Cash Stash

Financial Management Application

Team Members

Cameron Kubik – cmk0037@auburn.edu – Team Leader
Sean Ramuchak – str0005@auburn.edu – Designer
Dylan McCardle – dcm0033@auburn.edu – Programmer
Daniell Yancey – dby0001@auburn.edu – Requirements Engineer
Zachary Hayes – zgh0001@auburn.edu – Architect

Professor Yilmaz
Software Modeling and Design
March 19, 2018

Table of Contents

The financial planning application provides financial management utilities for customers via a mobile application interface. Customers should be able connect existing financial tools to the application which generates analysis and graphical interaction. The application will collaborate with Banking, Credit, and Investment institutions that hold accounts of our target customers. Banks and other institutions will process transactions and maintain accounts on their own; the application will access this information through externally. The customer can use the application for personal or institutional purposes with the same result. The application is for customers who have financial assets and liabilities in many places and need a single place to manage them.

The application will enable users to view bank accounts, credit accounts, and investment portfolios in a single point of entry which expedites and organizes the process of evaluating a financial position. Along with creating a single access point for all of a customer's financial utilities, the application provides functionality to programmatically create a budget based on evaluation of bank and income statements. After analyzing a bank statement required by the Budget component, the application will be able to create a spending tracker and cash flow statement. Customers will be able to set up weekly, monthly and yearly bill reminders. These will be utilized by a Calendar component that reminds customers of upcoming and overdue bills.

The application will be used by individuals and organizations alike. The user interface is provided for mobile phones and tablets as an application downloaded from a phones app store. The app can be used at home, the office, or on the go without sacrificing functionality.

The application provides a solution for the need to consolidate financial assets and liabilities into a single point of access. The ability to create and manage budgets is necessary for those living on a fixed income or anyone looking for a better way to manage their money. The application is intended for day-to-day use and can be utilized at any point in time. Customers' financial well-being changes every day, this application serves to keep users up to date with those changes.

The application will work by providing a user interface through phones and tablets. The application system will collaborate with Bank, Credit Union, and Investment Firm systems to compile financial information needed for the UX. Each collaborating system (Banks, Credit Unions, and Investment Firms) will hold and maintain individual accounts on the customers behalf but will allow access to the confidential data after login credentials have been supplied.

A user must have an account to use the system. To create an account, the person must enter a valid e- mail address and choose a secure password. Once an account has been created, the user must verify their account by clicking a link in an automated email sent by the system. Once the

account has been marked as verified, the user may choose to add additional information to their account such as a phone number, address, date of birth and social security number.

To link external services to an account (this could be a bank, credit card company, billing service), the user must choose a service from the list of supported ones and then provide the username and password associated with the account they wish to link. Once the user submits the data, an external services manager will attempt to verify the information. If the information was able to be verified then the users account data will be synced with their Cash Stash account; otherwise, the user will receive an error message detailing the problem.

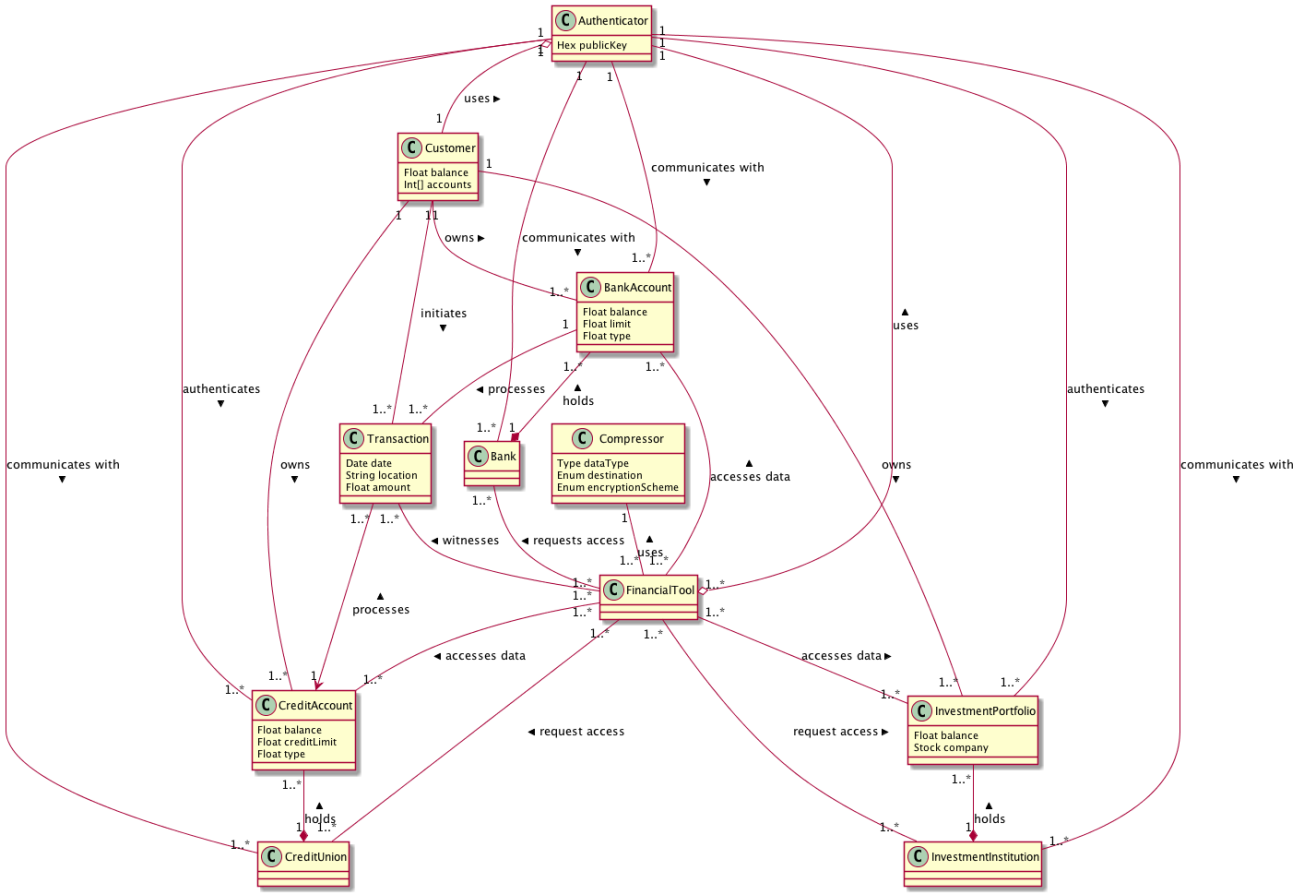
Users may also create transactions manually. To create an expense transaction the user would need to specify a name, date and recipient; similarly, to create an income transaction the user would need to specify a name, date and source of the income. The user may also specify if a transaction should be recurring; if so, the user would need to provide the next date that the transaction should take place as well as the frequency on which to perform the transaction. Other attributes such as a category, memo, transaction type and transaction number are optional.

One of the key features of the system allows the user to setup a monthly budget. To create a new budget the user will need to provide an estimate of their gross monthly or annual income. This will generate a suggested budget plan for the user with their monthly income broken down into categories by percentage. The user is free to change these values as well as add or remove categories to customize their budget.

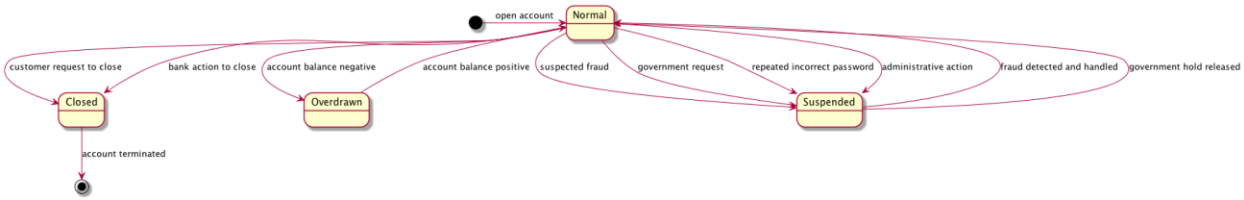
Another thing that the user may like to do is set up savings goals. When a user sets up a savings goal, a certain percentage of their monthly income will be set aside to ensure that they are able to meet their goal. To set this up a user would choose a name for the goal, the amount that they wish to save and the date by which they would need the savings. If a user has a budget setup, then these savings will be deducted from the amount of “free money” set aside for that month. If the user does not have a budget setup, then the user may manually update the value of their savings.

The user can also generate financial reports that provide various ways of summarizing their financial activity. To generate a financial report the user must have entered at least one transaction into the system. Some reports may require additional input from the user such as defining a period for the summary. Like reports, a user can choose to track their credit score through their Cash Stash account. To set up credit tracking, the user must have a complete profile (meaning they have added their address, phone number, social security number, etc.)

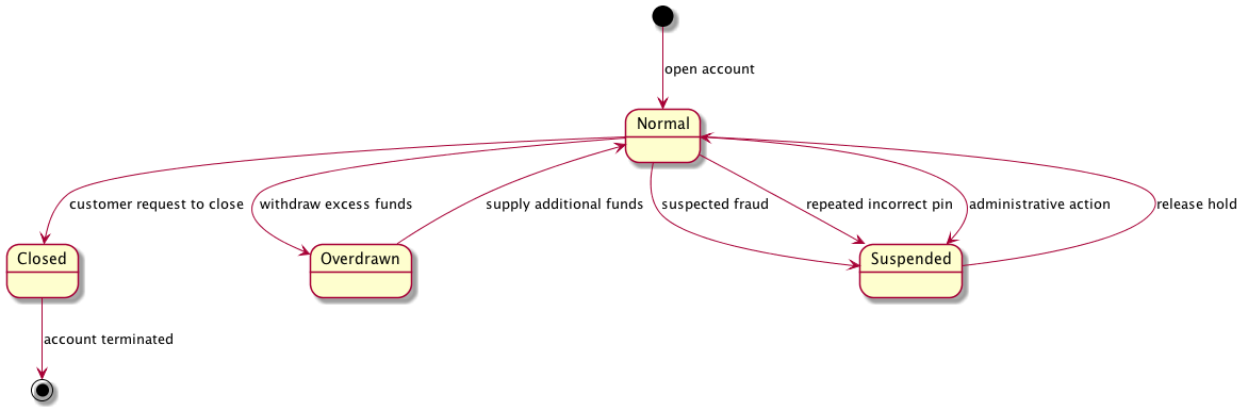
Conceptual Domain Model



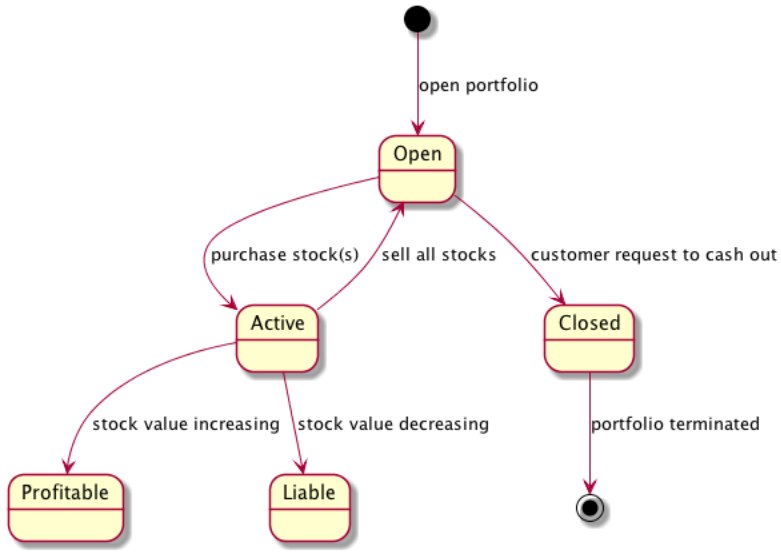
Bank Account State Model



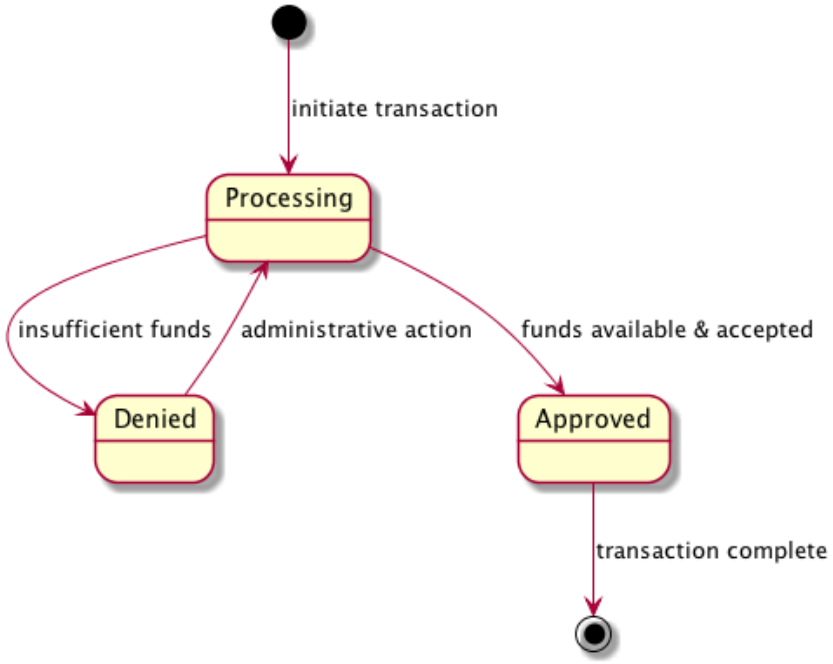
Credit Account State Model



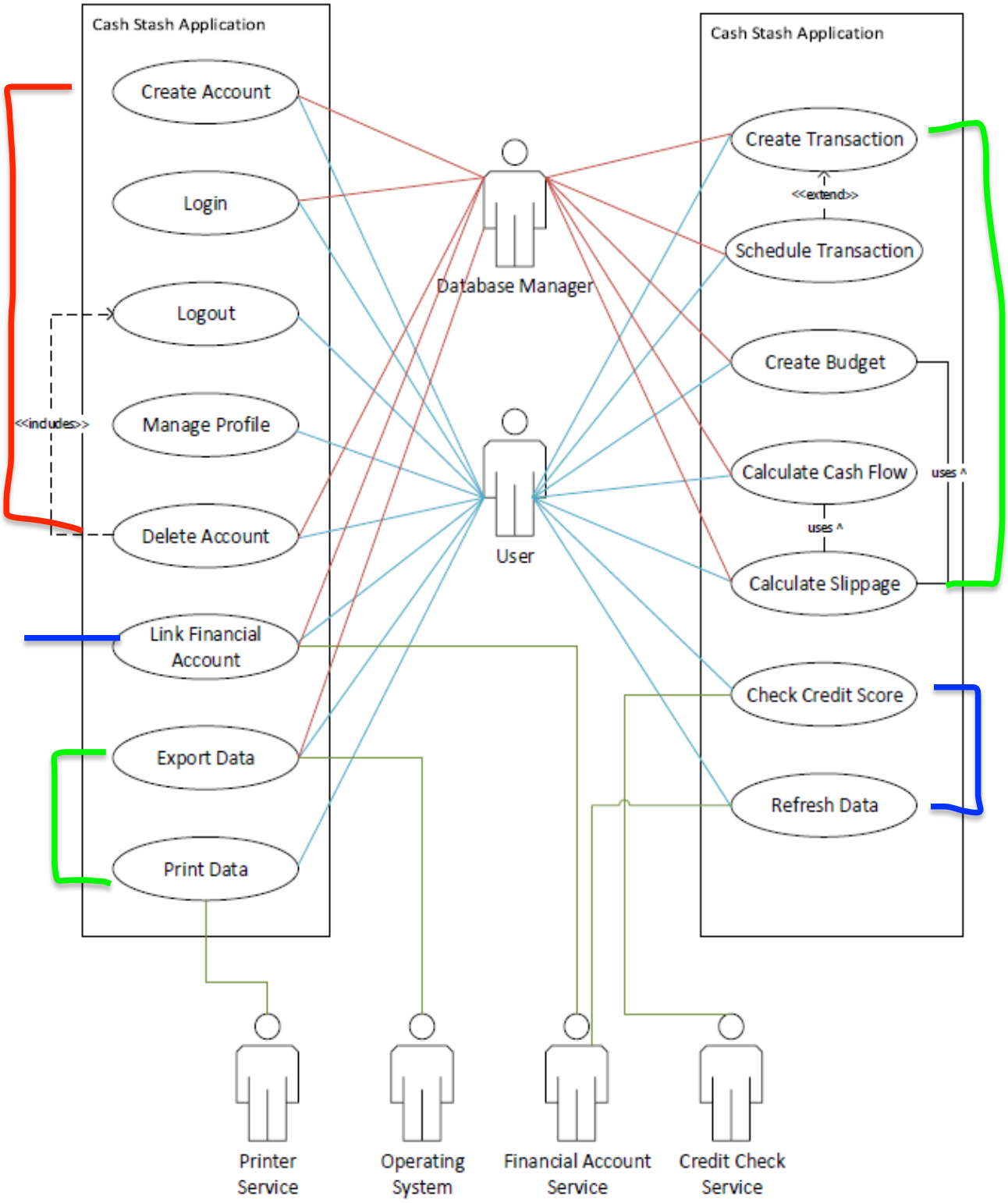
Investment Portfolio State Model



Transaction State Model



Use Case Diagram



Essential Use Cases

Create Account
Pre-condition: The user is on the Cash Stash login screen.
Actors: User (Primary), System (Secondary)

