TRANSACTION MODULE

1. Description :

In our project the Transaction module handles all the financial transactions related to the mess. It is responsible for recording and tracking the transactions such as deducting a common average per meal amount from the wallet. This module consists of three classes ManagerDB, TransactionModule and UserWallet.

2. Motivation :

The motivation behind developing a transaction module is because it is a commonly faced issue by every hostel student since they tend to mess many meals if not atleast quite a few meals every now then so by creating a mess wallet we can reduce the wastage of money which gets accounted for even the unused meals. So by developing a mess wallet so that the money from the wallet gets only when the student logs in and consumes his meal in the mess and the remaining amounts get retained in the mess wallet. If the balance runs out it also allows the user to add balance amount into the wallet. So if this gets in real life hostels it'd be really helpful for the students.

3. Relevance :

This module can be interpreted as the backbone of the project since it maintains the wallet of the user and debits the money from their account automatically once they log in. Also it keeps a track of the balance left in the user's wallet so that he/she can keep track of the amount of money left on his/her account and once after the money gets debited from the user's account it next gets into the notification module from which the user will be able to know that an amount has been deducted from their account. In a nutshell the main reason of maintaining a transaction module is because it allows the user to keep track of the money left out in their respective accounts and enables hassle free transactions from their wallet. This module also enables transparency to the user enabling them to know the balance amount remaining in the account and also they gets notified when money gets debited from their account.

4. Basic Idea behind this Module and its working:

In this module the user will be able to add create a wallet for themself and also after creating a wallet for themselves using their roll numbers, they'll be able to load money into it which further reflects in the Database which stores all the data. This module enables the user to view their balance in their respective wallets. For deducing the balance after each meal that has been utilized it goes to the Database and in there it deducts a wage of Rs.30 for every meal that has been consumed by the user. Unlike the regular orthodox way of paying mess fees this is different because in our system the money gets deducted only if the user consumes his/her meal if not consumed the money remains static in wallet, but in the orthodox way there is no such way of retaining back the money for the un used meals.

4.1 ManagerDB Class:

This class contains a method named creditMoney which is responsible for updating the balance of the manager/user in a CSV file. The creditMoney method adds the money to the manager/user's balance by taking an input parameter called amount, which represents the money to be added. Firstly it opens the file specified by the filePath variable. This file contains the manager/user's data including their current balance. It then reads each line of the file to find the line that contains the user's balance, after finding it, it extracts the existing balance value from that line. Then the program adds the 'amount' parameter to the existing balance which we've mentioned above. After adding the updated amount is added back to the line in the file. After reading and modifying all the lines, the program closes the file. Next, it opens the file again, but this time for writing. It writes the updated content, including the modified line with the updated balance, back to the file. Finally, it closes the file again, completing the process. To put it all together this class reads a file that stores the user's data and their account balance. It then finds the line containing the user's balance and further adds the specified amount to that balance, and updates the CSV file with the new balance.

4.2 **TransactionModule Class:**

This class contains a method named makeEntry that handles making an entry for a user in CSV file and performing some related operations. Same as the previous class this has a public variable named ‘file path’ , which stores the path of the CSV file where the entry data is stored. Then the class created instances of other classes, such as UserDataBase, UserWallet and NotificationSender. These classes are responsible for handling the user data, managing the users wallets and sending notifications. Inside the same method there is a check to see If the provided RollNumber exists in the user database. This is performed by using yhe checkRollExists method of the UserDataBase class. If the RollNumber exists in the database, the code proceeds to perform some operations related to making an entry, It then retrieves the phone number and name associated with the RollNumber from the user database using the RegFileReadNum and RegFileReadName methods of the UserDataBase class. Then it creates a FileWriter to open the entry data file specified by filePath. It appends data to the file. Then it creates an entryData string that contains the RollNumber and the formatted date and time. The entryData is written to the file using append method of the FileWriter. The debitMoney method of the UserWallet class is called to deduct 30 rupees from the user’s wallet balance, which is associated with the provided RollNumber after the consumption of each meal. After this the sendConfirmationSMS method of the NotificationSender class is called to send a confirmation SMS to the user, using their name and phone number which is a segment of the Notification module. As an else statement to the first if statement if the entered RollNumber doesn’t exist in the database, an error message is displayed. To put it all together this class handles the process of making an entry for a user in a CSV file which is a continuation of the User Management module. It checks if the user exists, retrieves their information, writes the entry data to the file, deducts a sum of rs.30 upon entry considering it consumption of a meal from their wallet balance, sends a confirmation SMS which is further explained in detail on the notification module and displays a message box indicating successful entry.

