their username and password to log in

else:

Ask if they want to register as a customer or an employee.

**Case:2**

If (user wants to register as a customer):

Ask for their full name, Emirate, telephone, email, username, and password and generate a 5-digit membership ID for the user and save their details in the system.

else:

if(user wants to register as an employee):

Ask for their employee ID, job title, full name, Emirate, telephone, email, username, and password.Generate a 5-digit membership ID for the employee and save their details in the system.

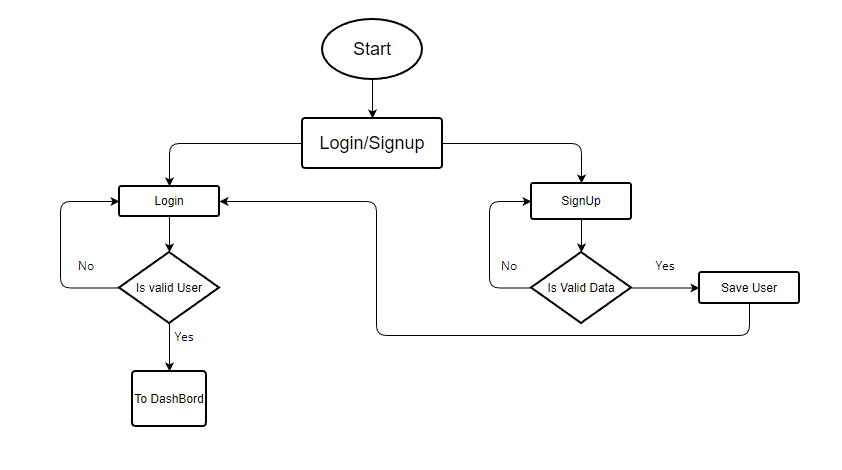
**Family Account**

If (user creates a family account):

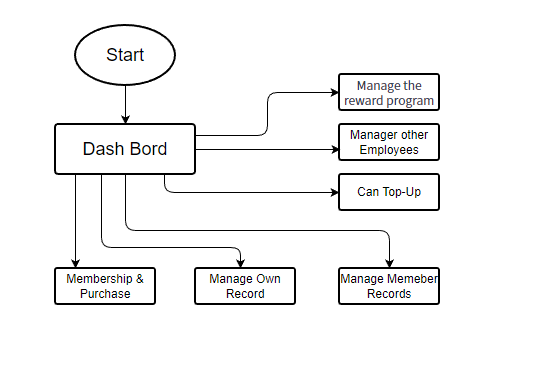
Display options for them to add extra accounts linked to the main account. Each extra account should have a different account ID, but the total money for all accounts should be linked.

## **6.3. Flow Diagram**

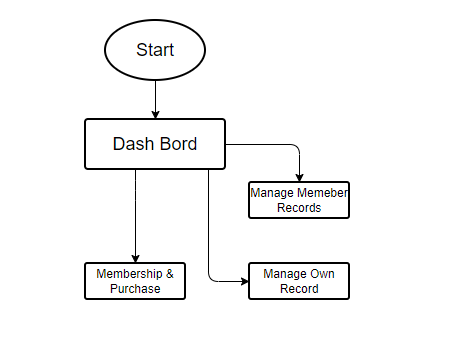
**Login/SignUp**



**Manager (After Login)**

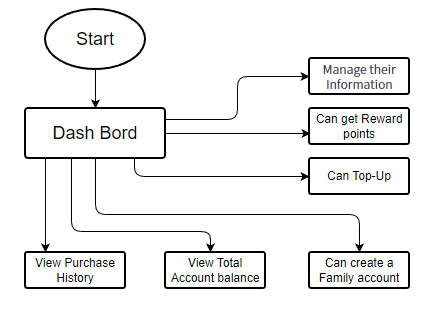
****

**Employee (After Login)**

****

Along with all the functionalities of the customer given below

**Customer (After Login)**

****

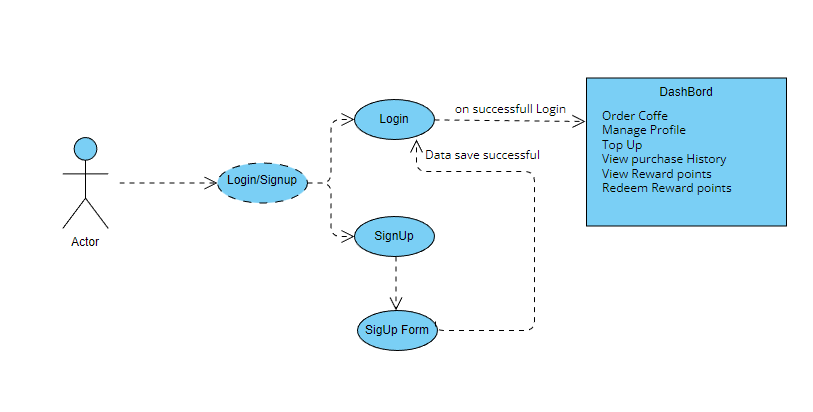
# **7. Implementation & Test Cases**

As mentioned in [Project Development](#_23ckvvd) the project is implemented using object-oriented programming using Python programming language. The classes were made by keeping in mind the basics of OOP concepts. There are total of four files as described in the requirement document named as “Product-Purchased”, “Records”, “WebUsers”, “Membership&Rewards”. All of these files mentioned previously were loaded and written in the system with help of “FM” class which implements the code to read and write and modify these text files in the system when being call upon according to respective function that is being called. GUI was implemented using PyQt5 module of python and Qt5 designer was used to design the front=end of system. Most of the validations were made using try except but a few were also done using simple if else due to complexity in validation. System will not accept any other credit card number than the real one which have been done by using the “Luhn algorithm”. Moreover, proper and necessary validations have been put in the system. And system will update its’ UI in real time i.e., when user will add money to the account it will instantly be updated in file system as well as in the dashboard. Same goes for other values like the reward points and username etc. The GUI implementation and working of the UI along with the test cases are explained below:

## **7.1 Run Case**

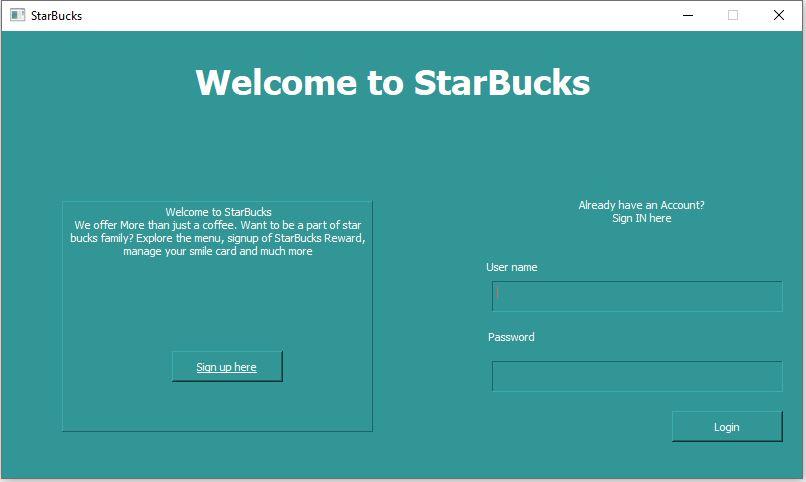
### **Case: 1**

When the user first runs the program, a login/signup screen will be displayed which then if the user is new, user will go to the signup screen and fill the required fields and the user will be registered on the platform. After the user is successfully registered on the platform, the user will be directed back to the login/signup screen where the user will use their username and password to login. If the entered username and the password are correct the user will be given access to the dashboard.