1. User provides account information.
2. System validates user information.
3. System creates account.

Exceptions:

3a. If the account information couldn't be validated, then the user will receive an error message.
3b. If the password provided is less than 7 characters, then the user will receive an error message.

Post-condition: The user account is created.

Login
Pre-condition: The user is on the Cash Stash login screen and has already created an account.
Actors: User (Primary), System (Secondary)

1. User provides login information.
2. System validates user information.
3. System logs the user in.

Exceptions:

2a. If the account information couldn't be verified, then the user will receive an error message.

Post-condition: The user is logged in.

Logout
Pre-condition: The user is logged in to Cash Stash.
Actors: User (Primary), System (Secondary)

1. User chooses the logout option.
2. System saves any open data.
3. System logs the user out and displays login screen.

Exceptions:

Post-condition: The user is logged out.

Manage Profile
Pre-condition: The user is logged in to Cash Stash.
Actors: User (Primary), System (Secondary)

1. User chooses the manage profile option.
2. System loads the account settings screen.
3. User submits account settings.
4. System saves account information and loads the main screen.

Exceptions:
4a. If the account settings are missing information, an error message is displayed.

Post-condition: The user's account information is updated.

Delete Account
Pre-condition: The user is on the "Account Settings" screen of the Cash Stash application.
Actors: User (Primary), System (Secondary)

1. User chooses the delete account option.
2. System prompts the user for their account password.
3. User submits account password.
4. System erases all data related to the user's account and displays the login screen.

Exceptions:
4a. If the user fails the account verification step, an error message will display.

Post-condition: The user's account and data no longer exist.

Link Financial Account
Pre-condition: The user is on the "Account Settings" screen of the Cash Stash application.
Actors: User (Primary), System (Secondary), Financial Account Service (Secondary)

1. User chooses the link financial account option.
2. System prompts the user for the account details.
3. User provides financial account details.
4. System sends account details to financial account service for authorization.
5. Financial account service authorizes the account details.
6. System syncs financial account information with Cash Stash account.

Exceptions:
4a. If there is information missing from the account details, the system will prompt the user to finish filling in the details.
6a. If the account details were not authorized, then the system will display an error message to the user and ask them to try again.
Post-condition: The user's Cash Stash account is now synced with their financial account.

Create Transaction
Pre-condition: The user is logged in to Cash Stash.
Actors: User (Primary), System (Secondary)

1. User chooses the create transaction option.
2. System displays new transaction screen.
3. User submits transaction details.
4. System verifies transaction details.
5. System saves the transaction.

Exceptions:

5a. If the transaction verification fails, the system will display an error message instead of saving the transaction.

Post-condition: The transaction is saved within the user's account.

Schedule Transaction
Pre-condition: The user is logged in to Cash Stash.
Actors: User (Primary), System (Secondary)

1. User chooses the schedule transaction option.
2. System displays new transaction screen.
3. User submits transaction details.
4. System verifies transaction details.
5. System saves the recurring transaction.

Exceptions:

5a. If the transaction verification fails, the system will display an error message instead of saving the recurring transaction.

Post-condition: The recurring transaction is saved within the user's account.

Create Budget
Pre-condition: The user is logged in to Cash Stash.
Actors: User (Primary), System (Secondary)

1. User chooses the create budget option.
2. System displays the new budget screen.
3. User submits budget settings.
4. System verifies budget settings.
5. System displays the budget report.

Exceptions:

5a. If the budget settings are missing information, the system will display an error message instead of creating the budget.
5b. If the sum of percentages for categories is less than 0 or greater than 100, the system will display an error message.

Post-condition: The budget report is displayed.

Calculate Cash Flow

Pre-condition: The user is logged in to Cash Stash.

Actors: User (Primary), System (Secondary)

1. User chooses the calculate cash flow option.
2. System prompts the user for a timeframe.
3. User chooses a timeframe for the cash flow.
4. System calculates income and expense transactions for given timeframe.
5. System displays cash flow report.

Exceptions:

4a. If the account doesn't contain any transactions within the given timeframe, the system will display an error message.

Post-condition: The cash flow report is displayed.

Calculate Slippage

Pre-condition: The user is logged in to Cash Stash and has setup a budget.

Actors: User (Primary), System (Secondary)

1. User chooses the calculate slippage option.
2. System prompts the user to choose a timeframe.
3. User chooses the timeframe for which to calculate slippage.
4. System loads budget and calculates cash flow.
5. System displays a slippage report which compares values from the budget to the actual cash flow.

Exceptions:

Post-condition: The slippage report is displayed.

Check Credit Score
Pre-condition: The user is logged in to Cash Stash and has completed their account identity information.
Actors: User (Primary), System (Secondary), Credit Check Service (Secondary)

1. The user chooses the check credit score option.
2. System will query the credit check service for the user's credit score.
3. Credit check service verifies the identity information and gets the credit score.
4. System displays the user's current credit score.

Exceptions:

4a. If the credit check service returns an error, the error message is displayed to the user instead.

Post-condition: The user's credit score is displayed.

Export Data
Pre-condition: The user is logged in to Cash Stash.
Actors: User (Primary), System (Secondary), Operating System (Secondary)

1. User chooses the export data option.
2. System prompts the user to choose a location to save the file.
3. User chooses a save location.
4. System will compile an excel file of the account transactions.
5. System sends a request to the operating system to save the excel file.

Exceptions:

4a. If the account doesn't contain any transactions, an error message will display to the user.

Post-condition: The operating system receives the create file request.

Print Data
Pre-condition: The user is logged in to Cash Stash.
Actors: User (Primary), System (Secondary), Print Service (Secondary)

1. User chooses the print data option.
2. System will send a request to the printer service to print the current screen.

Exceptions:

Post-condition: The printer service receives the print request.

Refresh Data

Pre-condition: The user is logged in to Cash Stash.

Actors: User (Primary), System (Secondary), Financial Account Service (Secondary)

1. User chooses the refresh data option.
2. System request a sync operation from the financial account service.
3. Financial account service will respond with the appropriate data.
4. System reloads the current screen.

Exceptions:

4a. If any of the sync requests returned with an error, these error messages will be displayed to the user before reloading the current screen.

Post-condition: The user's Cash Stash account now contains any data in linked financial accounts.

Scenarios

Create Account

John Doe opens the login screen.
John Doe provides login information.
Joh Doe chooses the create account option.
System verifies account availability.
System creates a new account.
System establishes a steady connection.
System displays main screen.

Login

John Doe opens the login screen.
Joh Doe provides login information.
John Doe chooses the login option.
System verifies login information.
System establishes a steady connection.
System displays main screen.

Logout

John Doe logs in.
John Doe chooses the logout option.
System saves any open data.
System terminates steady connection.
System displays login screen.

Manage Profile

John Doe logs in.
John Doe chooses the manage profile option.
System grabs account settings.
System displays manage profile screen.
John Doe updates profile settings.
John Doe chooses the save option.
System validates new account settings.
System saves account settings.
System displays main screen.

Delete Account

John Doe logs in.
John Doe opens manage profile.
John Doe chooses the delete account option.
System prompts John Doe for the account password.
John Doe enters the account password.
John Doe chooses the delete option.
System verifies the account password.
System erases all account data.
System displays the login screen.

Link Financial Account

John Doe logs in.
John Doe opens manage profile.
John Doe chooses the link financial account option.
System prompts the user to provide the login information for the account.
John Doe chooses the financial institution and provides the account login information.
System sends the login information to the financial account service to request a connection.
Financial account service verifies the account information.
System creates a transaction for each record found in the financial account.
System displays a success message.

Create Transaction

John Doe logs in.
John Doe chooses the create transaction option.
System displays the new transaction screen.
John Doe enters transaction information.
John Doe chooses the save option.
System verifies the transaction details.
System saves the transaction.
System displays the transaction log screen.

Schedule Transaction

John Doe logs in.
John Doe chooses the schedule transaction option.
System displays the schedule transaction screen.
John Doe enters the transaction information.
Joh Doe chooses the save option.
System verified the transaction details.
System sets up a recurring transaction.
System displays the main screen.

Generate Budget Report

John Doe logs in.
John Doe chooses the create budget option.
System displays the budget settings screen.
John Doe creates budget categories and assigns percentage or fixed amount values to each category.
John Doe chooses the save option.
System verifies the budget settings.
System saves the budget.
System displays the main screen.

Calculate Cash Flow

John Doe logs in.
John Doe chooses the calculate cash flow option.
System verifies that the account contains transactions.
System prompts for a timeframe to display the cash flow for.
John Doe enters a timeframe.
John Doe chooses the calculate option.
System verifies that the timeframe has transactions.
System calculates cash flow.
System displays the cash flow report.

Print Data

Joh Doe logs in.
John Doe chooses the print data option.
System sends a request to the printer service to print the cash flow screen.

Refresh Data

John Doe logs in.
John Doe chooses the refresh data option.
System sends a request to each linked financial account to perform a data sync.
System reloads the current screen.

Calculate Slippage

John Doe logs in.
John Doe chooses the calculate slippage option.
System verifies that the account contains transactions.
System prompts for a month and year to display slippage for.
John Doe chooses a month and year.
Joh Doe chooses the calculate option.
System verifies that there are transactions available for the given month and year.
System calculates slippage.
System displays slippage report.

Check Credit Score

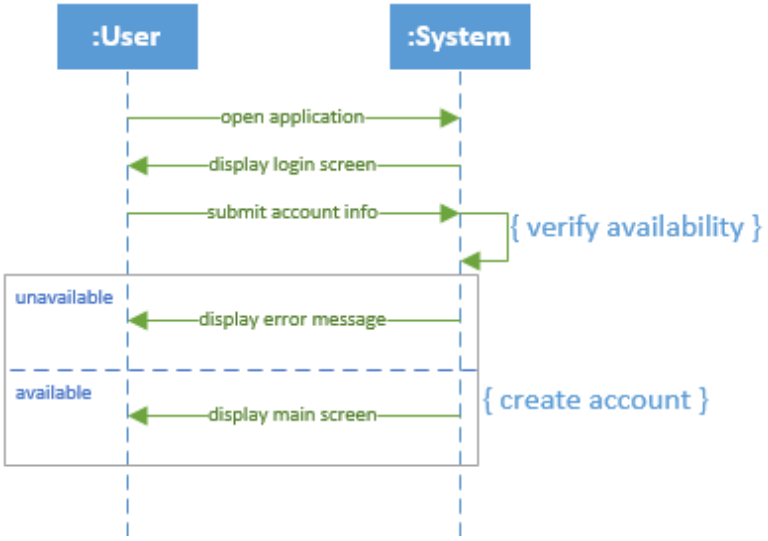
Joh Doe logs in.
John Doe chooses the check credit score option.
System grabs account information.
System queries the credit check service with identity information from account.
Credit check service verifies account information and retrieves a credit score estimate.
System displays the credit score.

Export Data

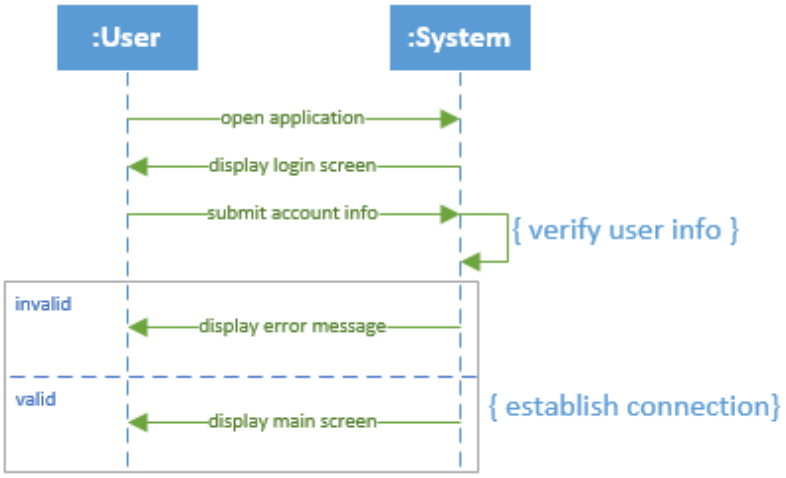
John Doe logs in.
Joh Doe chooses the export data option.
System prompts the user for a save location.
Joh Doe chooses a save location.
System will compile an excel document containing all transaction data saved in the account.
System sends a request to the operating system to save the excel file.
Operating System saves the excel file.