4. 3 UserWallet Class:

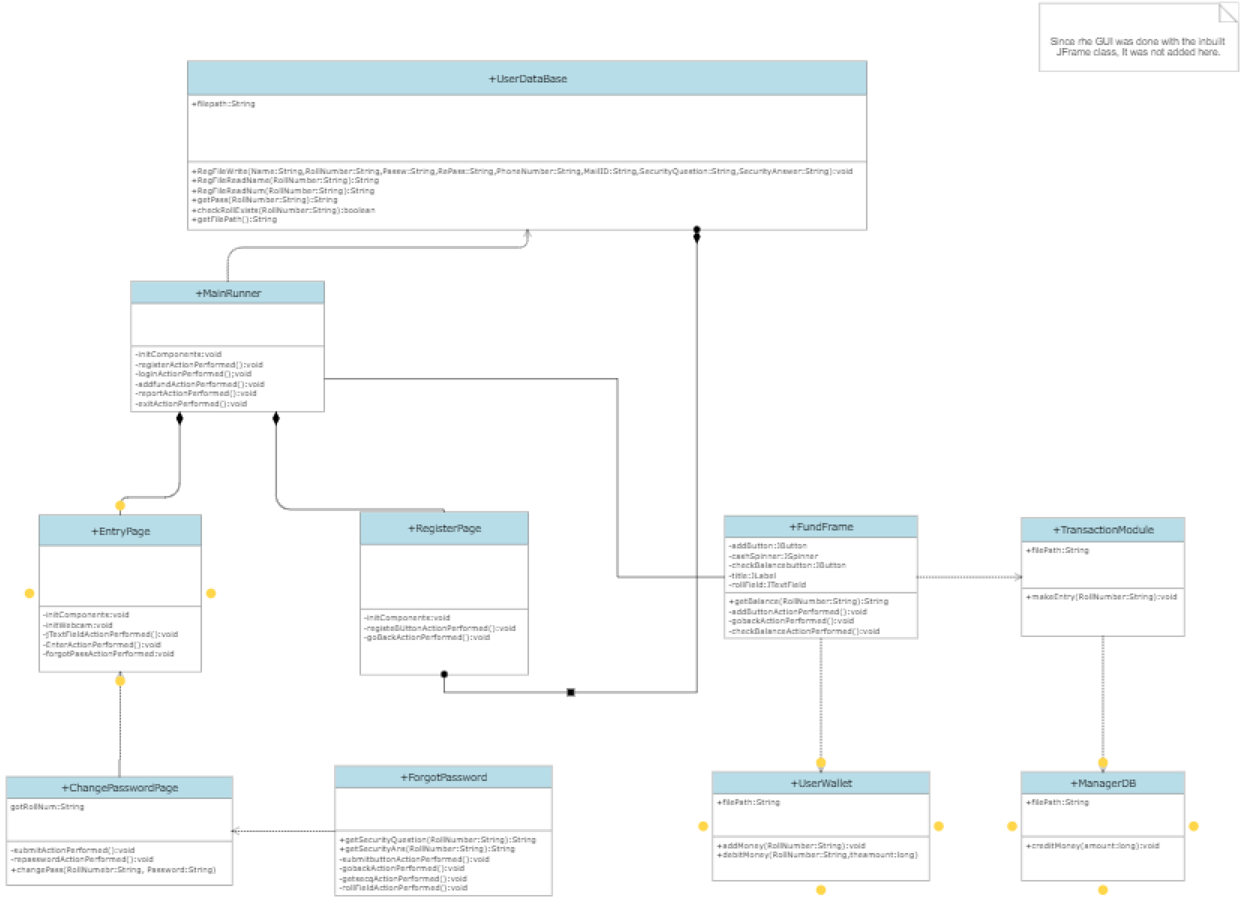
This class mainly focuses on the functionality related to user wallets. It allows users to add money to their wallets and debit money from their wallets this is what this class does in a nutshell. Firstly the ‘addMoney’ function has two parameters: RollNumber which is for identifying the user and amount for the money to be added. It further reads the contents of a file that stores user data, searches for the user within the specified RollNumber, and updates their wallet balance by adding the provided amount. If the user exists, the function displays a message indicating that the money has been credited to their wallet. If the user doesn’t exist, it displays an error message. The ‘debitMoney’ function also works in a similar way. It takes the RollNumber and theamount as parameters. It reads the user data file, searches for th4e user with the specified RollNumber and updates their wallet balance by theamount which is a sum of Rs.30. Suppose incase the user has insufficient funds in their wallet or the user doesn’t exist it displays an error message. In addition to this it calls a creditMoney function from the ManagerDB class to credit the debited amount elsewhere. To put them all together in a concise way these functions allows the users to add money to their wallets and spend money from their wallets, keeping track of their balance in our mess management system.