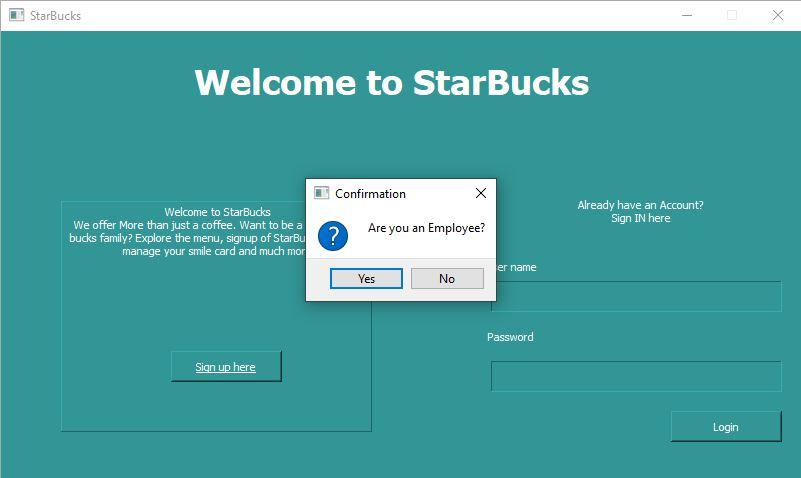


#### **Case-1 Implementation**

**Login/Sign Up Screen**

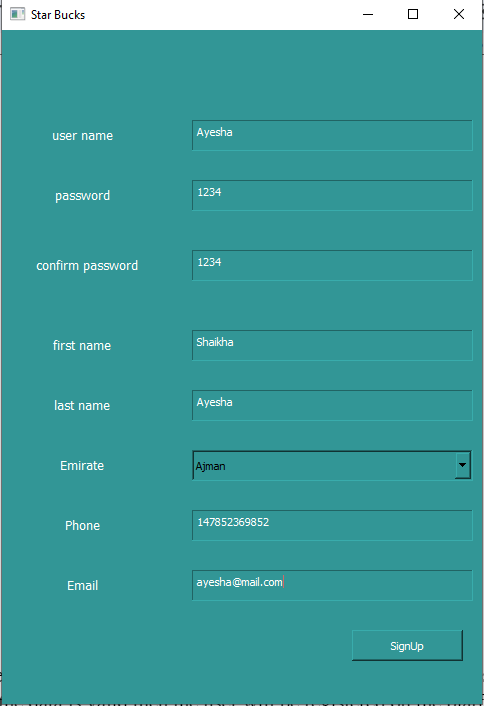


When the new user taps on the **Sign up here** button he will get the following alert box:

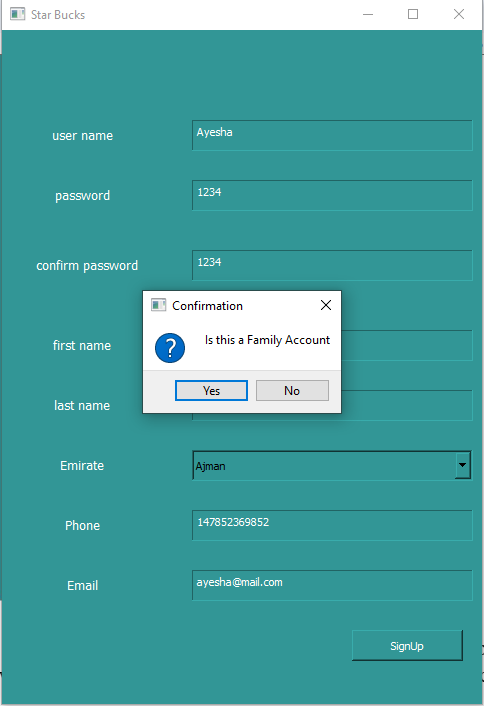


This alert box will ask if the user is an employee or a simple customer. If the user is customer the system will proceed to the customer section:

**Sign Up Form Screen**



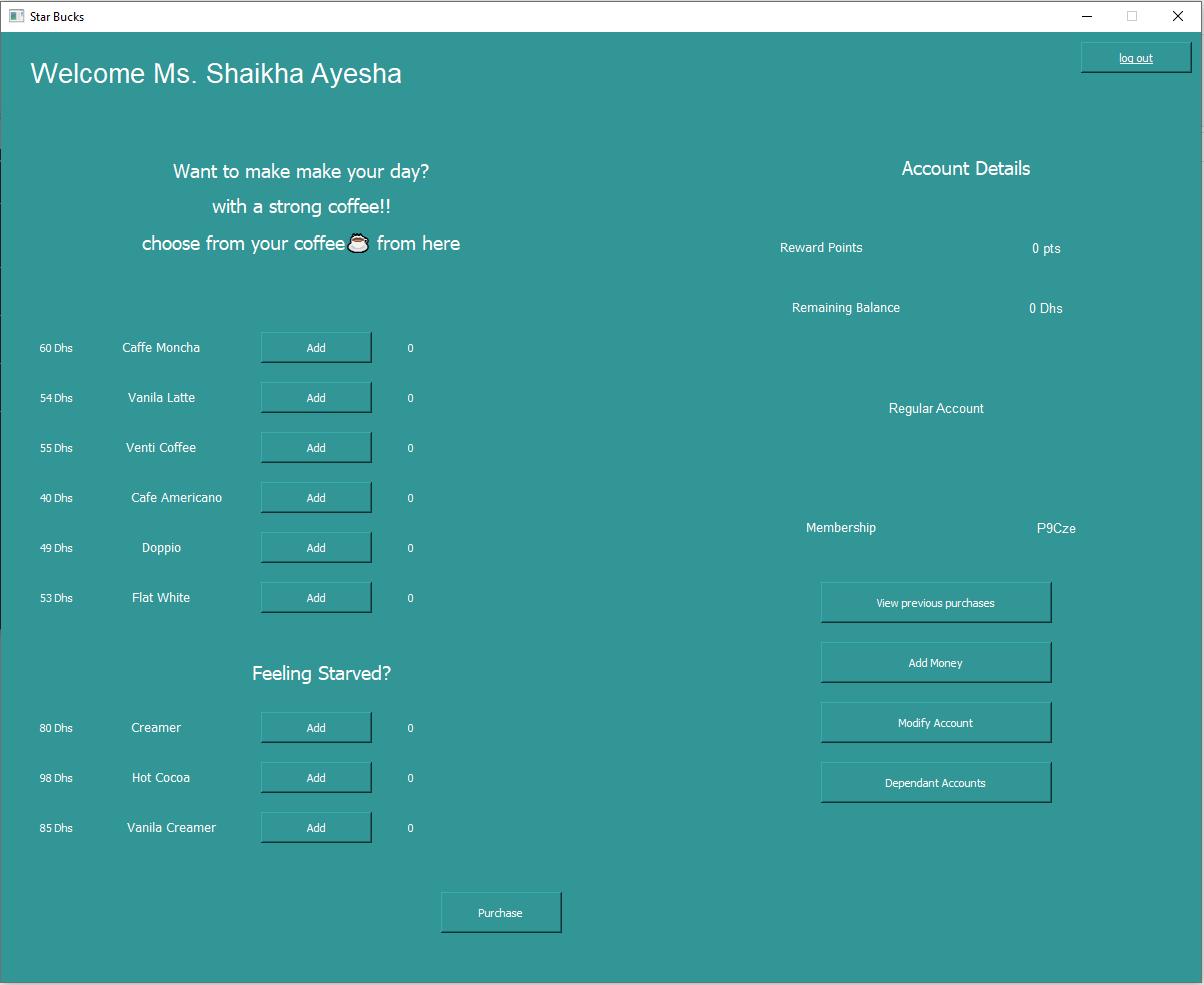
After the customer (User) has filled out all the required information press the **SignUp** button, if the data is valid then the user will be registered on the platform.