High-Level SSD

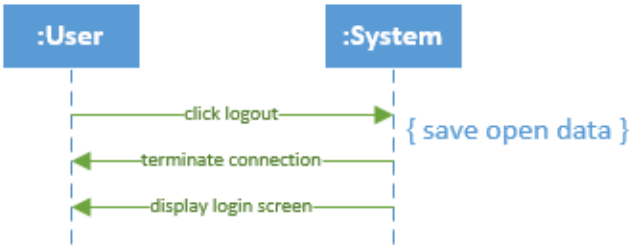
Create Account



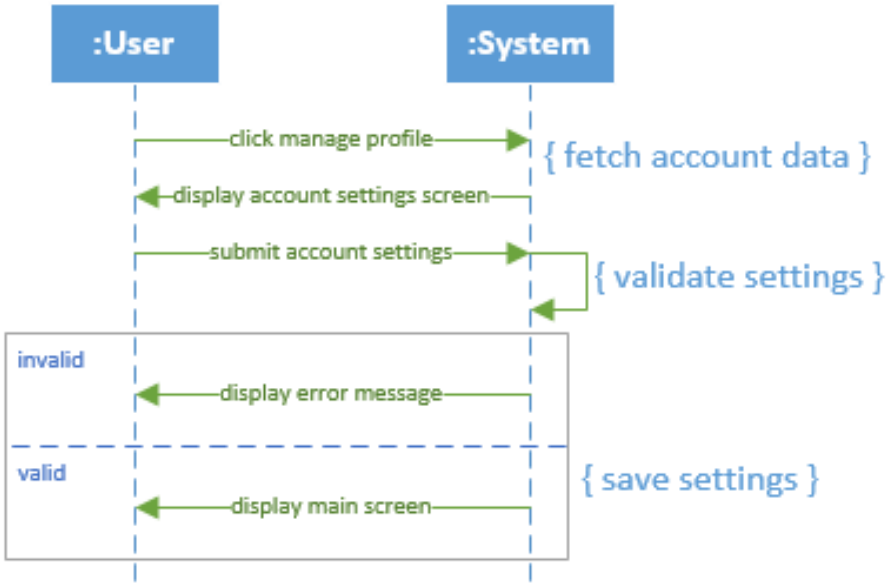
Login



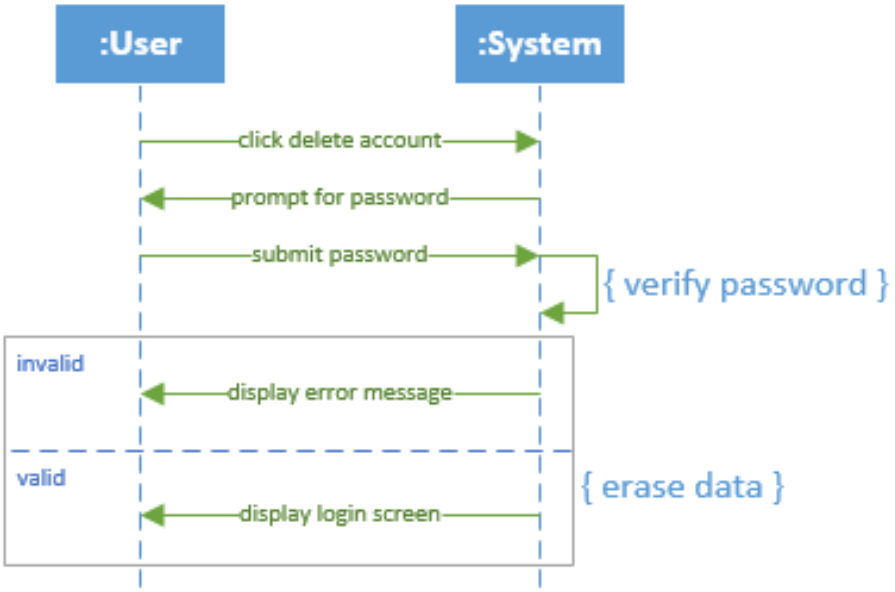
Logout



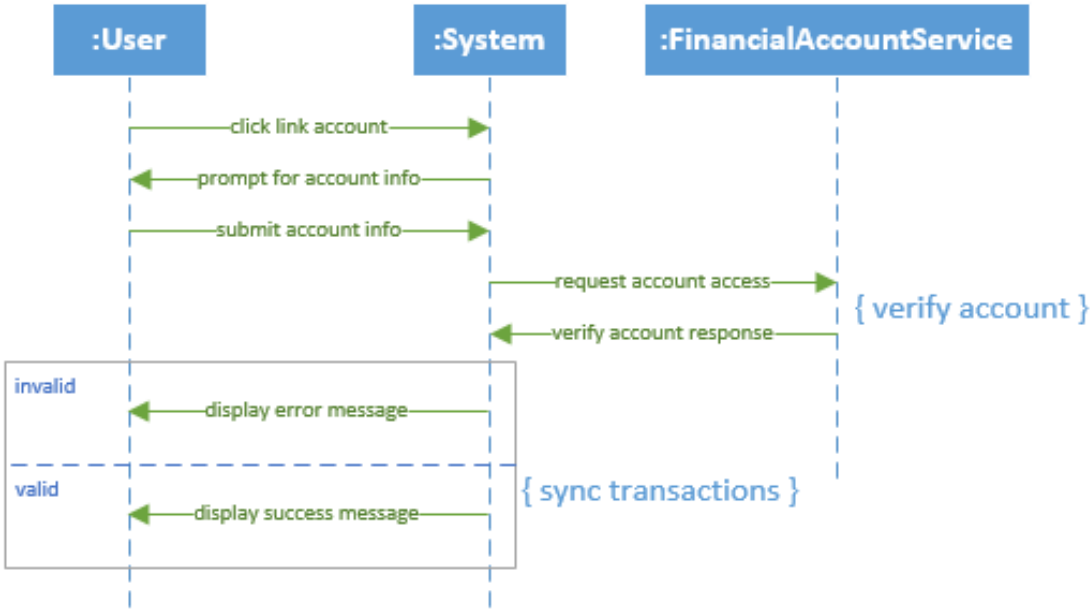
Manage Profile



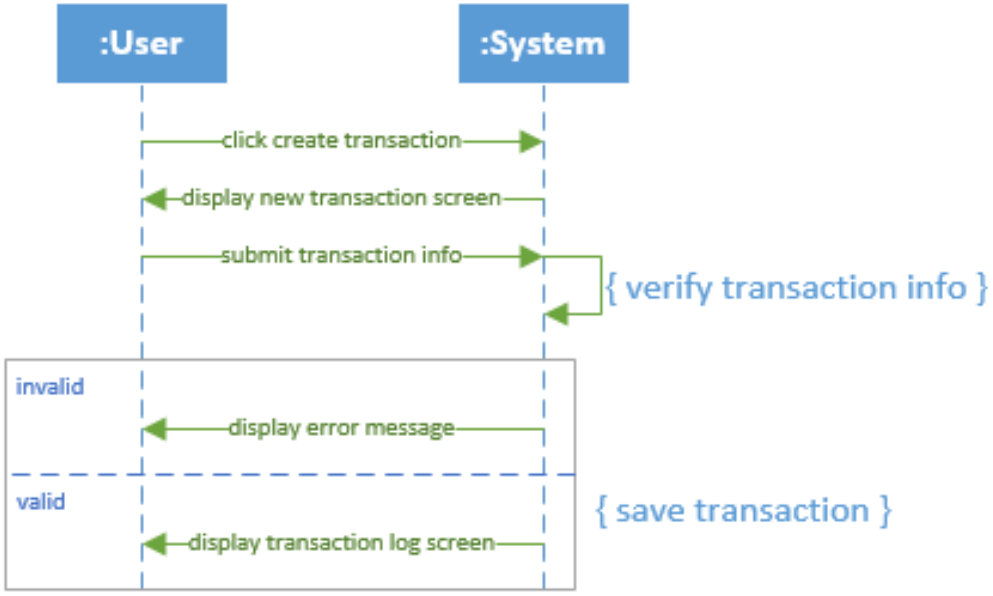
Delete Account



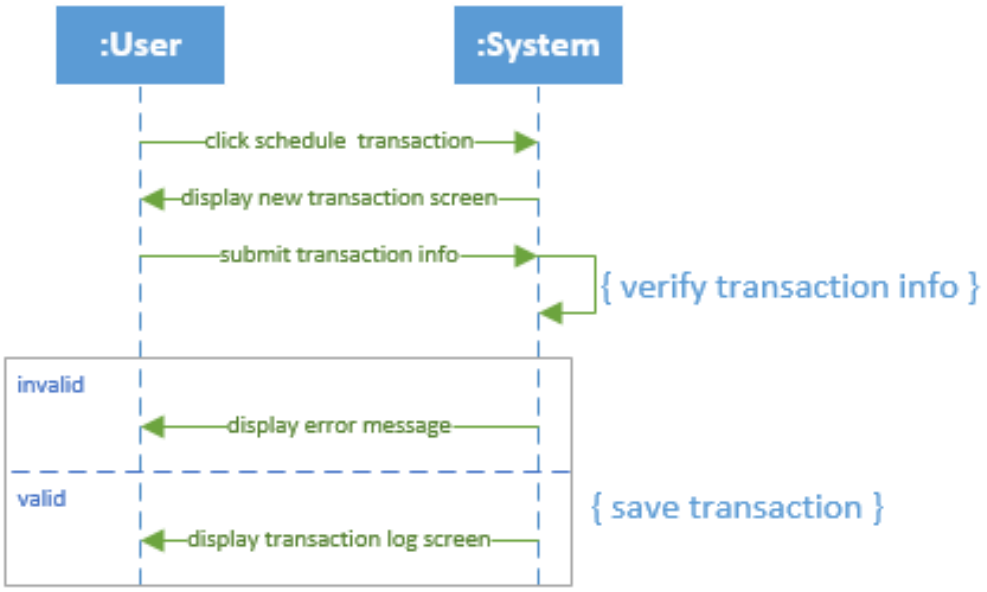
Link Financial Account



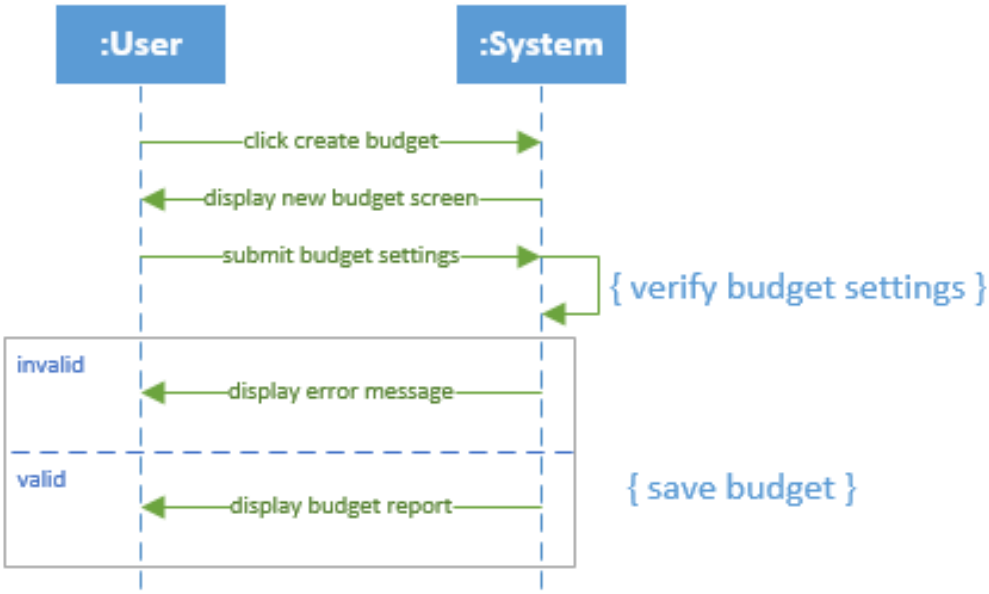
Create Transaction



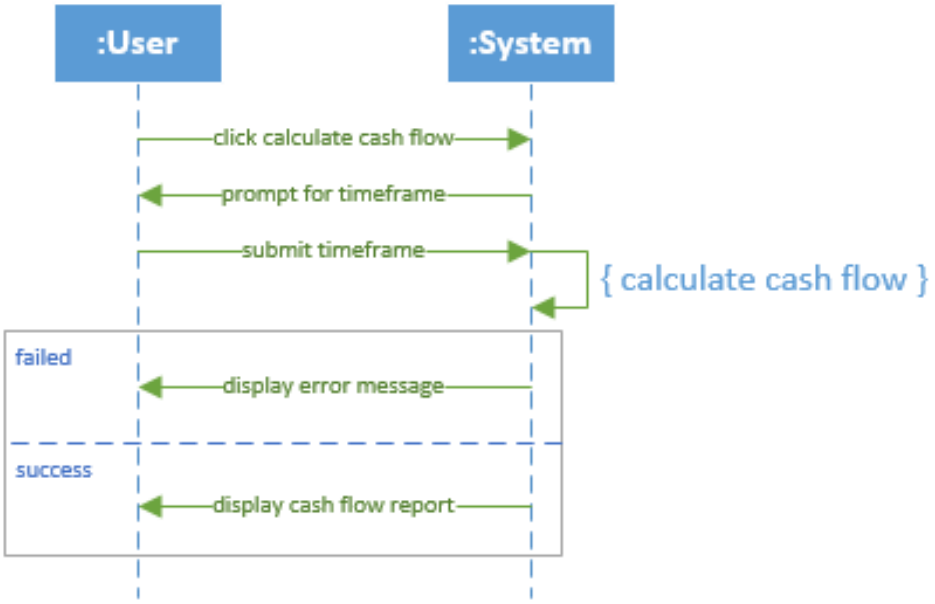
Schedule Transaction



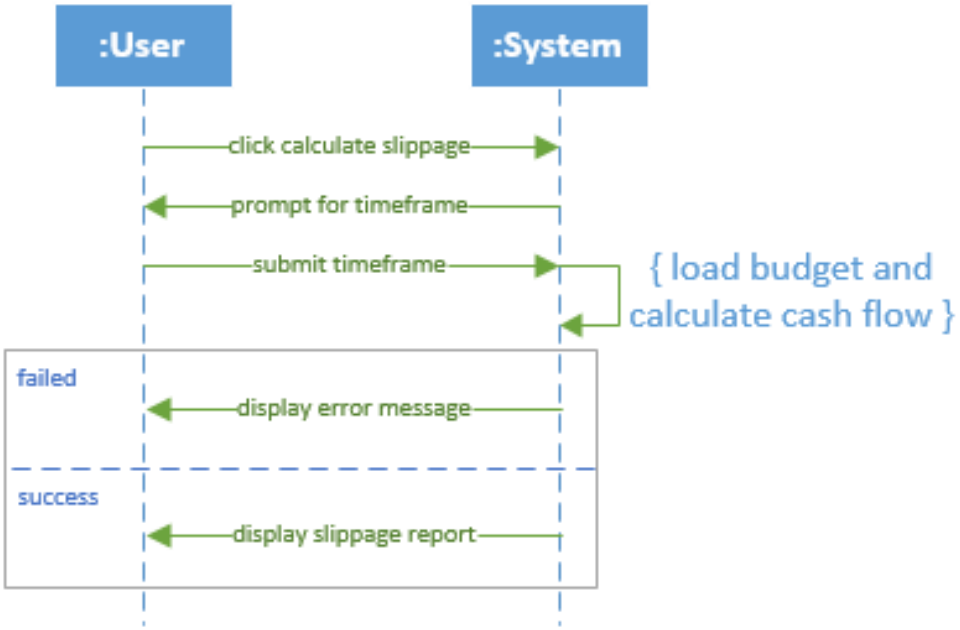
Create Budget Report



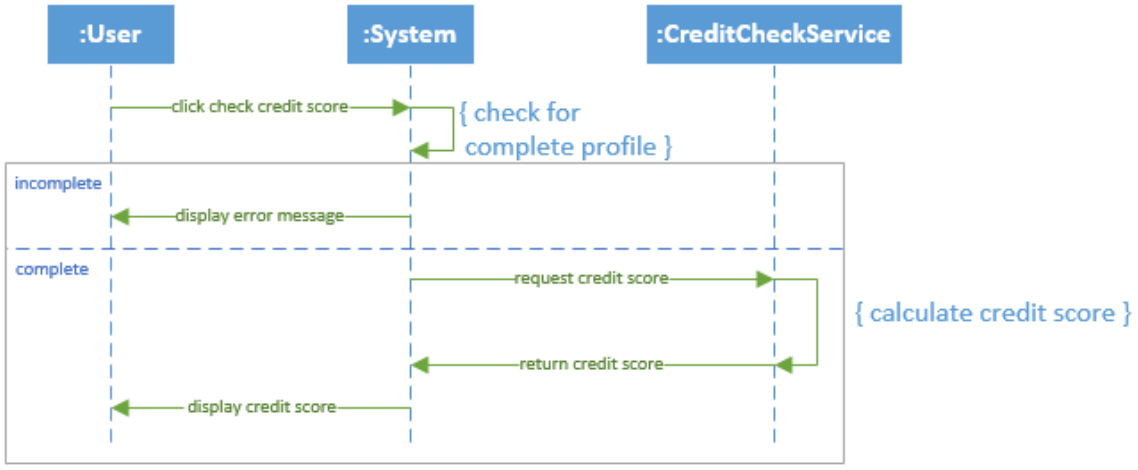
Calculate Cash Flow



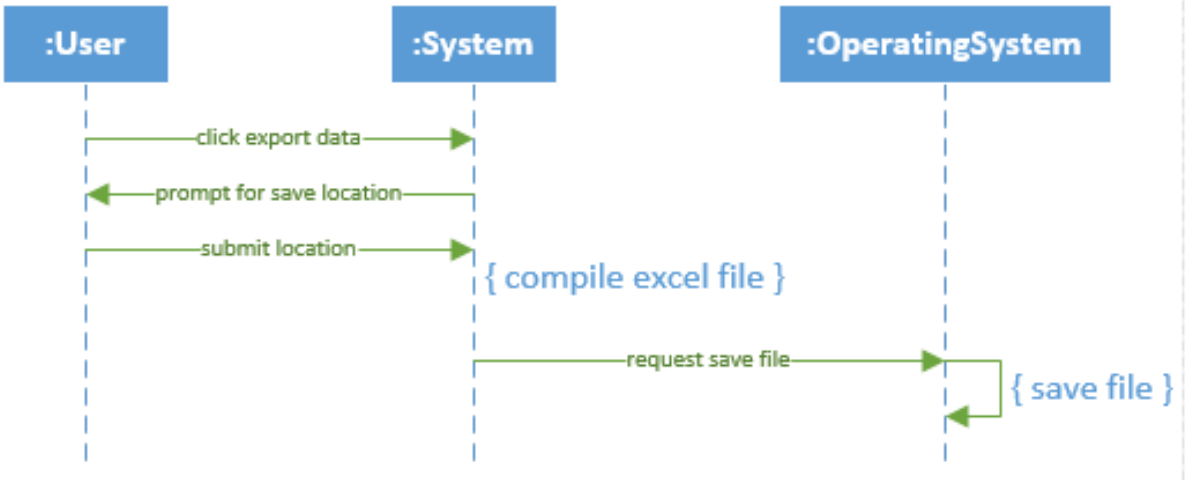
Calculate Slippage



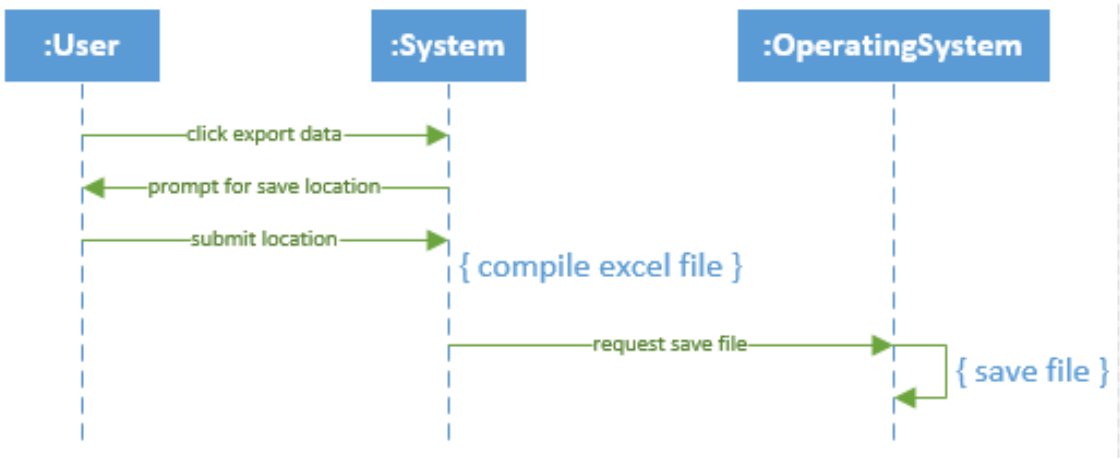
Check Credit Score



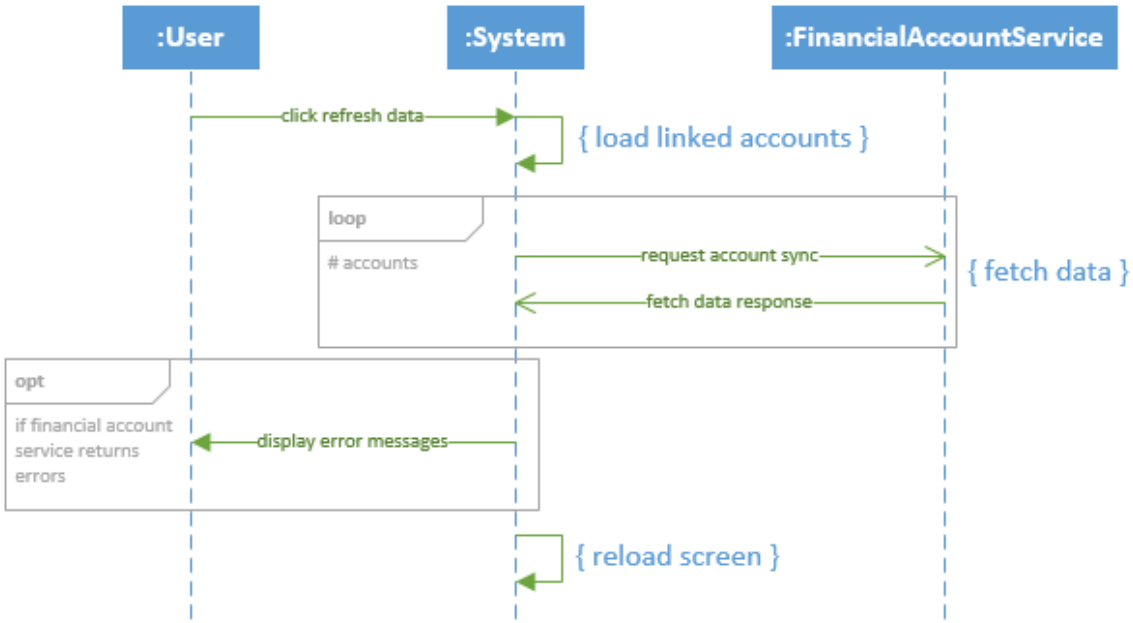
Export Data



Print Data



Refresh Data



Concrete Use Cases

Create Account

Pre-condition: The user is on the application login screen.
Event Flow:

1. The user fills the login form with the email address and password they wish to sign up with and then clicks the “Create Account” button.
2. The system validates the account. Meaning that it checks to make sure the email address provided does not belong to an existing account.
3. The system creates a new account in the database using the provided email address and password and the user is logged in to the application.
    a. If an account was found with the matching email address, then an error message displays informing the user that they must use a different email address.
    b. If the password entered is less than 7 characters, then an error message displays informing the user that they must choose a stronger password.

Post-condition: The user account is created and logged in.

Login

Pre-condition: The user is on the application login screen and has already created an account.
Event Flow:

1. The user fills the login with the email address and password corresponding to their account.
2. The system performs account verification by searching the database for an account with the matching email address and password combination.
    a. If no such account could be found, then an error message displays informing the user that the provided email address and password do not match an account.
3. The system logs the user in using the account found in account verification.

Post-condition: The user account is logged in.

Logout

Pre-condition: The user is logged in to the application.
Event Flow:

1. The user clicks the “Logout” button.
2. The system checks the current screen for any unsaved data and attempts to save it, then the system logs the user out and displays the login screen.

Post-condition: The user is logged out of the application.

Manage Profile

Pre-condition: The user is logged in to the application.
Event Flow:

1. The user clicks the “Manage Profile” button.
2. The system displays the “Account Settings” screen with a form containing account information populated from the database.
3. From this screen, the user can choose to edit their email, password, identity information, link and unlink financial accounts, create alerts and delete their account. When the user finishes making changes, they click the “Save” button.
4. The system saves any changes made to the account in the database and displays the main screen.
    a. If the account settings are missing information, then an error message is displayed informing the user of which fields need to be corrected.

Post-condition: The user’s account information is updated.

Delete Account

Pre-condition: The user is logged in and has loaded the “Account Settings” screen.
Event Flow:

1. The user clicks on the “Delete Account” button.
2. The system displays a prompt for the user to enter their password as a confirmation that they wish to delete all their account data from the application database.
3. The user enters their account password and clicks the “Delete” button.
4. The system erases all the data associated with the user’s account and displays the application login screen.
    a. If the supplied password does not match the one for the account, an error message displays informing the user than they entered an incorrect password.

Post-condition: The user’s account data no longer exists.

Link Financial Account

Pre-condition: The user is logged in and has loaded the “Account Settings” screen.
Event Flow:

1. The user clicks on the “Link Financial Account” button.
2. The system displays a prompt for the user to choose a supported financial institution from a dropdown list and then provide the username and password associated with the account they wish to link.
3. The user chooses a financial institution, enters their username and password and clicks the “Continue” button.
4. Based on the financial institution selected by the user, the system sends a request to the financial account service to access an account that matches the provided username and password.
    a. If the user did not select an institution or provide a username and password, then an error message displays asking the user to correct those issues.
5. The financial account service receives the request and looks for an account with a matching username and password. If an account is found, then the service responds with authorization to access the account data. If the account could not be found, the service responds with an error message.
6. The system establishes a connection to the financial account through the financial account service and begin downloading a history of transaction data.

   a. If the financial account service returned an error, this error message is displayed back to the user instead of establishing a connection.

Post-condition: The financial account is now linked to the user's Cash Stash account.

Create Transaction

Pre-condition: The user is logged in to the application.
Event Flow:

1. The user clicks on the "Create Transaction" button.
2. The system displays the "New Transaction" screen. This screen contains a form with fields for the user to enter transaction information. Required fields include a title, date of transaction, amount and type of exchange (cash, check, credit, etc.). Optional fields include a category, memo and transaction number (check #, etc.).
3. When the user finishes filling out the form, they click the "Save" button.