5. Conclusion:

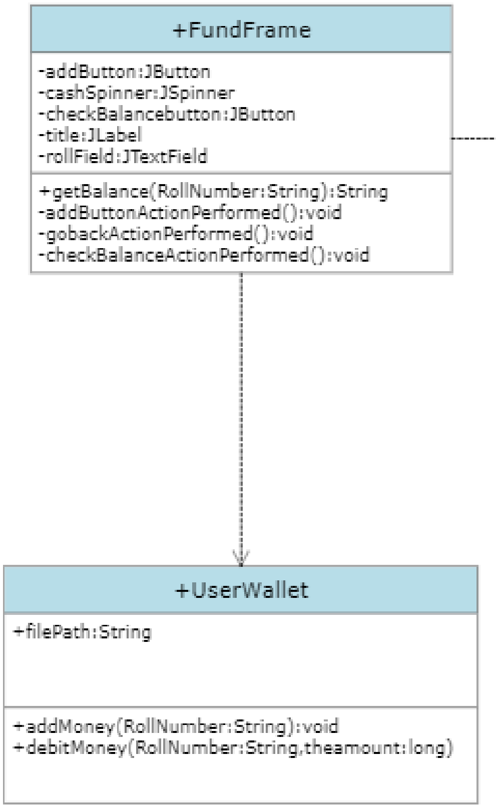
The Transaction Module in whole helps the users maintain a balance in their respective wallets, facilitating financial transactions within the mess management system. With this module users can conveniently add funds to their wallets at ease and upon consumption of every meal a fixed sum of Rs.30 gets deducted from the users wallet.

Appendix:

Class diagram of the whole project:

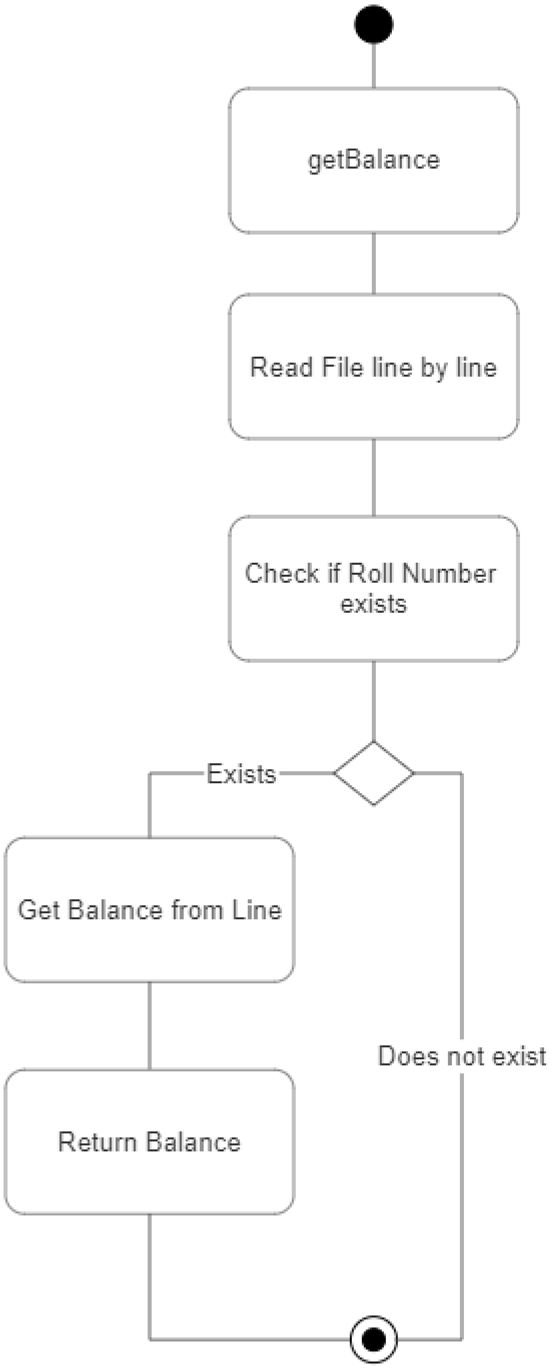

Since rhe GUI was done with the inbuilt JFrame class, It was not added here.
+UserDataBase
+filepath::String
+RegFileWrite(Name::String,RollNumber::String,Passw::String,RePass::String,PhoneNumber::String,MailID::String,SecurityQuestion::String,SecurityAnswer::String):void
+RegFileReadName(RollNumber::String):String
+RegFileReadNum(RollNumber::String):String
+getPass(RollNumber::String):String
+checkRollExists(RollNumber::String):boolean
+getFilePath():String
+MainRunner
-initComponents:void
-registerActionPerformed():void
-loginActionPerformed():void
-addfundActionPerformed():void
-reportActionPerformed():void
-exitActionPerformed():void
+EntryPage
-initComponents:void
-initWebcam:void
-jTextFieldActionPerformed():void
-EnterActionPerformed():void
-forgotPassActionPerformed:void
+RegisterPage
-initComponents:void
-registeButtonActionPerformed():void
-goBackActionPerformed():void
+FundFrame
-addButton::JButton
-cashSpinner::JSpinner
-checkBalancebutton::JButton
-title::JLabel
-rollField::JTextField
+getBalance(RollNumber::String):String
-addButtonActionPerformed():void
-gobackActionPerformed():void
-checkBalanceActionPerformed():void
+TransactionModule
+filePath::String
+makeEntry(RollNumber::String):void
+ChangePasswordPage
getRollNum::String
-submitActionPerformed():void
-repasswordActionPerformed():void
+changePass(RollNumebr::String, Password::String)
+ForgotPassword
+getSecurityQuestion(RollNumber::String):String
+getSecurityAns(RollNumber::String):String
-submitbuttonActionPerformed():void
-gobackActionPerformed():void
-getsecqActionPerformed():void
-rollFieldActionPerformed():void
+UserWallet
+filePath::String
+addMoney(RollNumber::String):void
+debitMoney(RollNumber::String,theamount::long)
+ManagerDB
+filePath::String
+creditMoney(amount::long):void

Class diagram of the module:

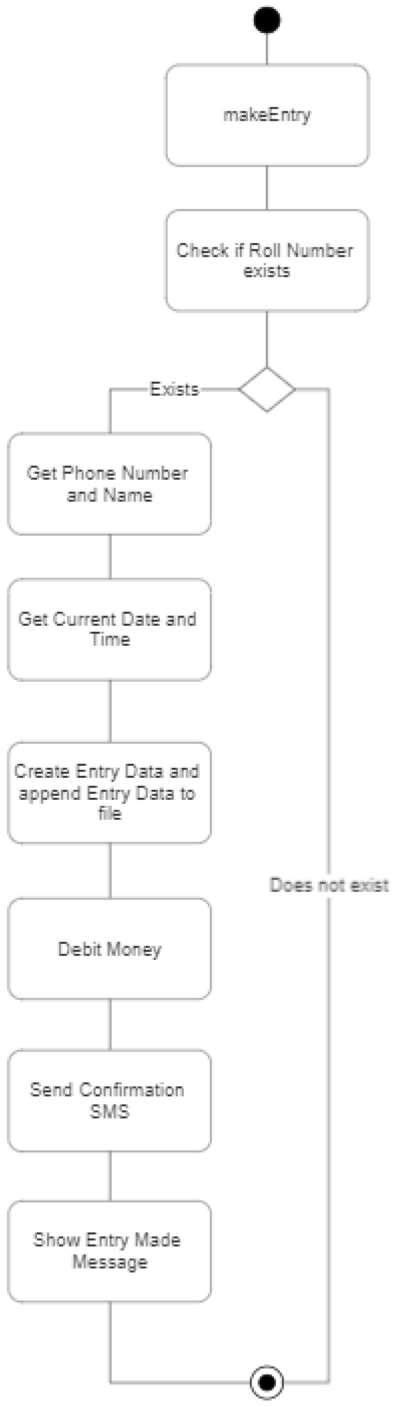

+FundFrame
-addButton:JButton
-cashSpinner:JSpinner
-checkBalancebutton:JButton
-title:JLabel
-rollField:JTextField
+getBalance(RollNumber:String):String
-addButtonActionPerformed():void
-gobackActionPerformed():void
-checkBalanceActionPerformed():void
+TransactionModule
+filePath:String
+makeEntry(RollNumber:String):void
+UserWallet
+filePath:String
+addMoney(RollNumber:String):void
+debitMoney(RollNumber:String,theamount:long)
+ManagerDB
+filePath:String
+creditMoney(amount:long):void

Activity Diagram of each function:

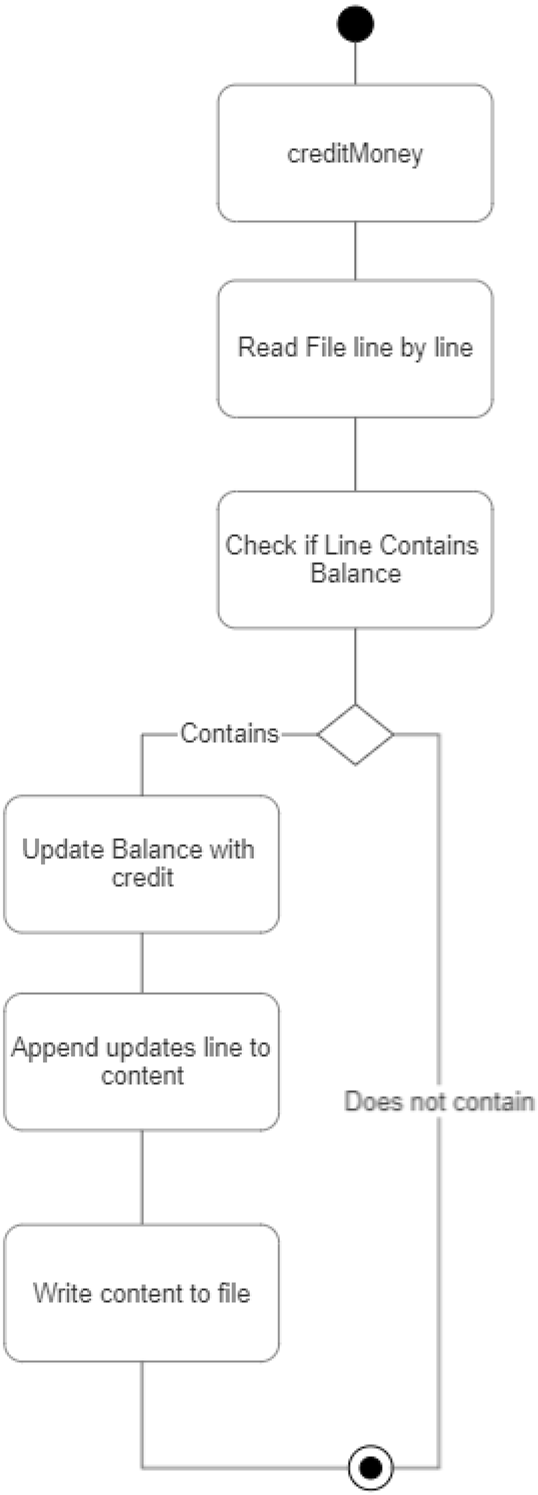
getBalance:



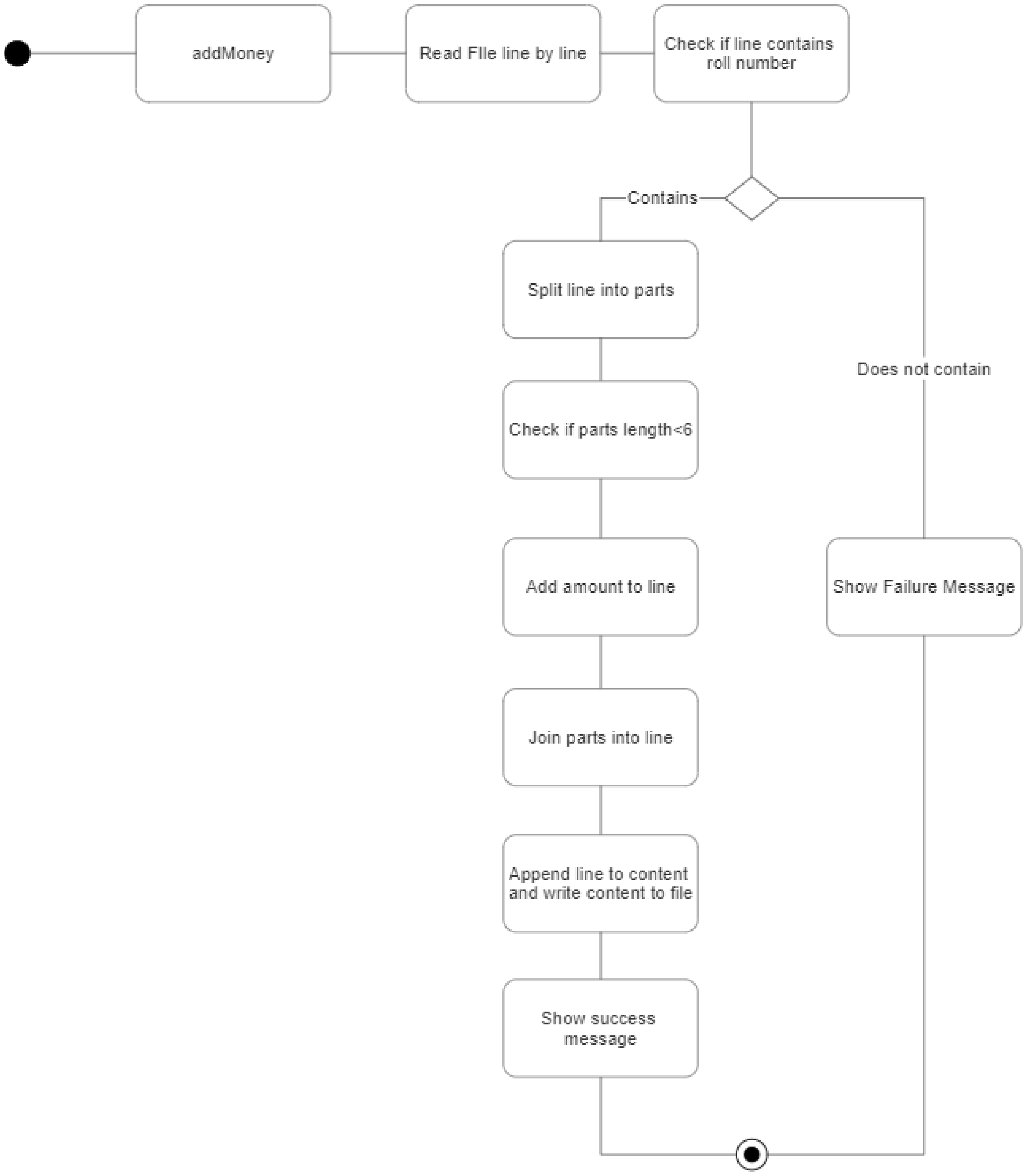
makeEntry:


makeEntry
Check if Roll Number exists
Exists
Does not exist
Get Phone Number and Name
Get Current Date and Time
Create Entry Data and append Entry Data to file
Debit Money
Send Confirmation SMS
Show Entry Made Message