****

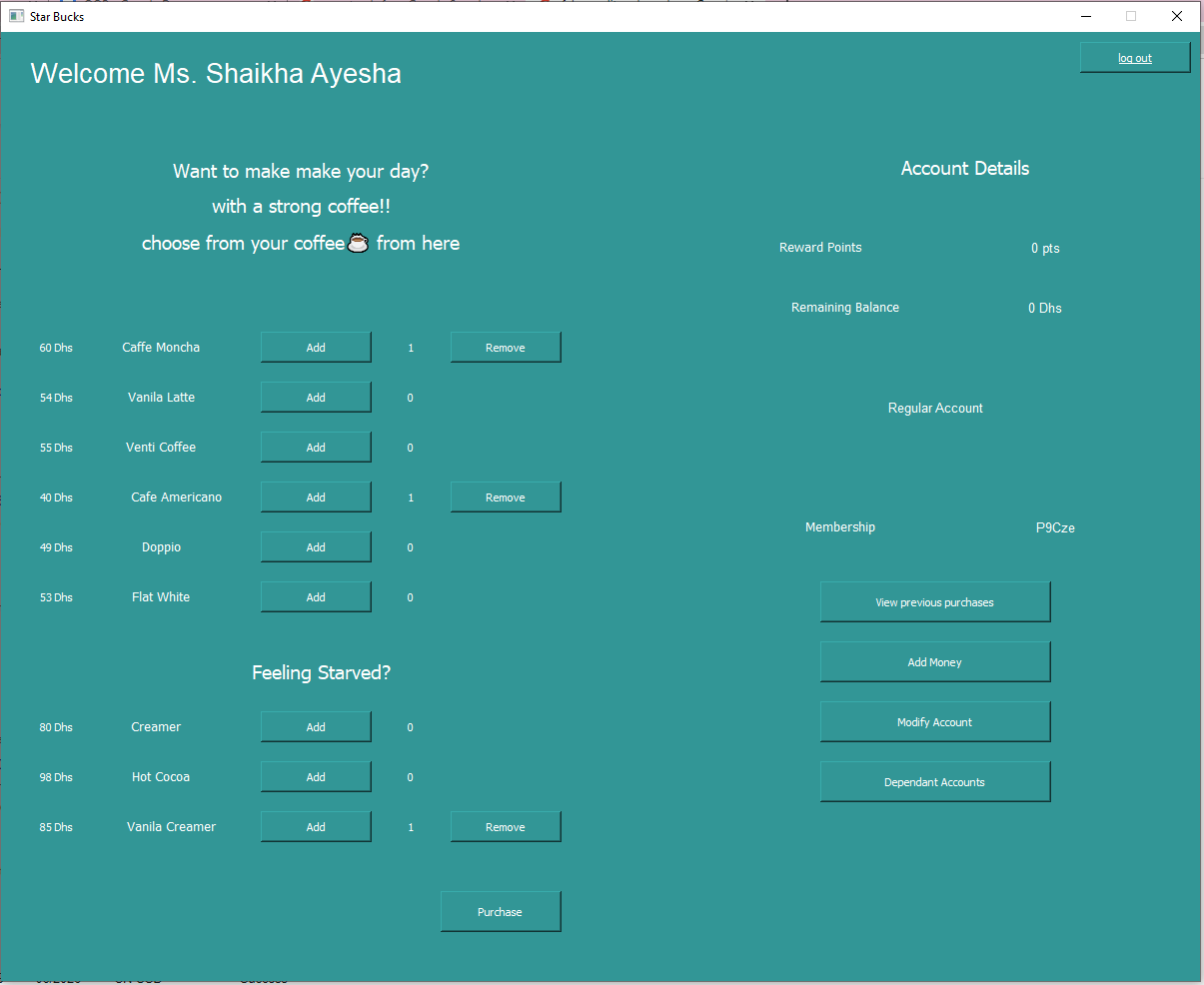
Before registering the user on the platform, the above shown alert box will pop up to ensure if the new account that is registering on the platform is a family account or not.

After a successful sign-up user will be again directed to the login screen where the user will use their username and password to get access to the dashboard of the platform:

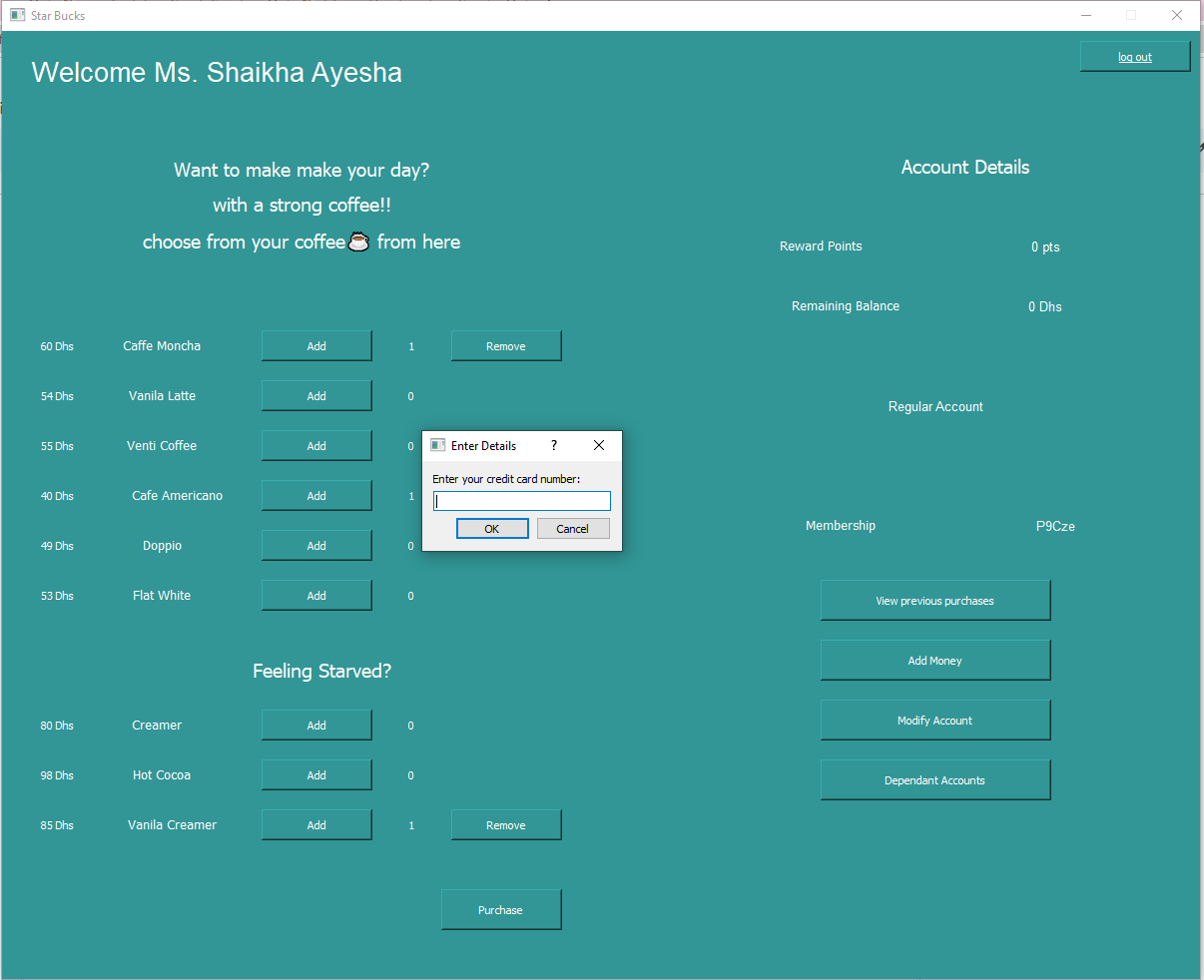
**Customer DashBoard**



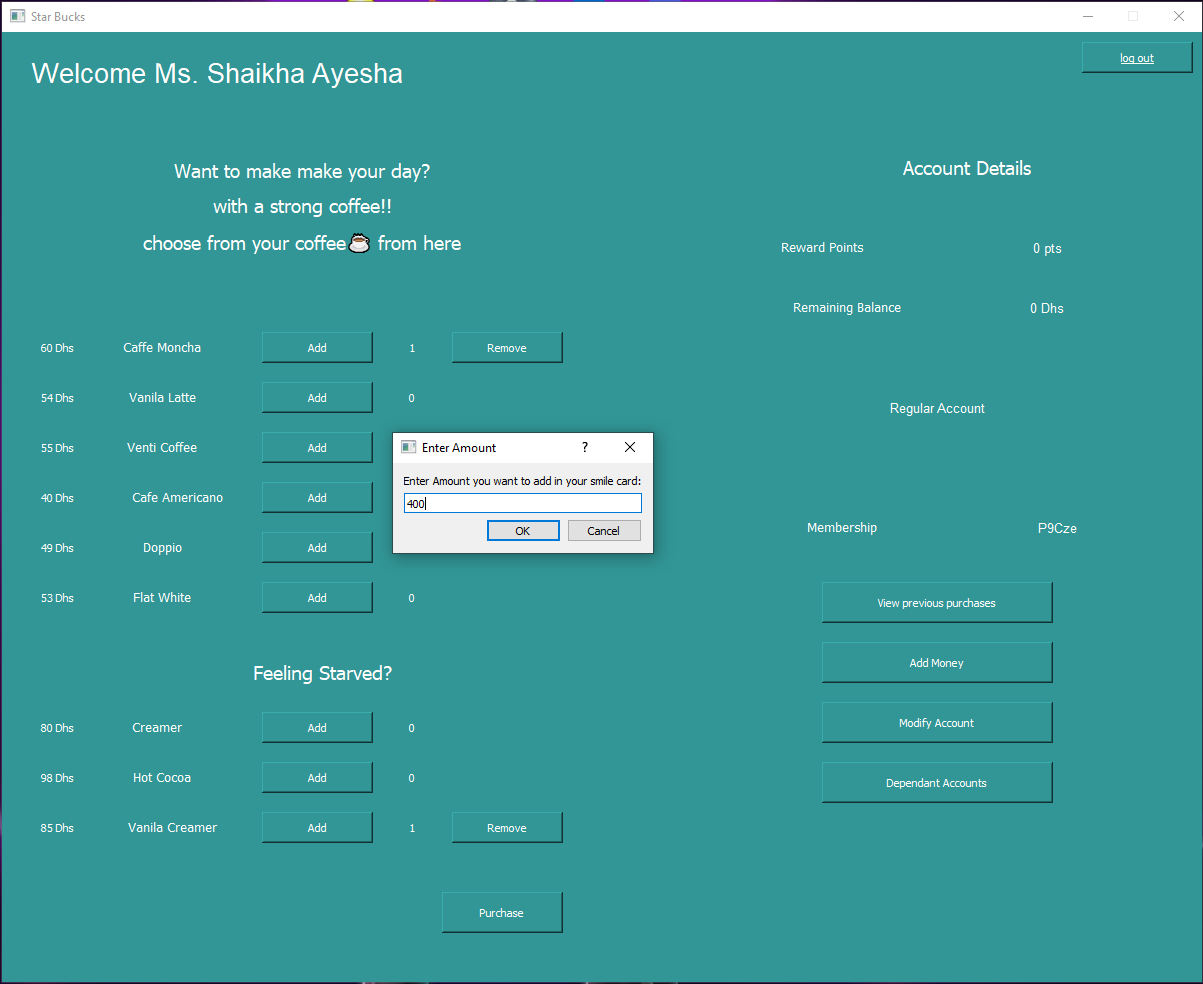
The above figure is the dashboard of the platform where the customer can view his account info, reward points, remaining balance and the list of products user and buy. Here is the working sample of the dashboard:



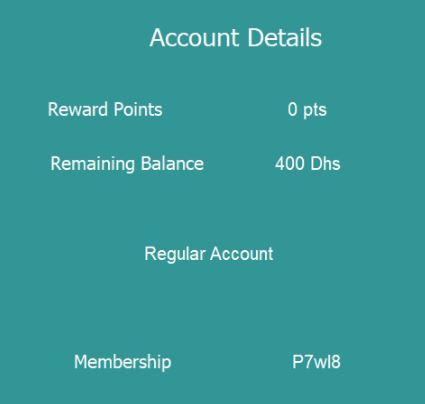
After the dashboard has been accessed by the user the first thing that user is advised to to is to Top-Up using card or by paying cash, to do that the user is simply required to tap the **Add Money** button and the following alert box will pop up:

****

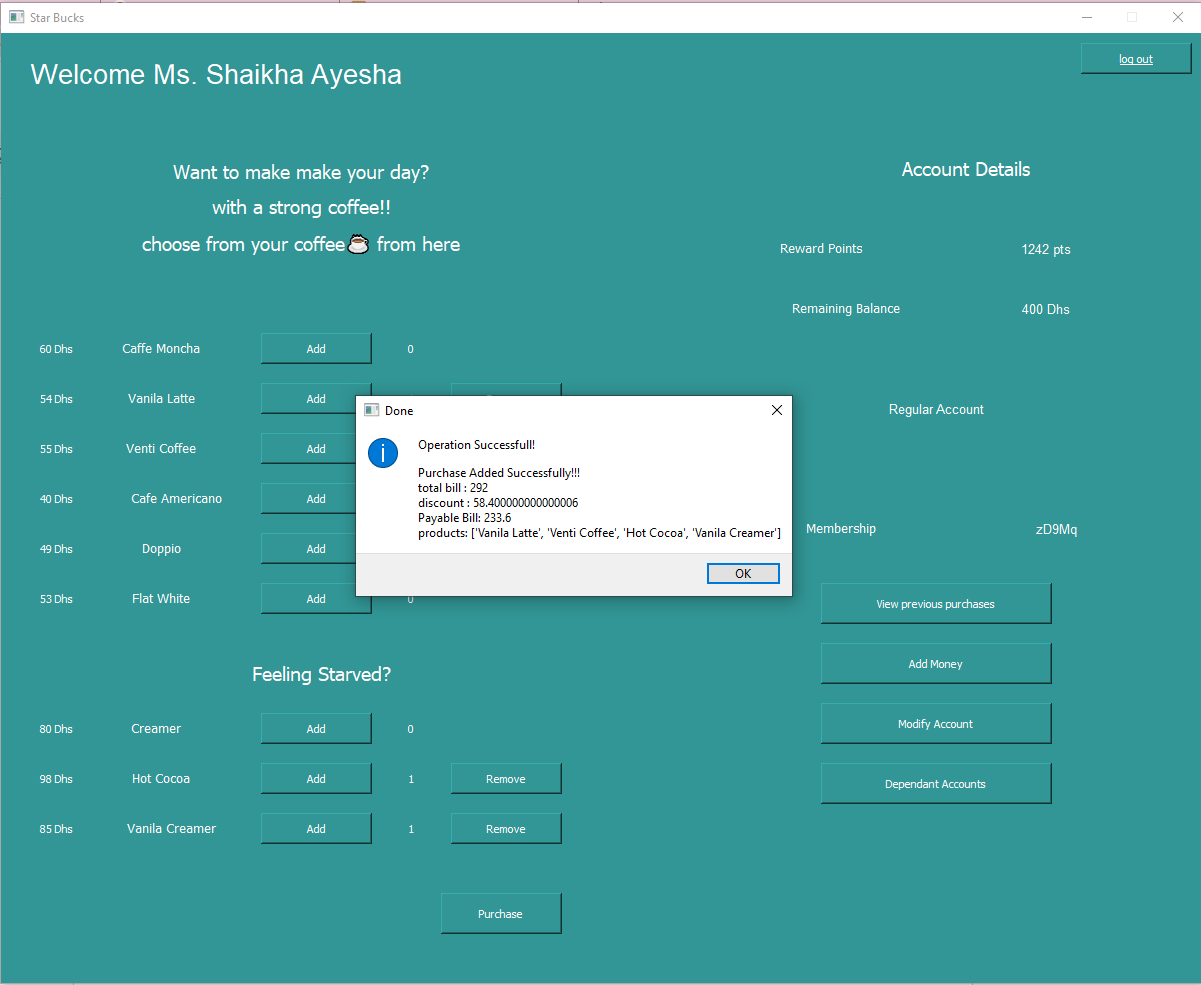
User have to provide a valid Card information to continue and after success the following pop will show asking the user how much money he wants to Top-Up:



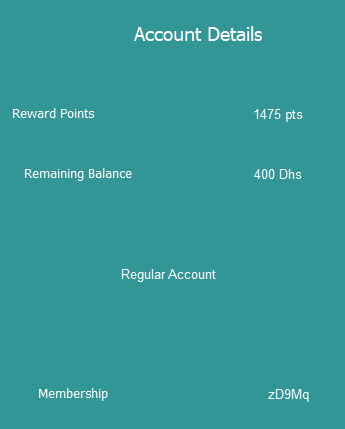
In the above test case, we have added 400 Dhs to the account and now the updated account is:



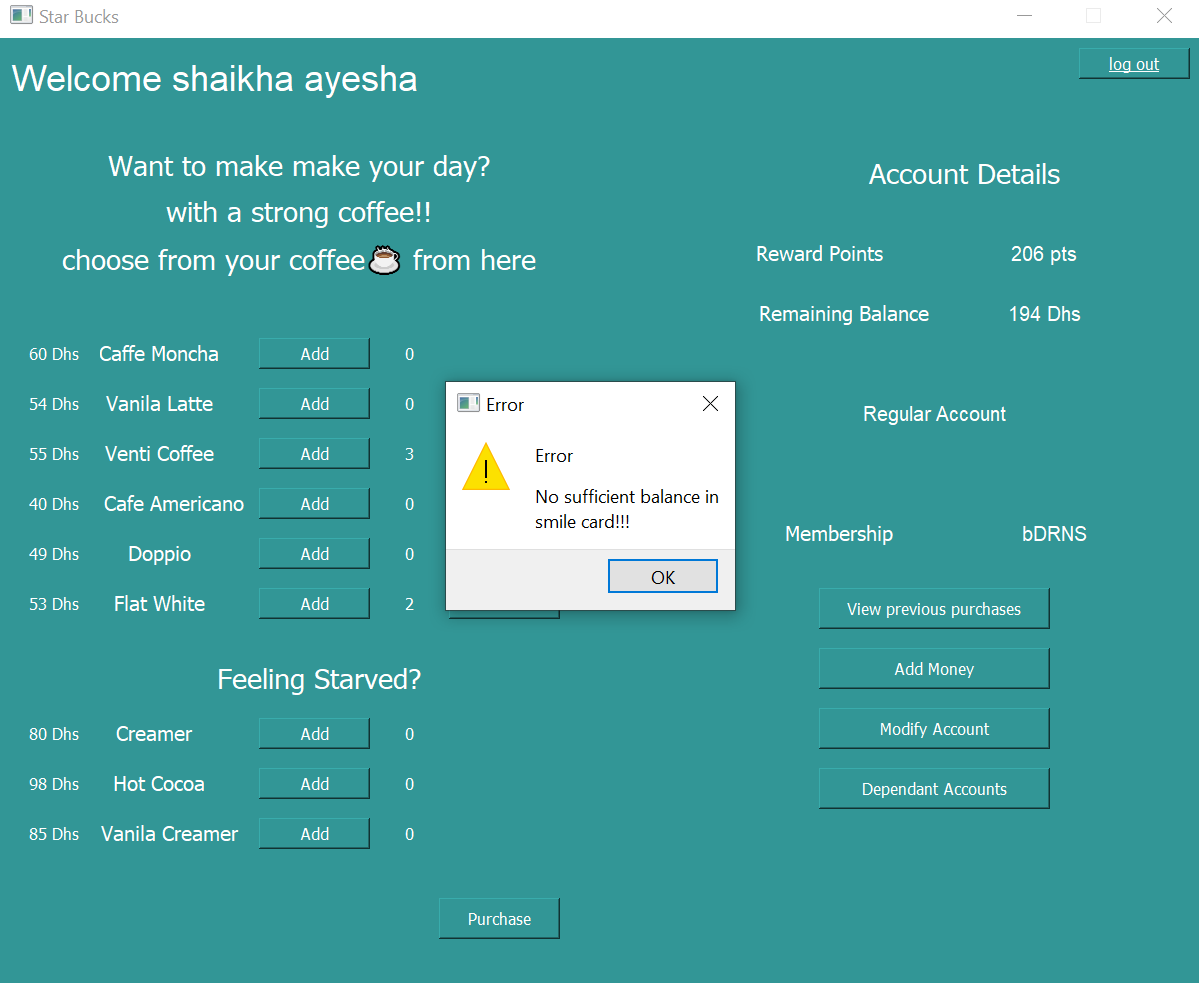
After topping up the account user can add the list to his cart that user wants to buy and after the user is done with that, the user is required to simply click on the purchase button to buy those products and a pop up will be displayed with the total bill and the list of products the user has bought:



Now after the purchase has been made user will earn the reward points and the users’ details will be updated accord to the purchase the user has made of:

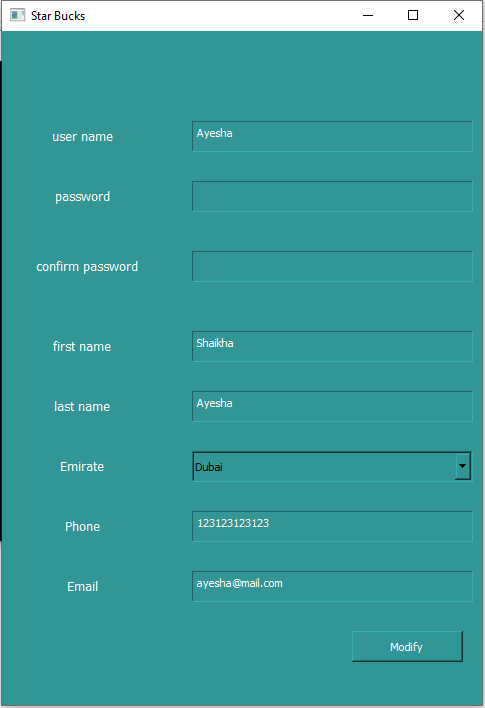


And user can not purchase if the available balance is less than the current purchase as in following screen:



The user can also view the purchase history by clicking on the **View Previous Purchases** button, along with that if the user made a mistake while entering personal information while sign up, the user is provided a **Modify Account** button that will pop up the following screen:

**Modify Account Screen**



The user can modify their account information from this screen and once the modification is made the user is logged out of the system and the user is required to re login into the system to continue.