4. The system verifies that all the required information for creating a transaction has been filled out properly.
5. The system creates a new transaction with the details provided by the user and displays the "Transaction Log" screen.
   a. If transaction detail verification fails, then an error message is displayed informing the user of which fields need to be corrected before attempting to save the transaction.

Post-condition: The transaction is saved in the user's account.

Schedule Transaction

Pre-condition: The user is logged in to the application.
Event Flow:

1. The user clicks on the "Schedule Transaction" button.
2. The system displays the "New Transaction" screen. This screen contains a form with fields for the user to enter details about the recurring transaction. Required fields include a title, date of transaction, amount, type of exchange (cash, check, credit, etc.), the next date the transaction should occur and the frequency at which to keep performing the transaction. Optional fields include a category, memo and transaction number.
3. When the user is finished entering the transaction details, they click the "Save" button.
4. The system verifies that all the required information for creating a recurring transaction has been filled out properly.
5. The system creates a new recurring transaction with the details provided by the user and displays the "Transaction Log" screen.
   a. If transaction detail verification fails, then an error message displays informing the user of which fields need to be corrected before attempting to save the transaction.

Post-condition: The recurring transaction is saved within the user's account.

Create Budget

Pre-condition: The user is logged in to the application.
Event Flow:

1. The user clicks the “Create Budget” button.
2. The system displays the “New Budget” screen. On this screen the user is required to provide their gross monthly income and define categories to be used in the budget. For each category the user also needs to provide a percentage or fixed amount of income they wish to set aside for it.
3. The user fills out the required fields in the budget report form. Once the user has finished setting up their budget, they click the “Save” button.
4. The system verifies that the user has filled out all the required fields in the “New Budget” screen.
5. Based on the settings chosen by the user, the system calculates how much of the user’s gross monthly income should be set aside for each category in the budget. The system then displays the budget report.
    a. If the verification step finds missing information, an error message displays that informs the user of which fields need to be corrected.
    b. If the verification step find that the sum of percentages defined for categories is less than 0 or greater than 100, then an error message displays informing the user to correct the percentage values.

Post-condition: A budget report is displayed based on settings chosen by the user.

Calculate Cash Flow

Pre-condition: The user is logged in to the application.
Event Flow:

1. The user clicks the “Calculate Cash Flow” button.
2. The system displays a prompt for the user to choose a month and year as a timeframe for generating the report.
3. The user chooses a month and a year from the provided lists and clicks the “Continue” button.
4. The system calculates the total income and expenses for transactions found within the timeframe chosen by the user. These totals are also broken down by category based on the categories defined within each transaction item. If a transaction item does not contain a category, it is assigned to the category “Other”.
    a. If no transactions were found within the chosen timeframe, an error message displays informing the user that the system couldn’t find any transactions.
5. The system displays a cash flow report that summarizes the details previously calculated.

Post-condition: A cash flow report is displayed based on settings chosen by the user.

Calculate Slippage

Pre-condition: The user is logged in to the application and has setup a budget.
Event Flow:

1. The user clicks the “Calculate Slippage” button.
2. The system displays a prompt for the user to choose a month and year as a timeframe for calculating slippage.
3. The user chooses a month and year from the provided lists and clicks the “Continue” button.
4. The system loads the budget associated with the user’s account and calculates the cash flow for the chosen timeframe.
5. The system compares the total values from the budget to the cash flow broken down by category. This shows the user how close they stayed to the goals setup in their budget. These results are displayed as a slippage report.

Post-condition: A slippage report is displayed based on settings chosen by the user.

Check Credit Score

Pre-condition: The user is logged in to the application and has completed their account identity information.
Event Flow:

1. The user clicks the “Check Credit Score” button.
2. The system loads the user’s identity information from their account and sends this to the credit check service, asking for a credit score estimate in return.
3. The credit check service verifies the identity and performs a credit score lookup.
4. The system displays the user’s credit score as received by the credit check service.
    a. If the credit check service couldn’t verify the identity information, it returns an error message. This message is displayed to the user instead of their credit score.

Post-condition: The user’s credit score is displayed.

Export Data

Pre-condition: The user is logged in to the application.
Event Flow:

1. The user clicks the “Export Data” button.
2. The system displays prompt for the user to choose a location for saving the data file.
3. The user chooses a save location and clicks the “Continue” button.
4. The system compiles an excel file; each row in the file contains information related to a transaction contained within the user’s account.
    a. If the system doesn’t find any transactions in the user’s account, an error message displays informing the user that it couldn’t find any transactions.
5. The system sends a request to the operating system to create the excel file at the user’s chosen save location.

Post-condition: The operating system receives the create file request.