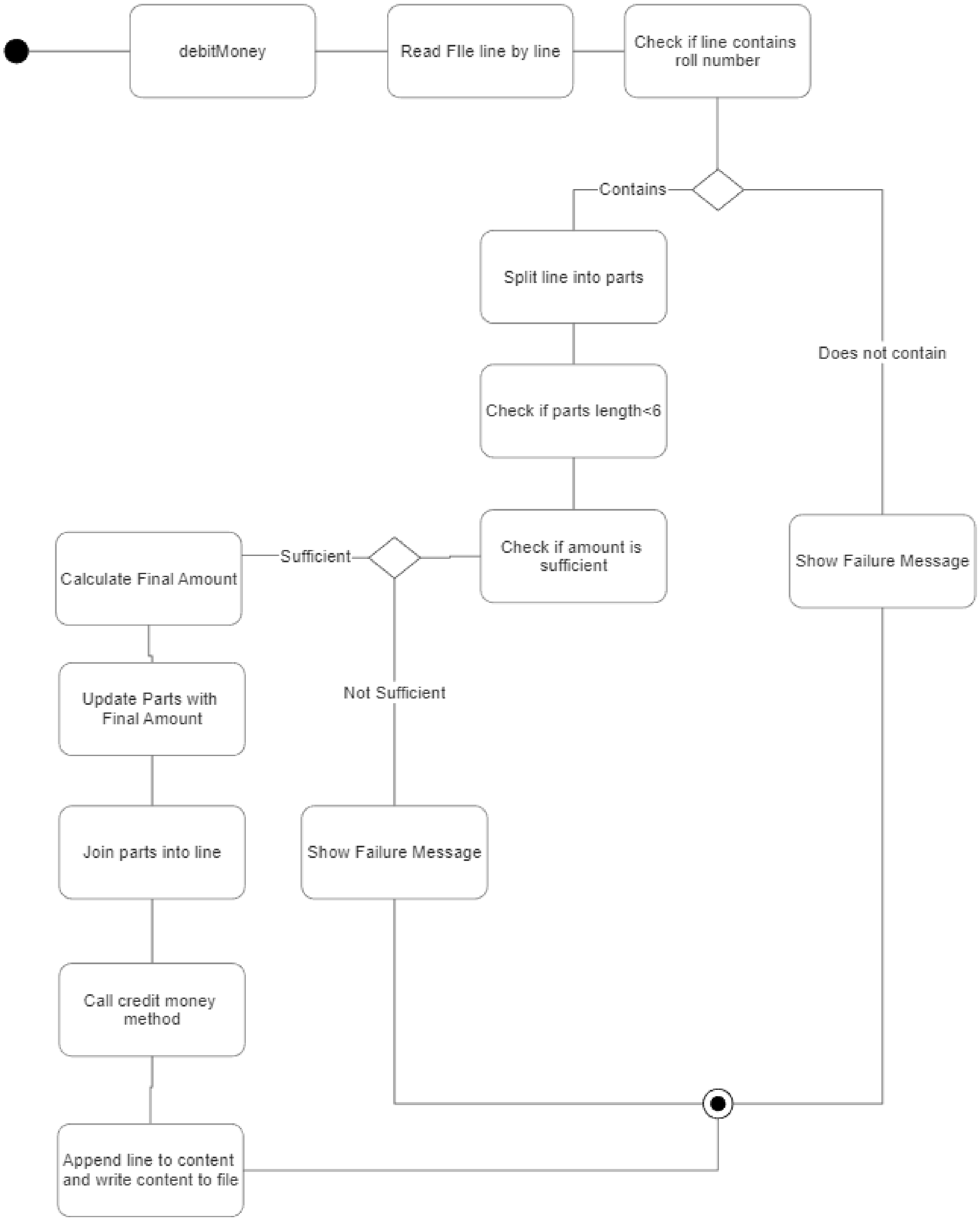
creditMoney:


creditMoney
Read File line by line
Check if Line Contains Balance
Contains
Does not contain
Update Balance with credit
Append updates line to content
Write content to file

addMoney:

