ID: Add Income Transaction

Scope: Personal Budgeting App

Level: User Goal

Stakeholders and Interests:

User

- person who is adding an income transaction to their budget.

Precondition: User has an income that they would like to add to their budget.

Postcondition: User's budget reflects this income and can calculate accordingly.

Main Success Scenario:

1. User wants to add a new income to their budget
2. User selects "add income" from the interface
3. User adds a title to this transaction
4. User enters the amount of income
5. User enters the date of the transaction
6. User selects whether this is a recurring income or not
7. User selects transaction category from existing categories
8. User saves the new transaction to the system

Extensions:

a.* Anytime the system doesn't respond

User will restart the application

b.* Anytime the user decides not to save their changes

User will cancel the new transaction creation process

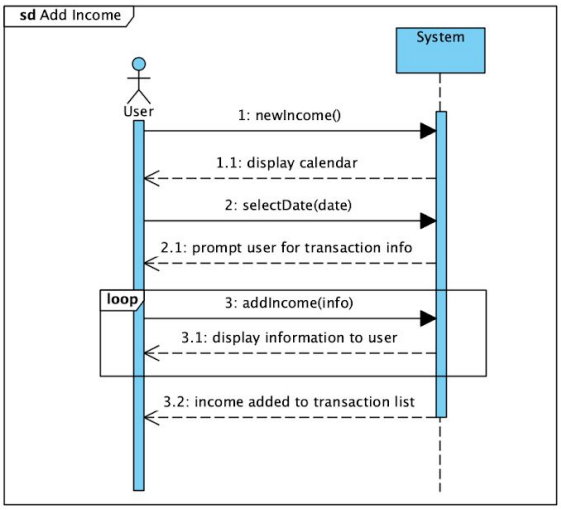
6.a If the user selects a recurring payment

1. *The system will prompt for the recurrence interval for this payment*

7.a If the desired category does not exist

1. *The user should cancel the transaction*

2. *The user should then create a new transaction category*
3. *The user then should return to create their transaction and will be able to select the new category*



Contract: New Income
Operation: newIncome()
Cross-References: Use Cases: Add Income
Pre-condition: User wants to make a new income transaction.
Post-condition: The system has created a new income from the user's specifications.

Contract: Select Date
Operation: selectDate(Date: date)
Cross-References: Use Cases: Add Income, Add Expense
Pre-condition: User wants to create a transaction and the system has prompted the user for the date
Post-condition: The date of transaction has been selected and associated with the transaction

Contract: Add Income Info
Operation: addIncome(String: info)
Cross-References: Use Cases: Add Income
Pre-condition: User is creating a new income transaction and has been prompted by the system to enter its information.
Post-condition: All of the information has been entered into the transaction

ID: Remove Income Transaction

Scope: Personal Budgeting App

Level: User Goal

Stakeholders and Interests:

User

- person who is removing an income transaction from their budget.

Precondition: User has an income that they would like to remove from their budget.

Postcondition: User's budget no longer reflects this income and can calculate accordingly.

Main Success Scenario:

1. User wants to remove an income from their budget
2. User selects "remove income"
3. User selects which income to remove from the list of existing incomes
4. User saves their changes
5. System removes the income from their budget

Extensions:

a.* Anytime the system doesn't respond

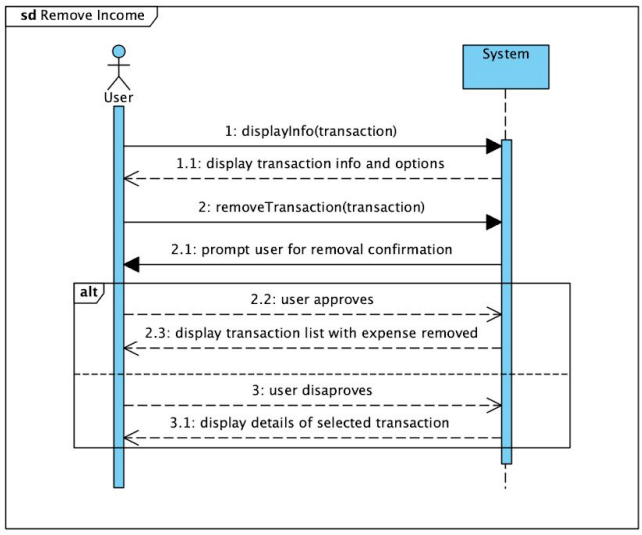
1. *User will restart the application*

b.* Anytime the user decides not to save their changes

1. *User will cancel the transaction removal process*

3.a The income that they wish to remove does not exist

1. *The user will cancel the transaction removal process*



Contract: Display Info
Operation: displayInfo(Transaction: transaction)
Cross-References: Use Cases: Remove Income, Remove Expense
Pre-condition: User wants to see information and settings for a given transaction
Post-condition: Displays information and settings for a given transaction

Contract: Remove Transaction
Operation: removeTransaction(Transaction: transaction)
Cross-References: Use Cases: Remove Income, Remove Expense
Pre-condition: User wants to remove a transaction from their list of transactions
Post-condition: The system has removed the given transaction from the list of transactions

ID: Create Account

Scope: Personal Budgeting App

Level: User Goal

Stakeholders and Interests:

User

- person who wishes to begin tracking their budget using our system.

Precondition: User wants to begin tracking their budget using our system.

Postcondition: User has created a new account and can now begin tracking their budget

Main Success Scenario:

1. User wants to create a new account
2. User selects “create new account”
3. System prompts user for a username and password
4. User enters their desired username and password
5. User selects “done”
6. The system creates the account

Extensions:

a.* Anytime the system doesn’t respond

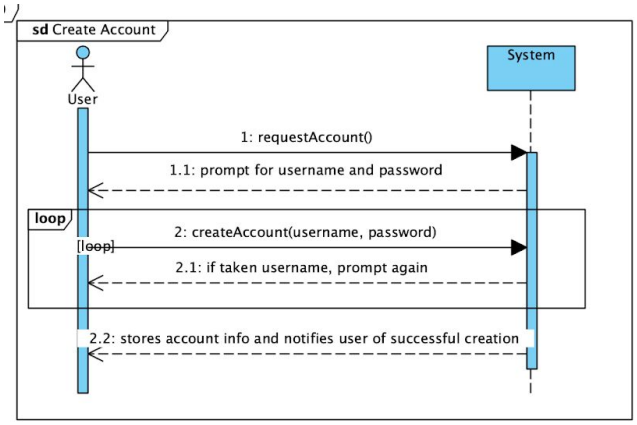
1. *User will restart the application*

1.a User already has an account

User selects “sign in” instead of “create new account”

5.* The given username is taken

1. *System informs the user that the provided username is already taken*
2. *System prompts user again for a new username*



Contract: Request Account
Operation: requestAccount()
Cross-References: Use Cases: Create Account
Pre-condition: User would like to create a new account
Post-condition: User begins the account creation process

Contract: Create Account
Operation: createAccount(username: String, password: String)
Cross-References: Use Cases: Create Account
Pre-condition: User would like to create a new account with this username and password
Post-condition: System checks for availability of the username