Print Data

Pre-condition: The user is logged in to the application.
Event Flow:

1. The user clicks the “Print Data” button.
2. The system sends a request to the printer service to print the current screen.

Post-condition: The printer service receives the print request.

Refresh Data

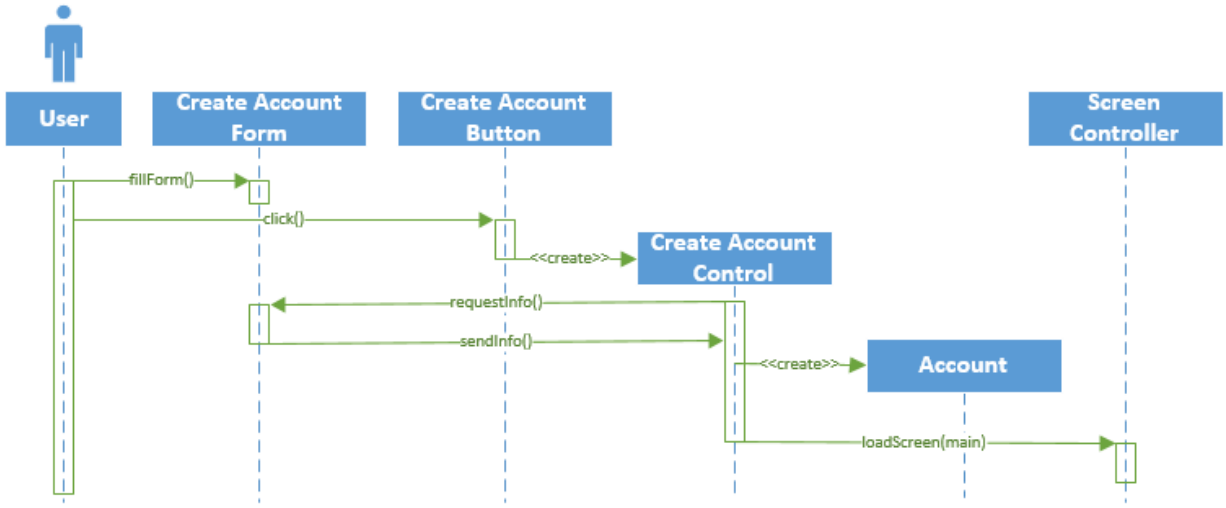
Pre-condition: The user is logged in to the application.
Event Flow:

1. The user clicks the “Refresh Data” button.
2. The system grabs each of the linked financial accounts for the current user. For each financial account the system sends a data sync request to the appropriate financial account service.
3. The financial account service validates the account information provided and respond with transaction data related to that account.
4. After all accounts have synced, the system reloads the current screen.
    a. If the financial account service returned error messages for any of the requests, then these error messages are displayed to the user before the screen gets reloaded.

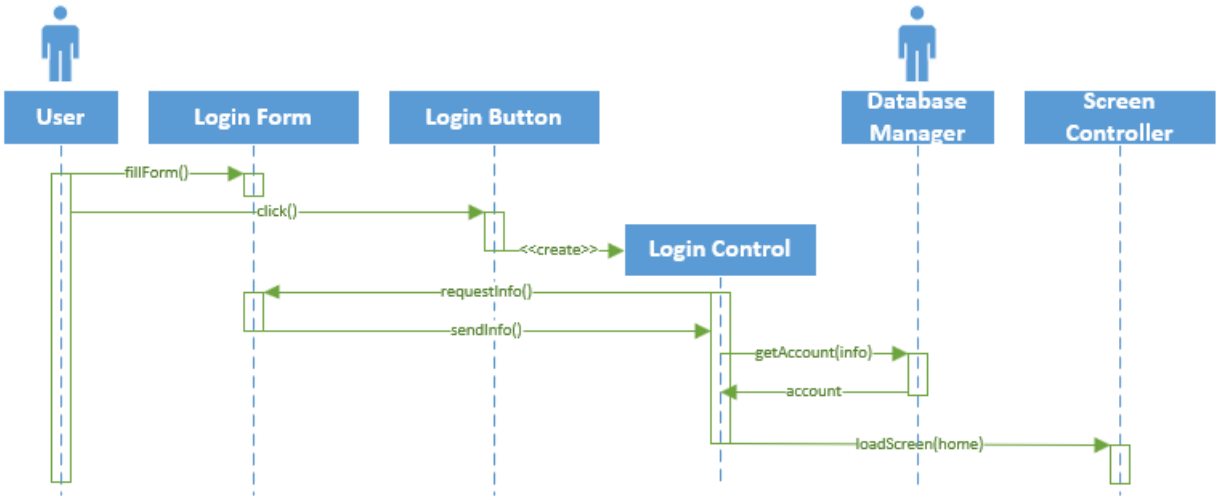
Post-condition: The user’s account now contains all transaction data from any linked financial accounts.