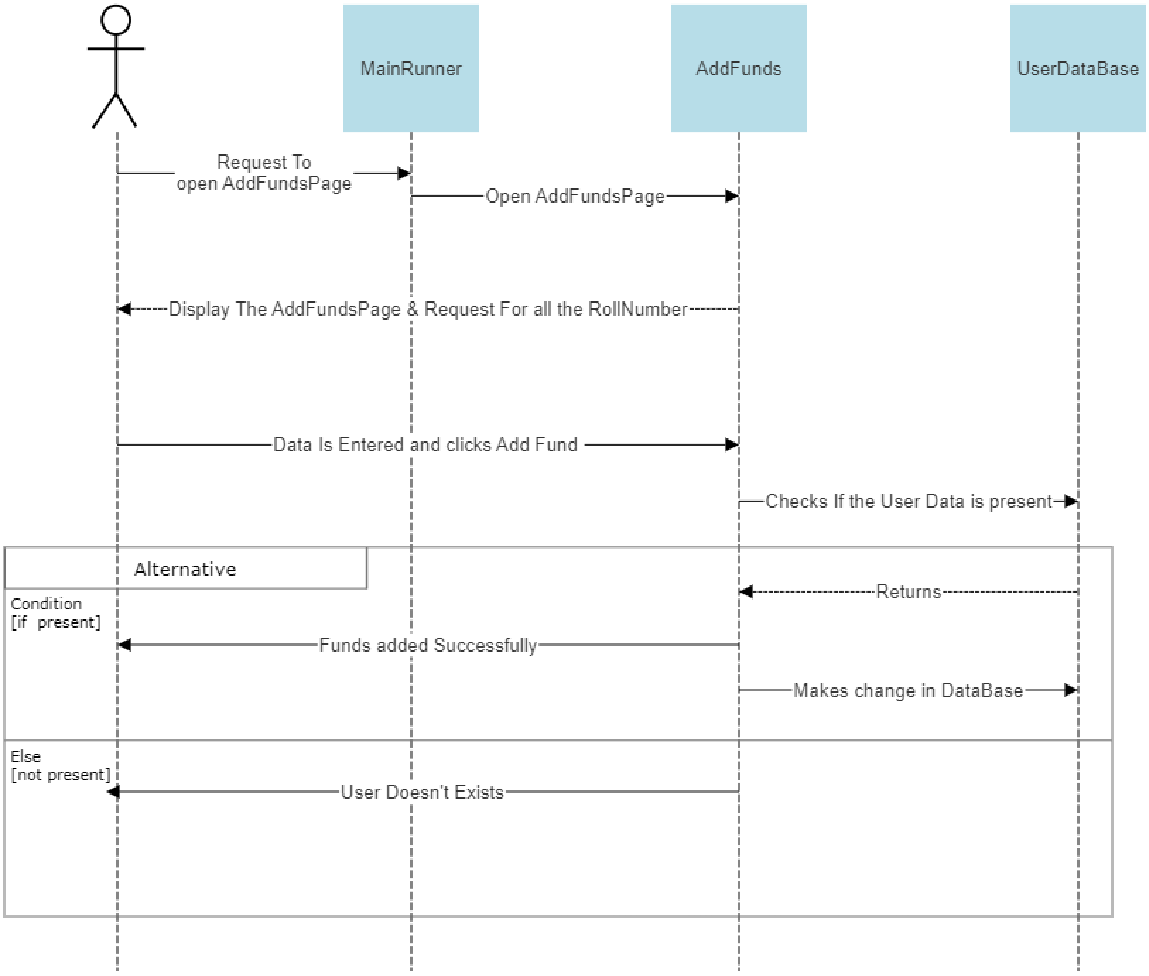

addMoney
Read FIle line by line
Check if line contains roll number
Contains
Does not contain
Split line into parts
Check if parts length<6
Add amount to line
Join parts into line
Append line to content and write content to file
Show success message
Show Failure Message

debitMoney:


debitMoney
Read FIle line by line
Check if line contains roll number
Contains
Does not contain
Split line into parts
Check if parts length<6
Check if amount is sufficient
Show Failure Message
Sufficient
Not Sufficient
Calculate Final Amount
Update Parts with Final Amount
Join parts into line
Call credit money method
Append line to content and write content to file
Show Failure Message

Sequence diagrams of the module:

Add Fund sequence:


MainRunner
AddFunds
UserDataBase
Request To open AddFundsPage
Open AddFundsPage
Display The AddFundsPage & Request For all the RollNumber
Data Is Entered and clicks Add Fund
Checks If the User Data is present
Alternative
Returns
Condition [if present]
Funds added Successfully
Makes change in DataBase
Else [not present]
User Doesn't Exists