#### **Case-1 Screens Flow**

Start > Sign Up > Sign Up Form > Login > DashBoard > Make Purchase

DashBoard > View Purchase

DashBoard > View Reward Points

DashBoard > View Account Balance

DashBoard > Top-Up

DashBoard > Modify Account

DashBoard > Modify Account > Form

If (Family Account)

DashBoard > Dependent Accounts

#### **Case-1 Validations**

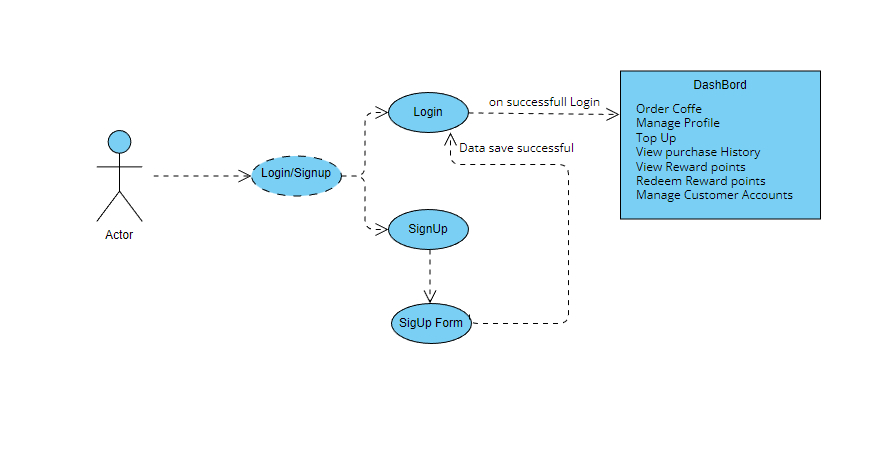
**Validation1:** SigUp form (Input Data Validation)

**Validation2:** Add Money (Input Card Details Validation)

**Validation3:** isAccountFamily(if account belongs to a family or not)

### **Case-2**

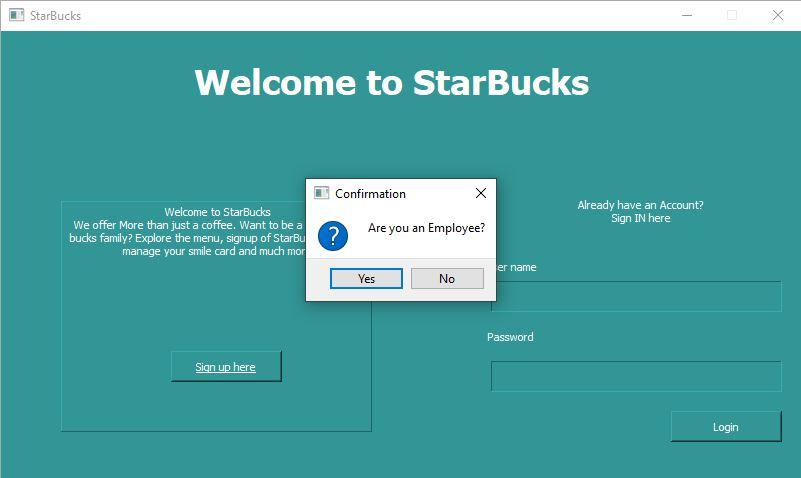
When the new employee (User) first runs the system, the work flow of screens is similar as shown:



The user is shown the same screen as the customer(User) of Login/SignUp:

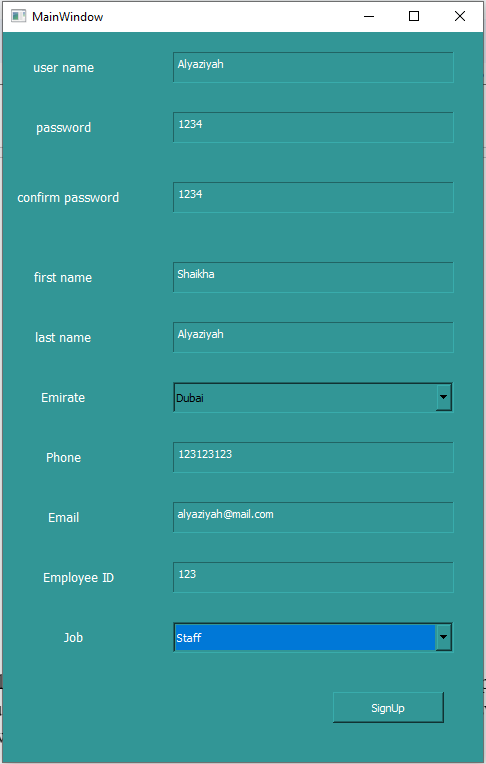
#### **Case-2 Implementation**

**Login/Sign Up Screen**



This time when the user tries to sign up, the user will click **yes** button as the user in the current context is an employee so after clicking the following form will be displayed:

**SignUp Form Screen**

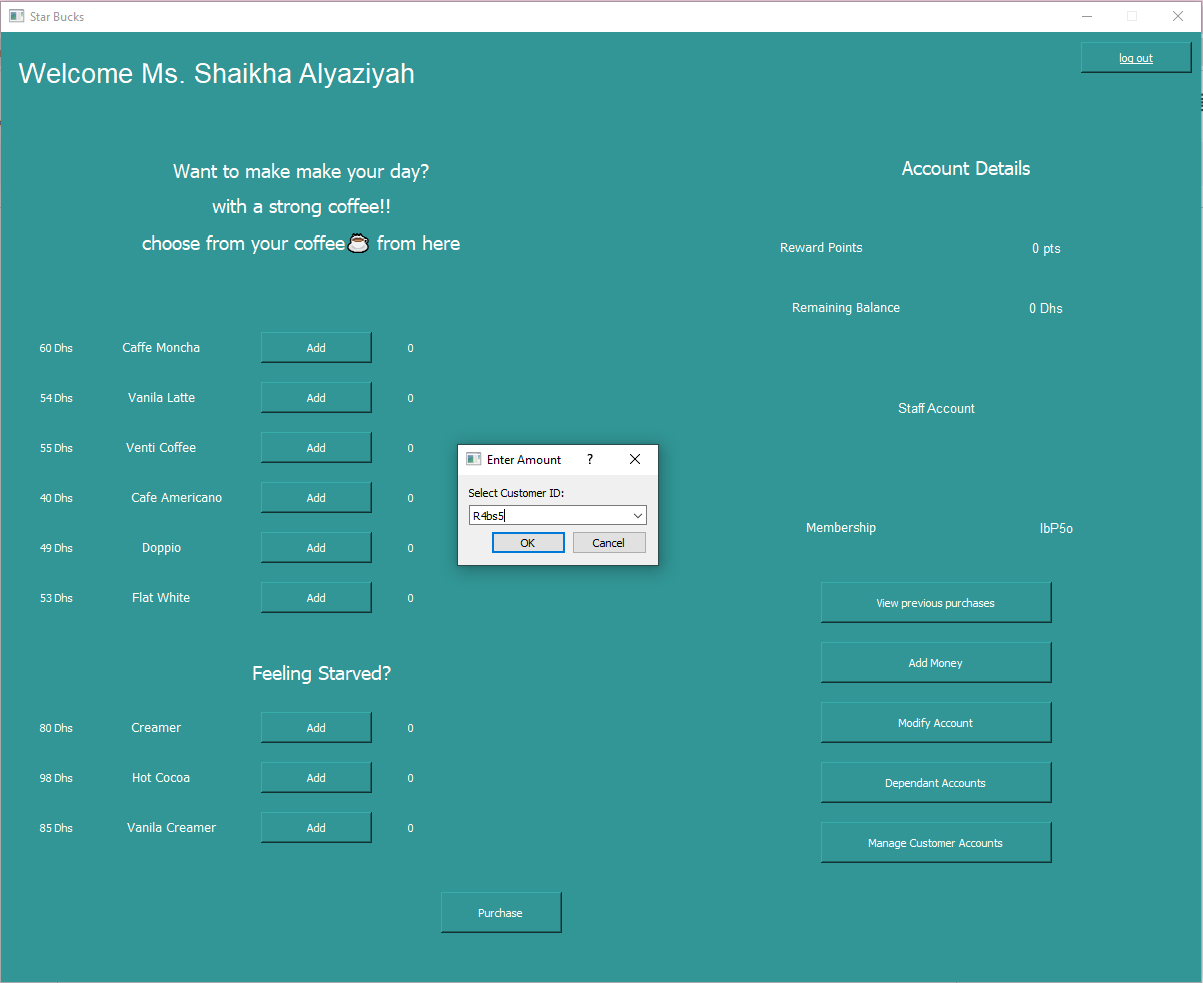
****

Job type can be selected by the given dropdown labeled as **Job.** After the employee is registered, the user is then required to log in using their username and passwords to access the following dashboard:

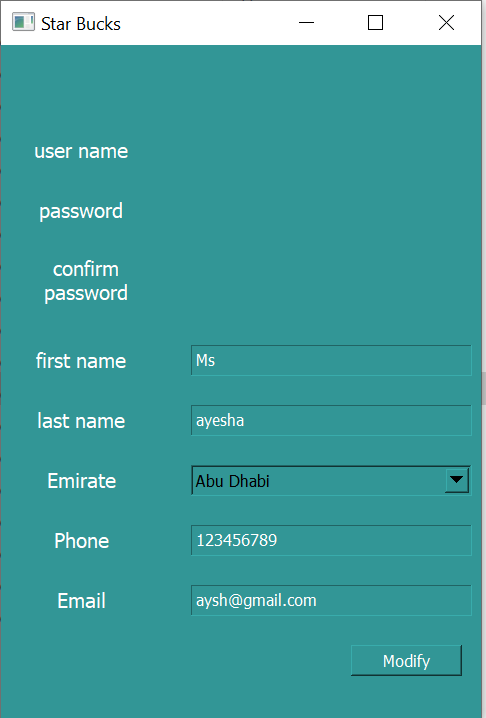
**Employee (Staff) DashBoard**

****

The working of this dashboard is nearly the same as of customer dashboard but one option available in this dashboard is what is different from the customer dashboard and that option is **Manage Customer Accounts.** The employee can use this option to manage the customer accounts, when the employee clicks on this option the following alert box pops up:



This alert box is already populated with the customer id that can be selected to open a modify form:



To modify the selected user details.

#### **Case-2 Screens Flow**

Start > Sign Up > Sign Up Form > Login > DashBoard > Make Purchase

DashBoard > View Purchase

DashBoard > View Reward Points

DashBoard > View Account Balance

DashBoard > Top-Up

DashBoard > Modify Account

DashBoard > Modify Account > Form

DashBoard > Manage Customer

DashBoard > Modify Customer > Form

If (Family Account)

DashBoard > Dependent Accounts

#### **Case-2 Validations**

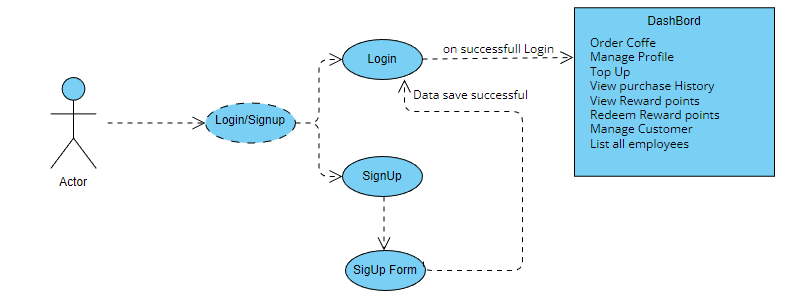
**Validation1** : SigUp form (Input Data Validation)

**Validation2** : Add Money (Input Card Details Validation)

**Validation3** : isAccountFamily

### **Case-3**

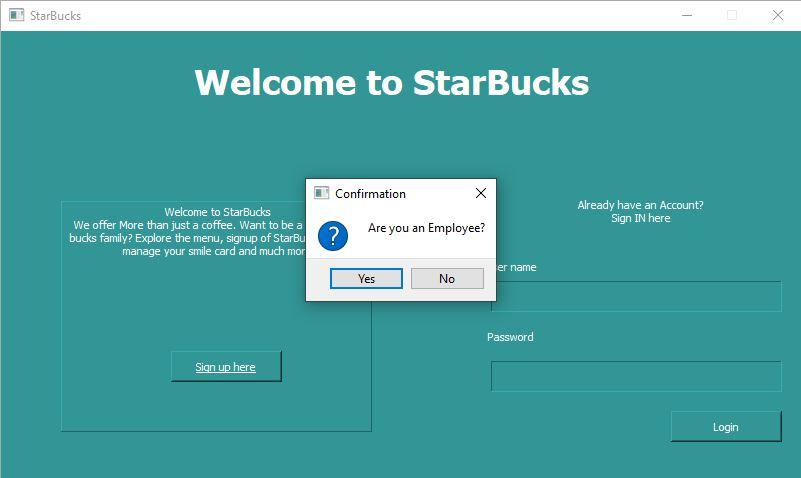
When the new employee (User-Manager) first runs the system, the work flow of screens is similar as shown:



The user is shown the same screen as the customer(User) and employee(User) of Login/SignUp:

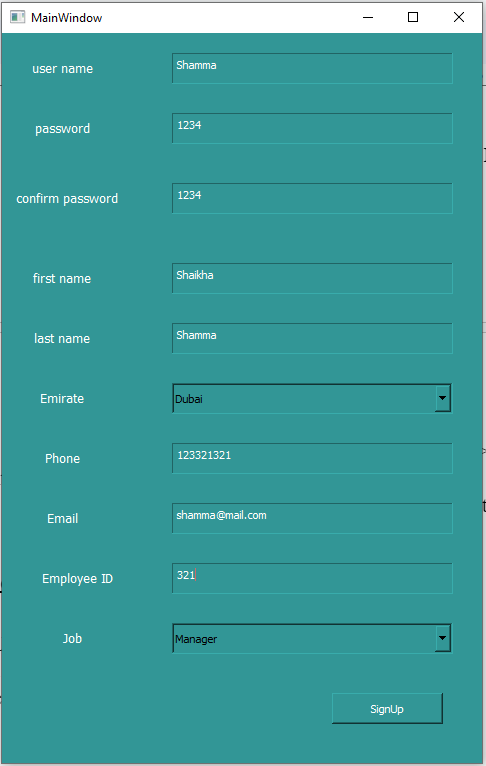
#### **Case-3 Implementation**

**Login/Sign Up Screen**



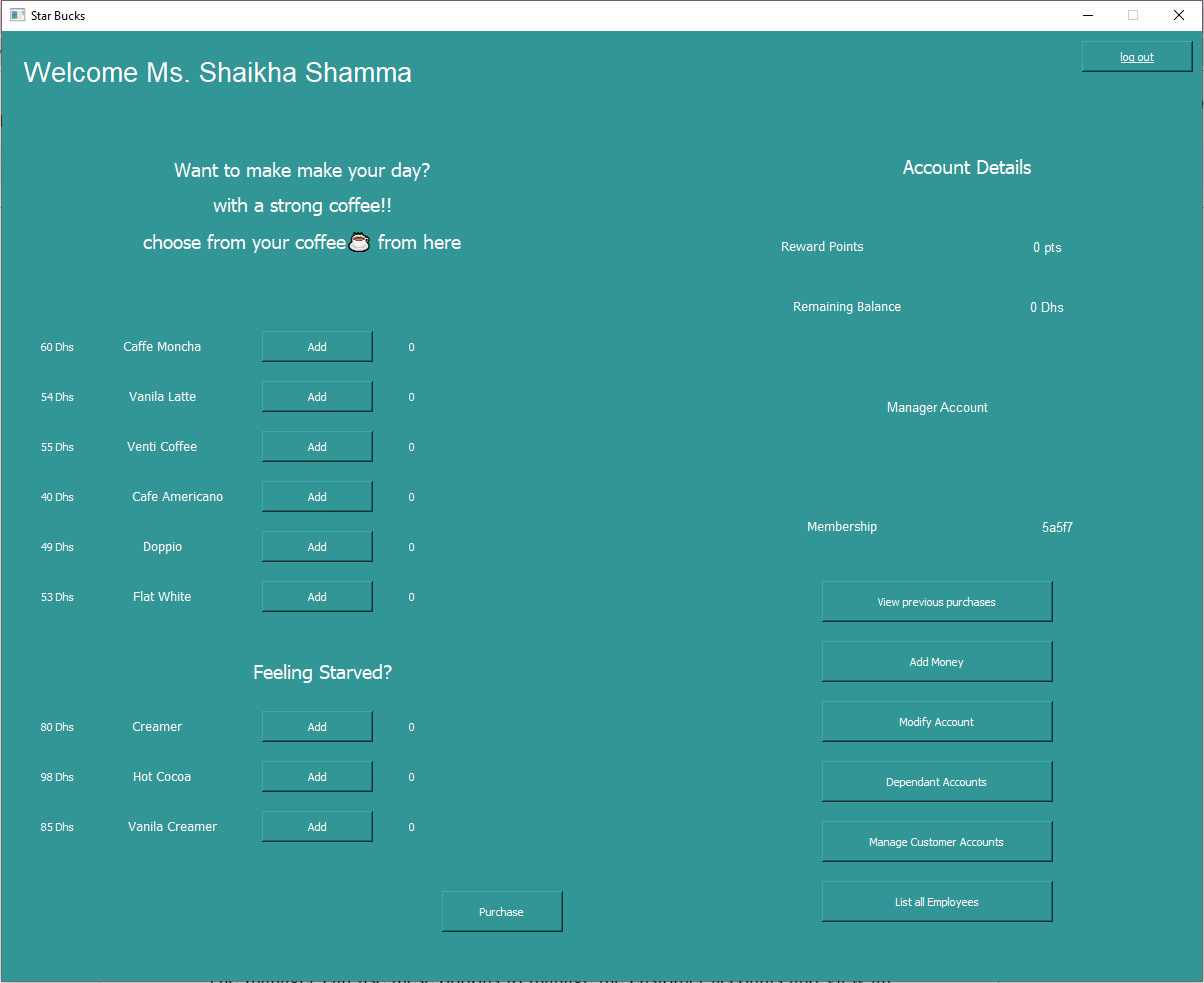
This time when the user tries to sign up, the user will click **yes** button as the user in the current context is an employee (Manager) so after clicking the following form will be displayed:

**SignUp Form Screen**

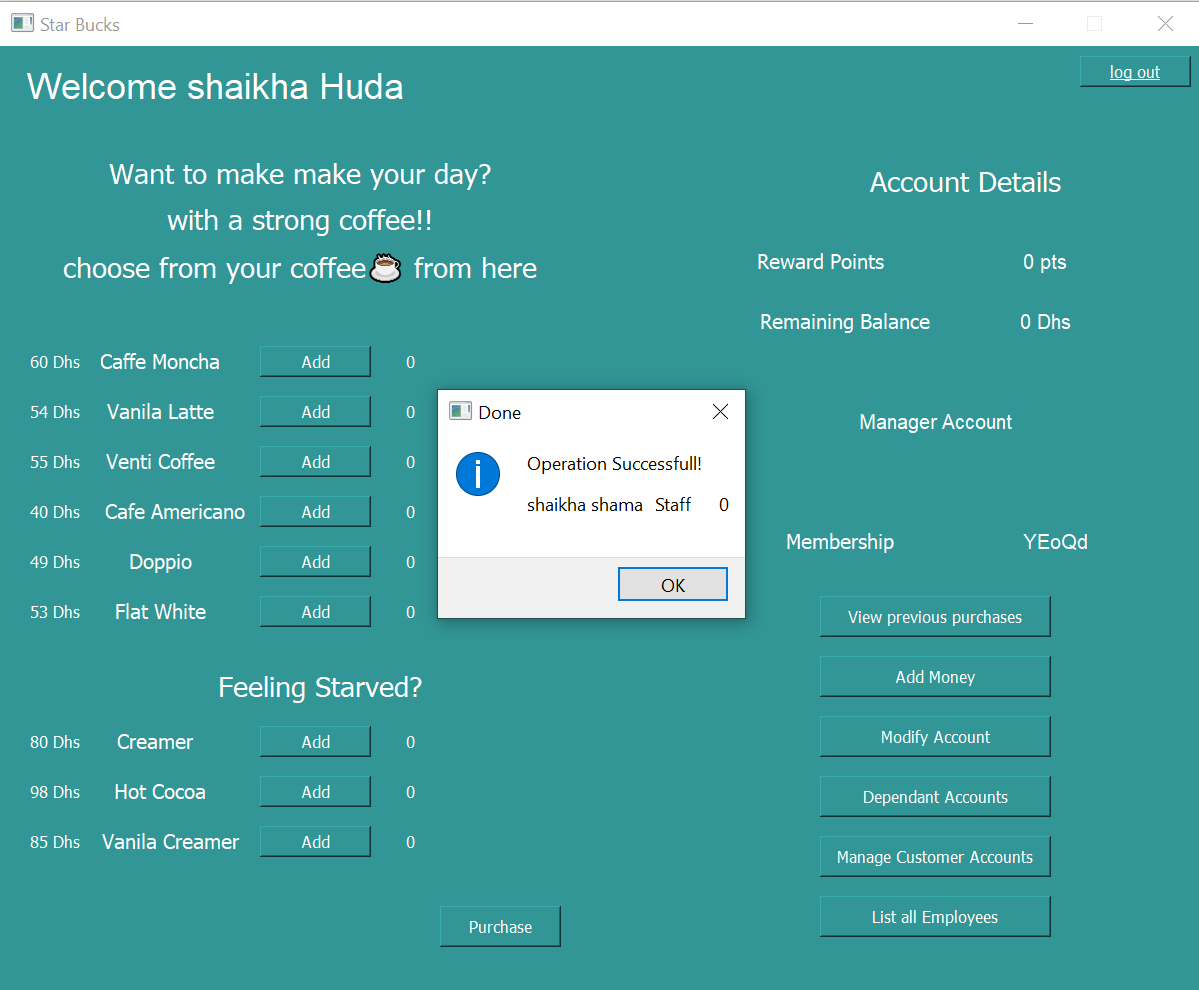
****

Job type can be selected by the given dropdown labeled as **Job.** After the the manager is registered, the user is then required to log in using their username and passwords to access the following dashboard:

**Employee (Manager) DashBoard**

****

The working of this dashboard is nearly the same as of customer dashboard but few options available in this dashboard are different from the customer dashboard and one of those options are **Manage Customer Accounts** and **List all Employees.** The manager can use these options to manage the customer accounts and view all staff members, when the Manager clicks on **Manage Customer Accounts** option the working of this option is same as for employee explained above, the other option available to the manager, when manager clicks on it the following alert box pops up with the list of all the staff members:



#### **Case-3 Screens Flow**

Start > Sign Up > Sign Up Form > Login > DashBoard > Make Purchase

DashBoard > View Purchase

DashBoard > View Reward Points

DashBoard > View Account Balance

DashBoard > Top-Up

DashBoard > Modify Account

DashBoard > Modify Account > Form

DashBoard > Manage Customer

DashBoard > Modify Customer > Form

DashBoard > List all Employees

If (Family Account)

DashBoard > Dependent Accounts

#### **Case-3 Validations**

**Validation1:** SigUp form (Input Data Validation)

**Validation2:** Add Money (Input Card Details Validation)

**Validation3:** isAccountFamily

# 

# 

# 

# 