Detailed SSD

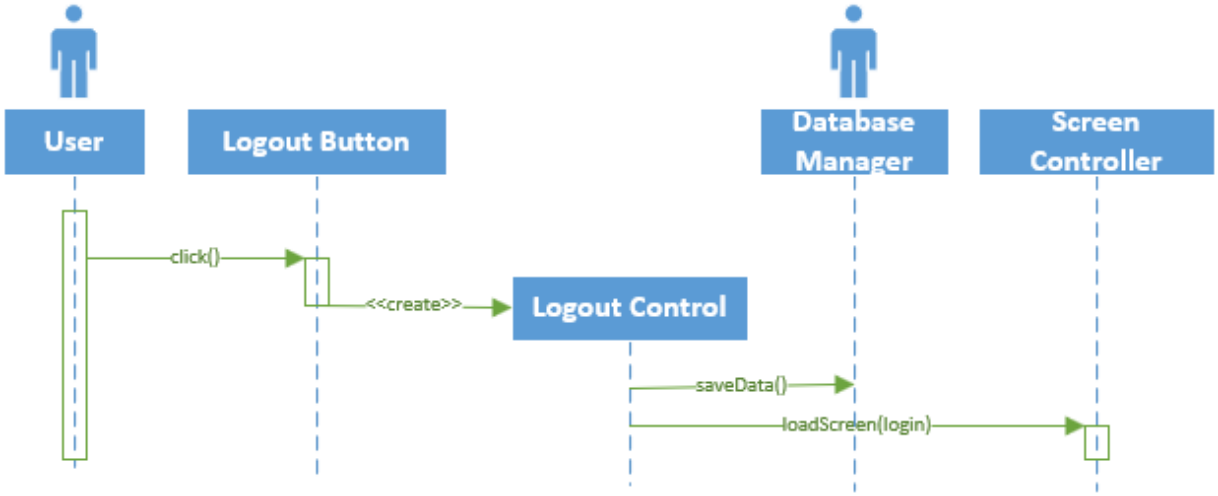
Create Account



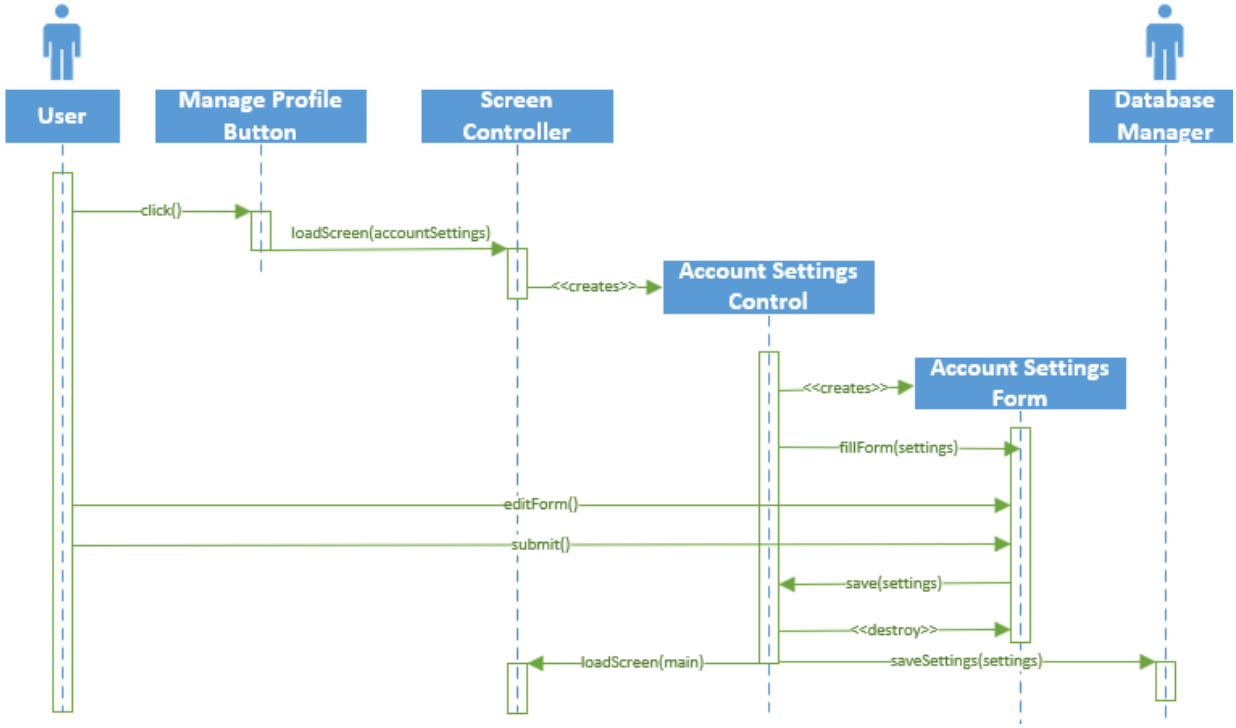
Login



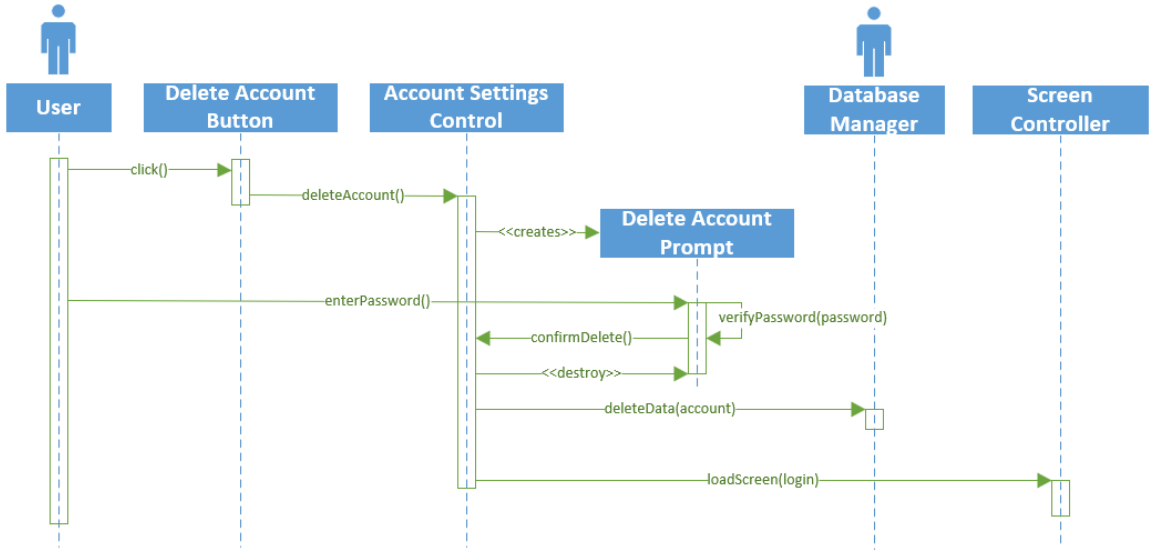
Logout



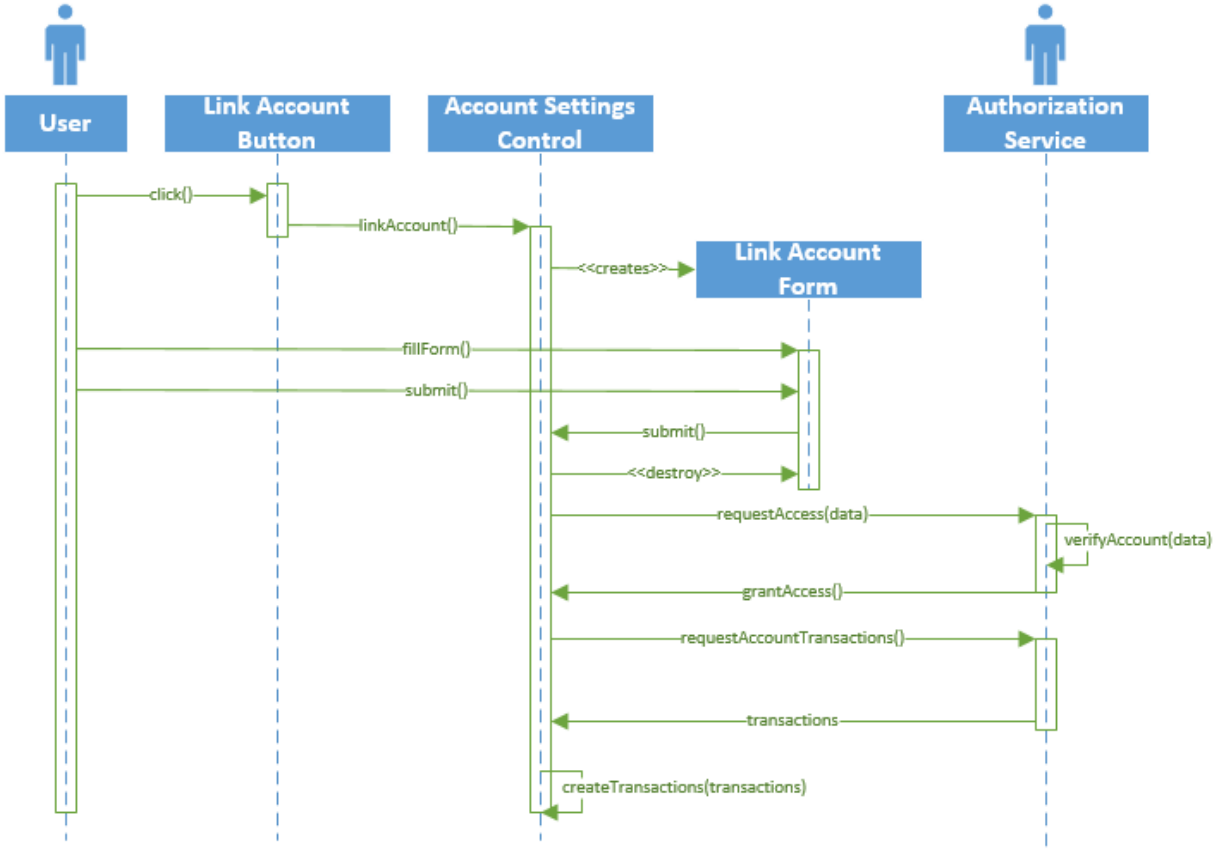
Manage Account



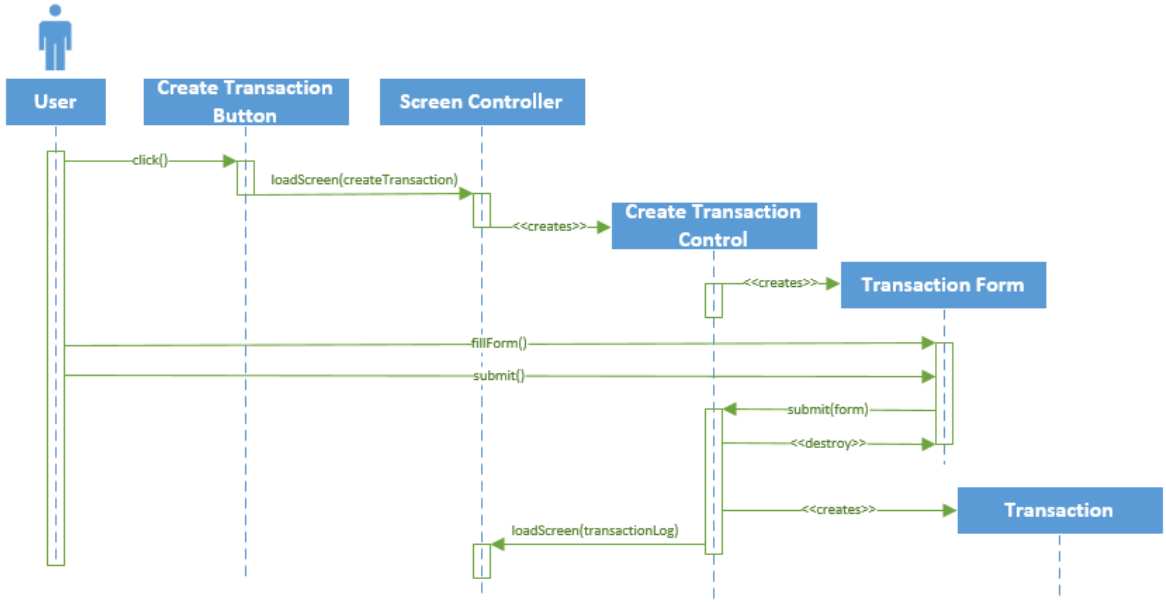
Delete Account



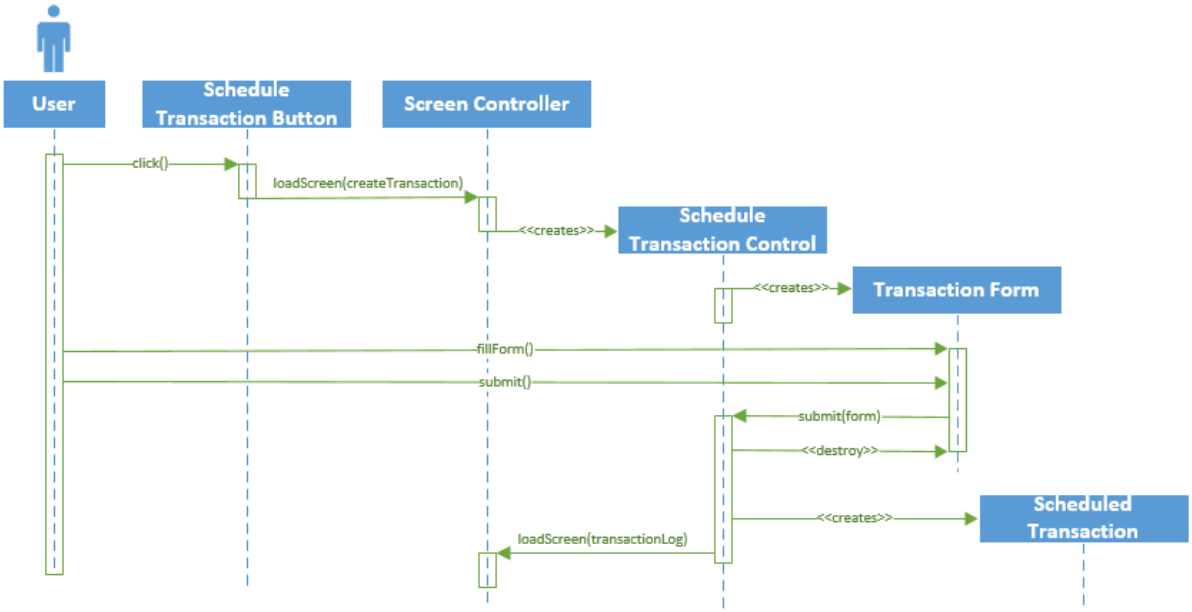
Link Financial Account



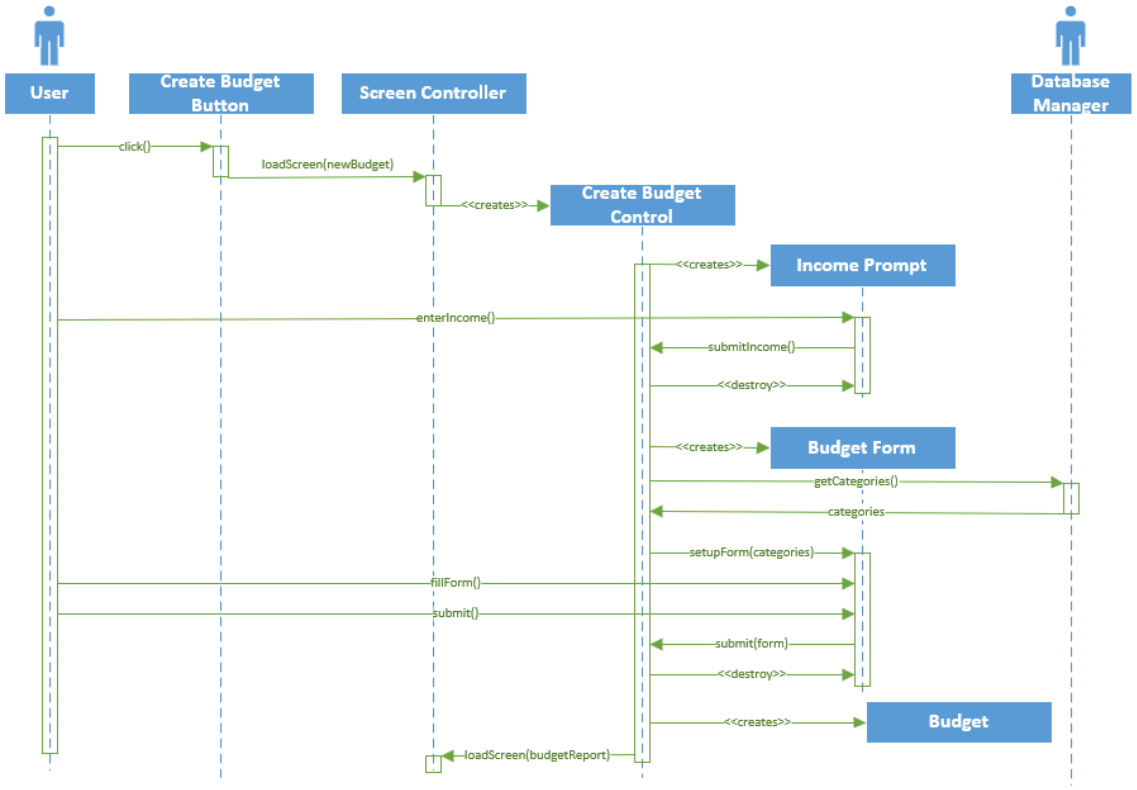
Create Transaction



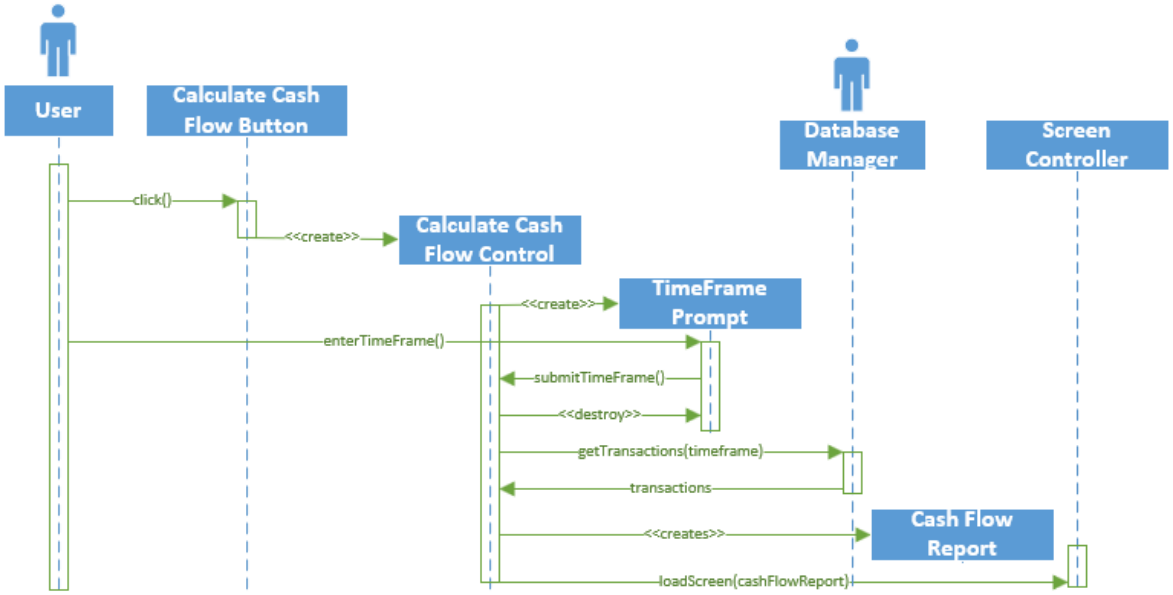
Schedule Transaction



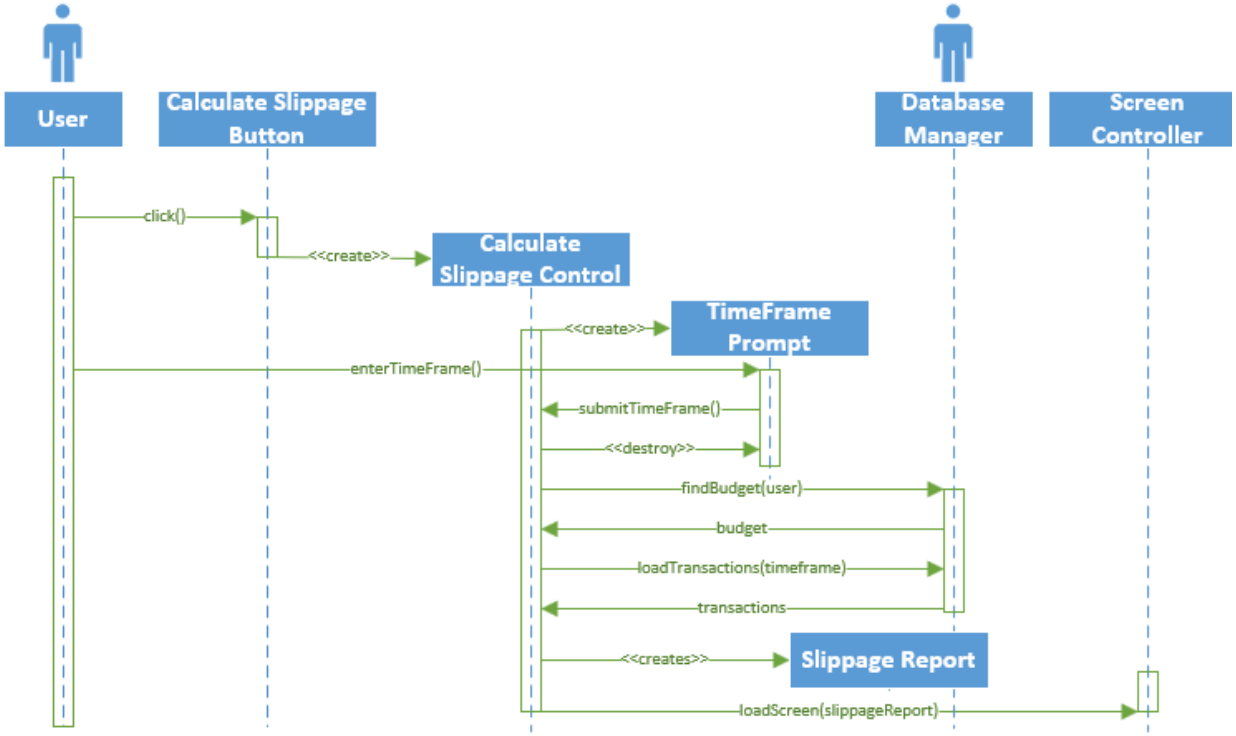
Create Budget



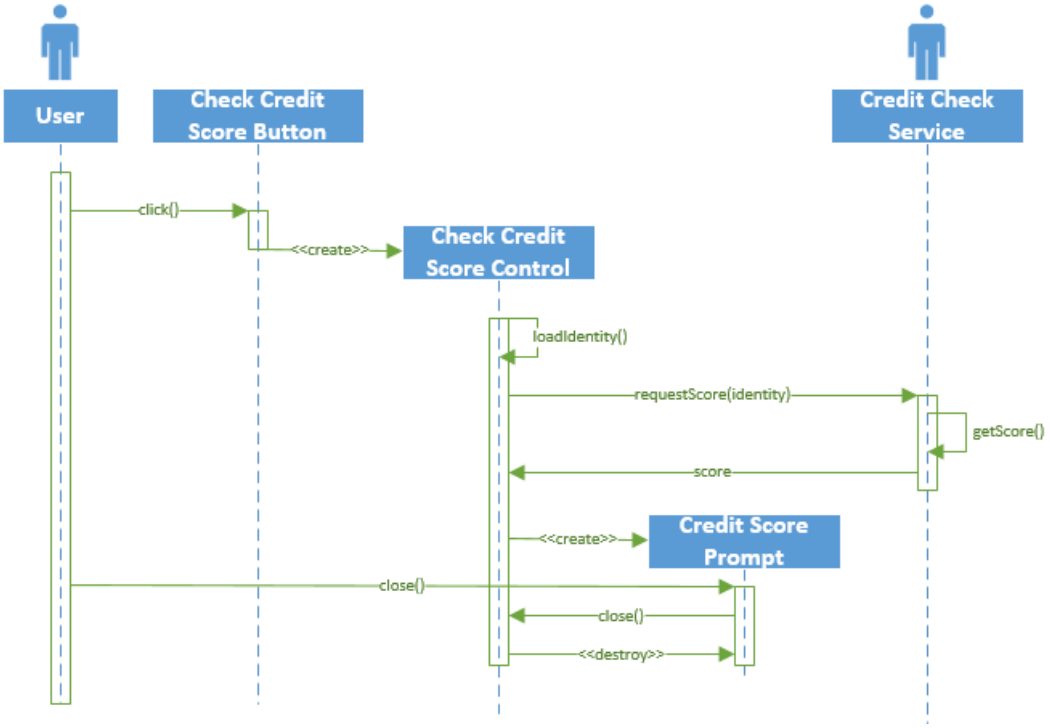
Calculate Cash Flow



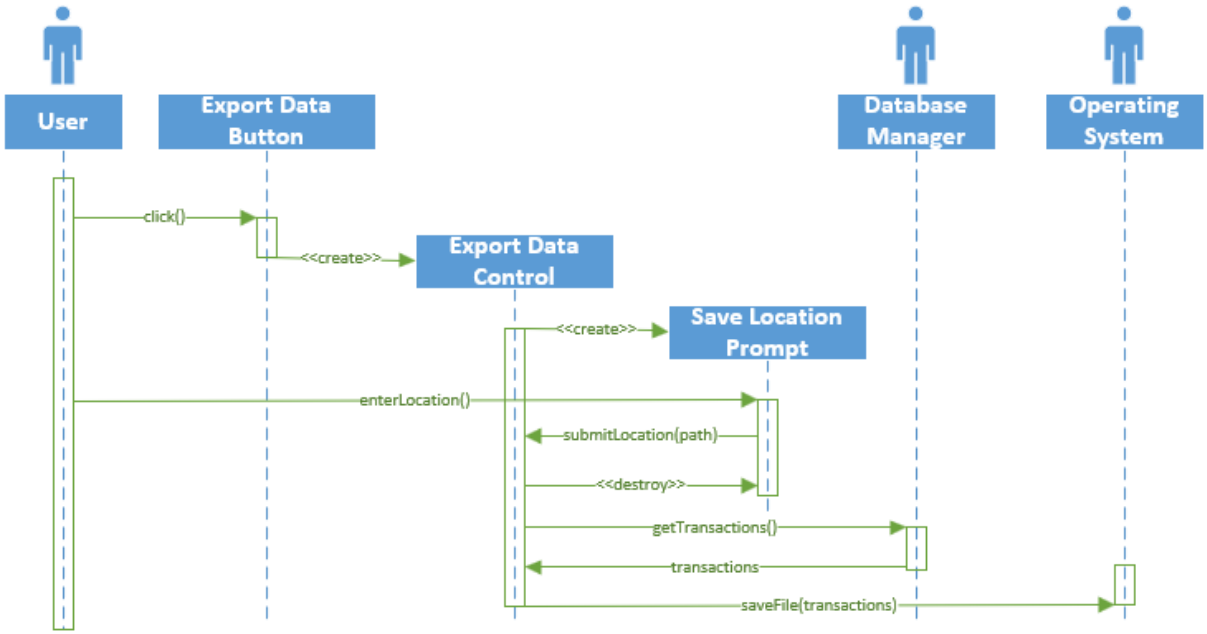
Calculate Slippage



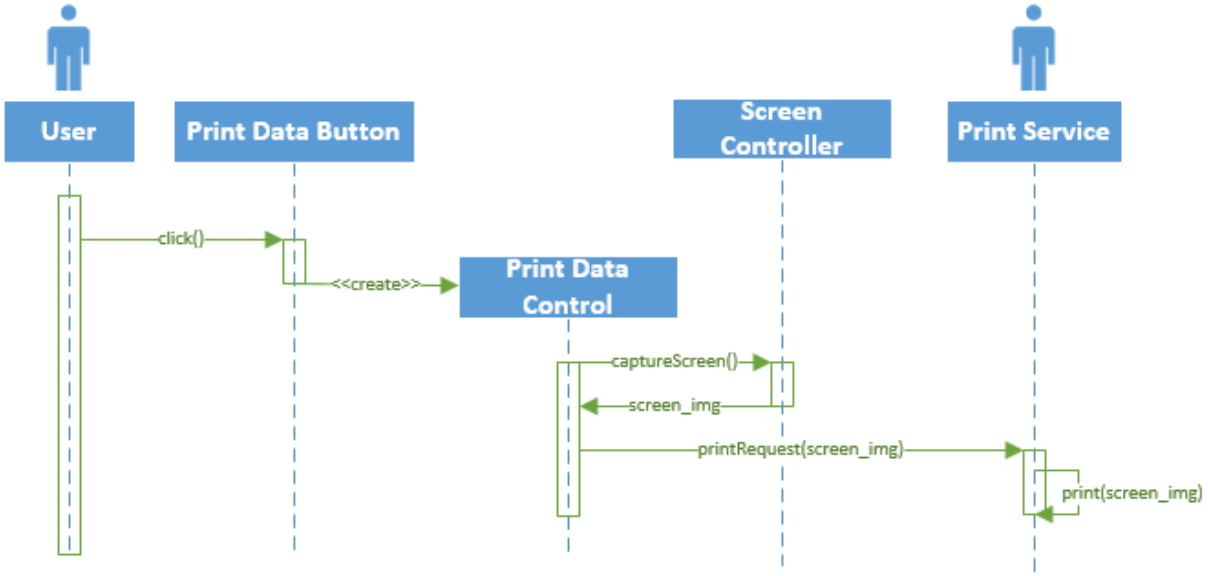
Check Credit Score



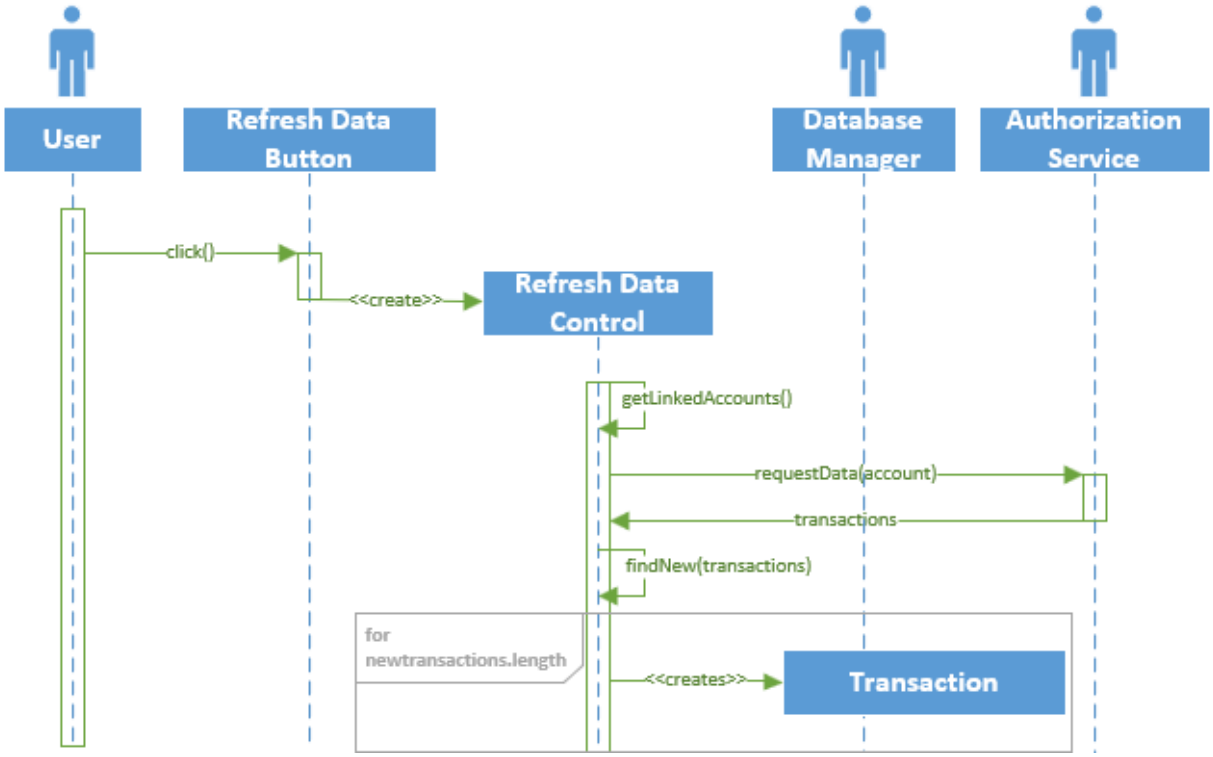
Export Data



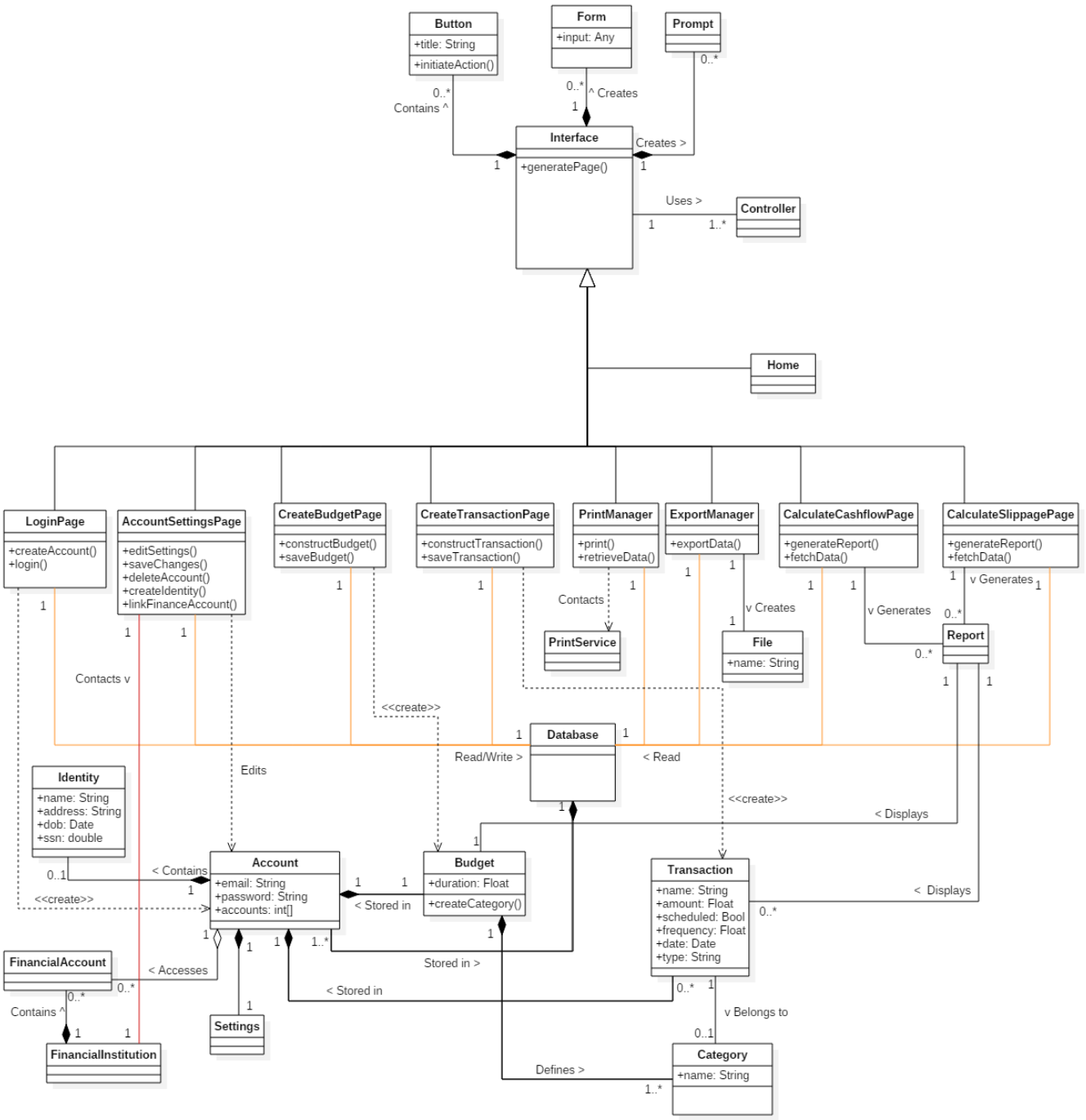
Print Data



Refresh Data



Application Class Model



OCL Constraints & Invariants

- Context Transaction inv:

self.amount != 0

- Context FinancialAccount inv:

Authenticator.authenticate(FinancialAccount) == Enum “account verified”

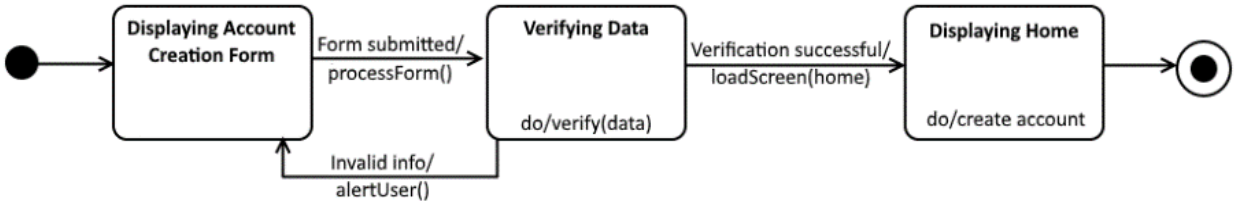
- Context Budget inv:

self.limit > 0
self.type != NULL

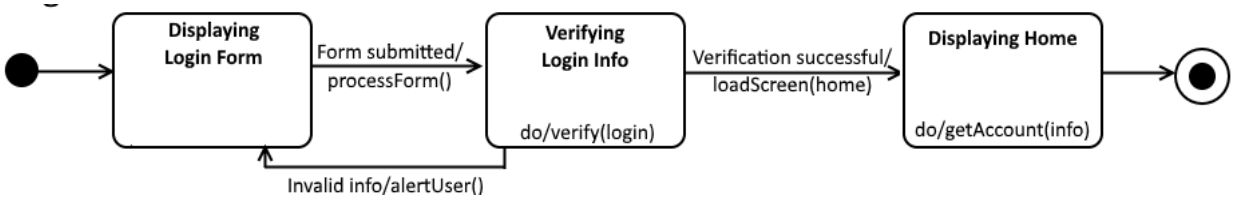
- context Login::login()

pre: UserAccount -> forAll(self.loggedIn = false)
post: userAccount.loggedIn = true

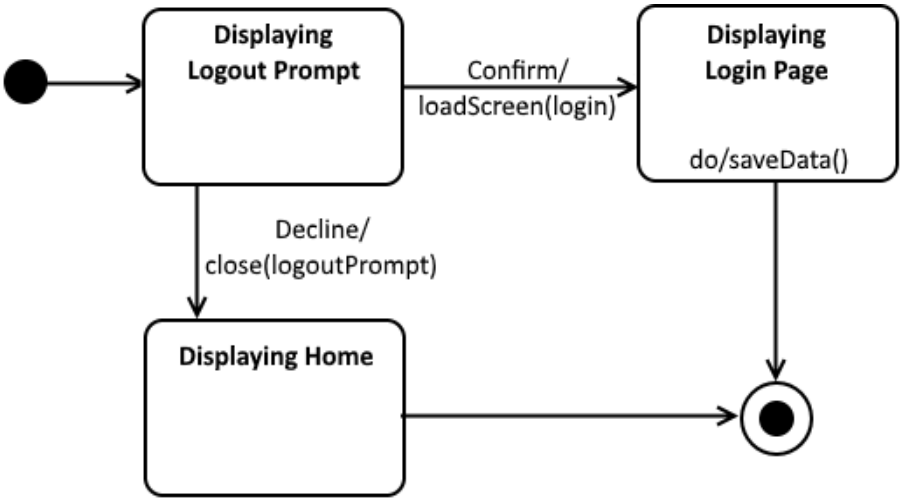
Create Account



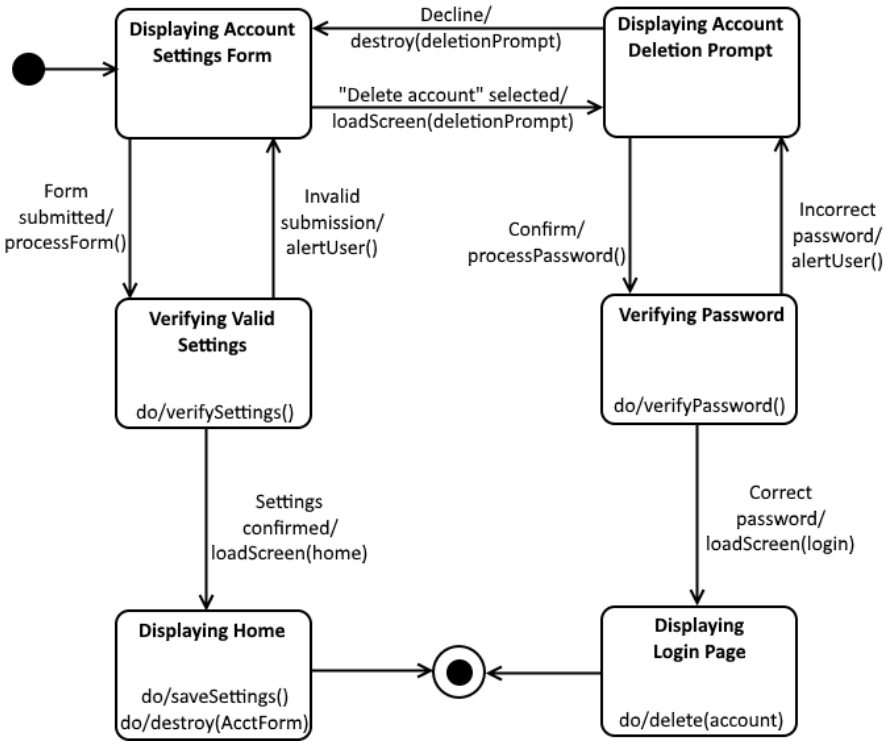
Login



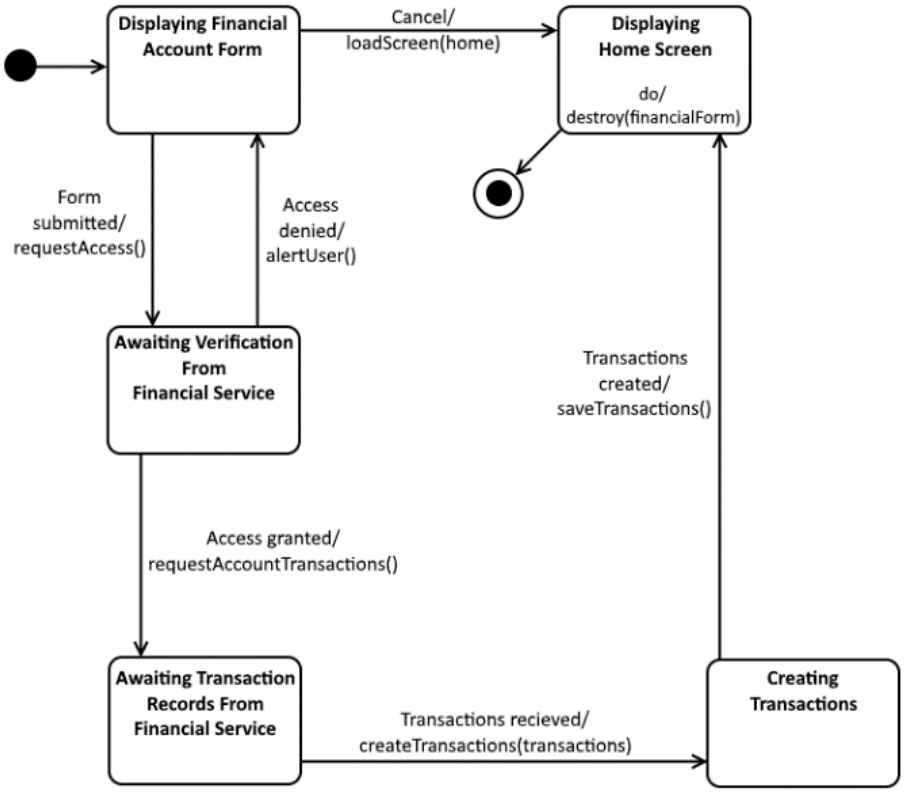
Logout



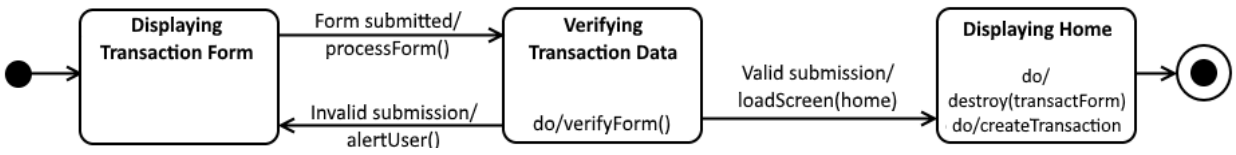
Manage and Delete Account



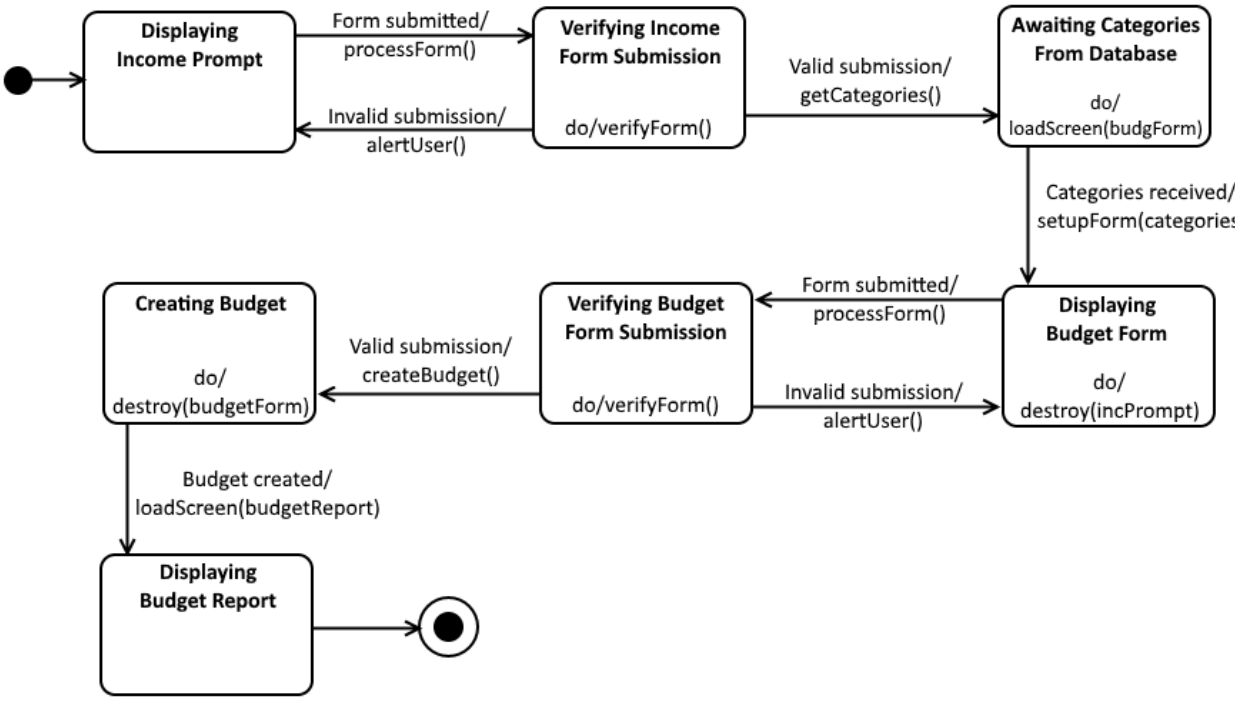
Link Financial Account



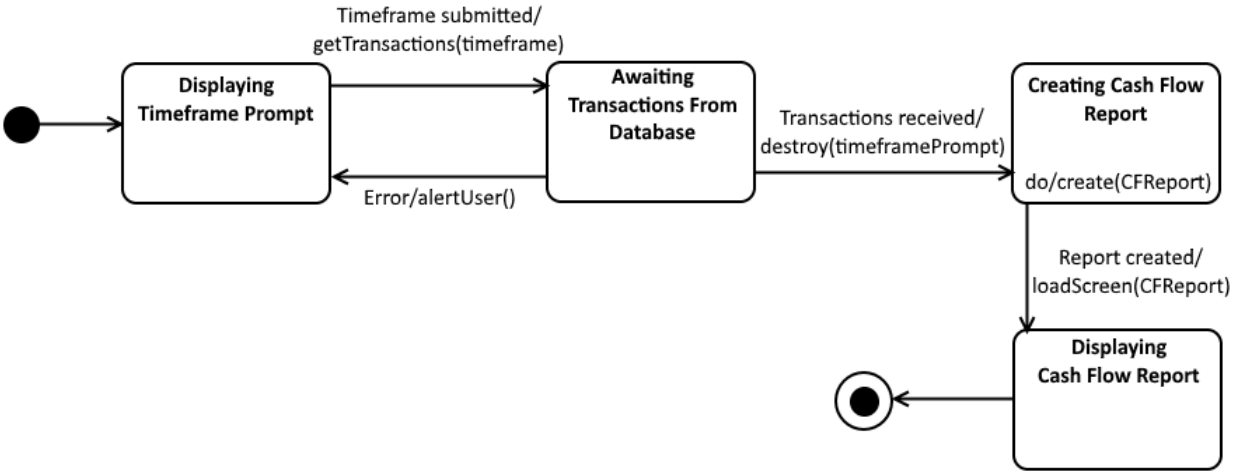
Create/Schedule Transactions



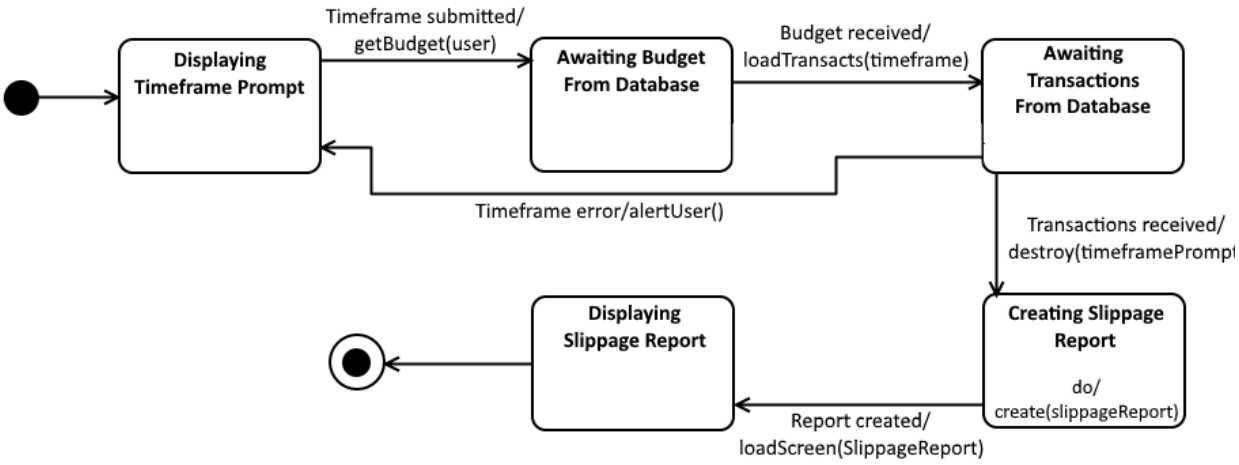
Create Budget Report



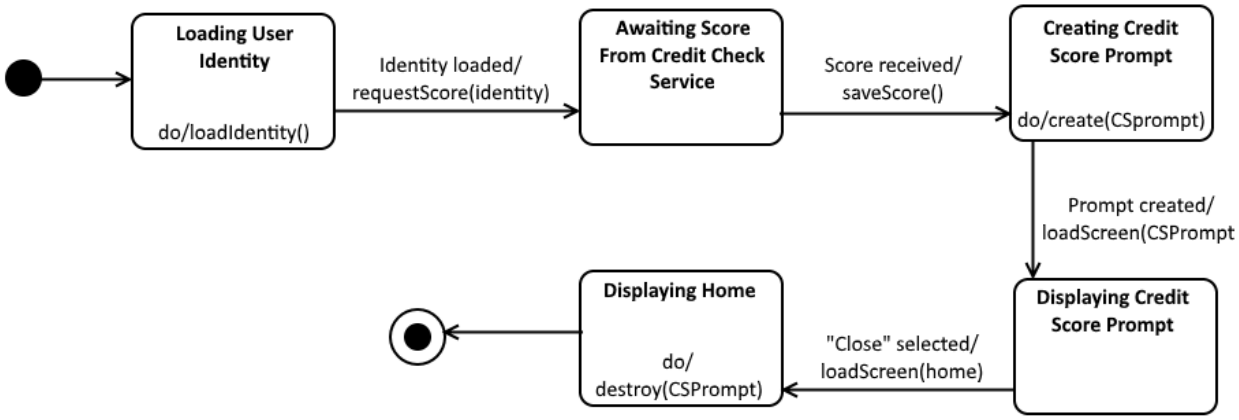
Calculate Cash Flow



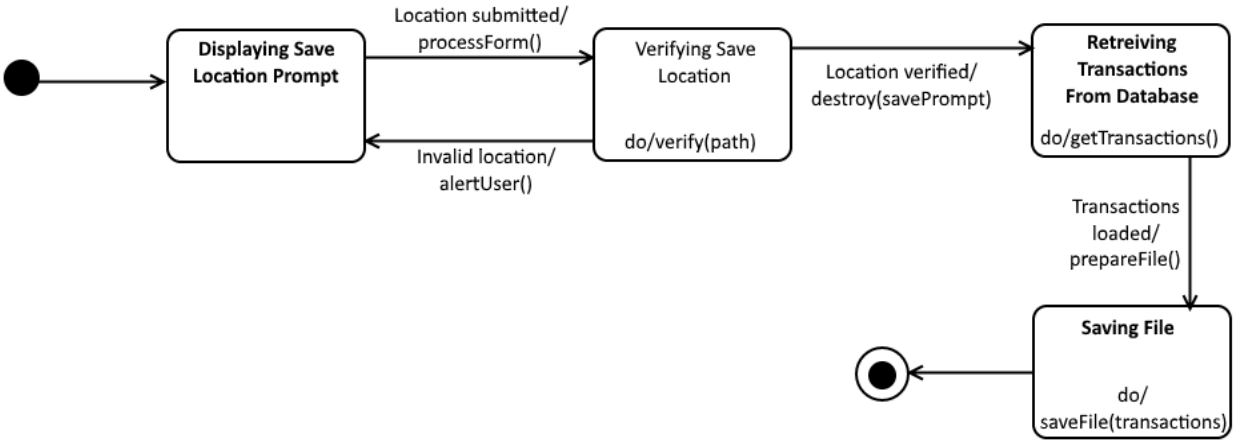
Calculate Slippage



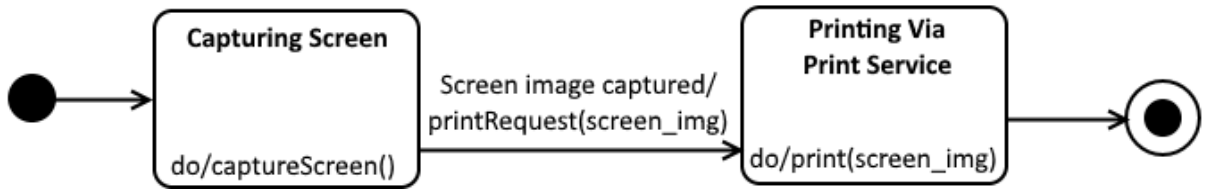
Check Credit Score



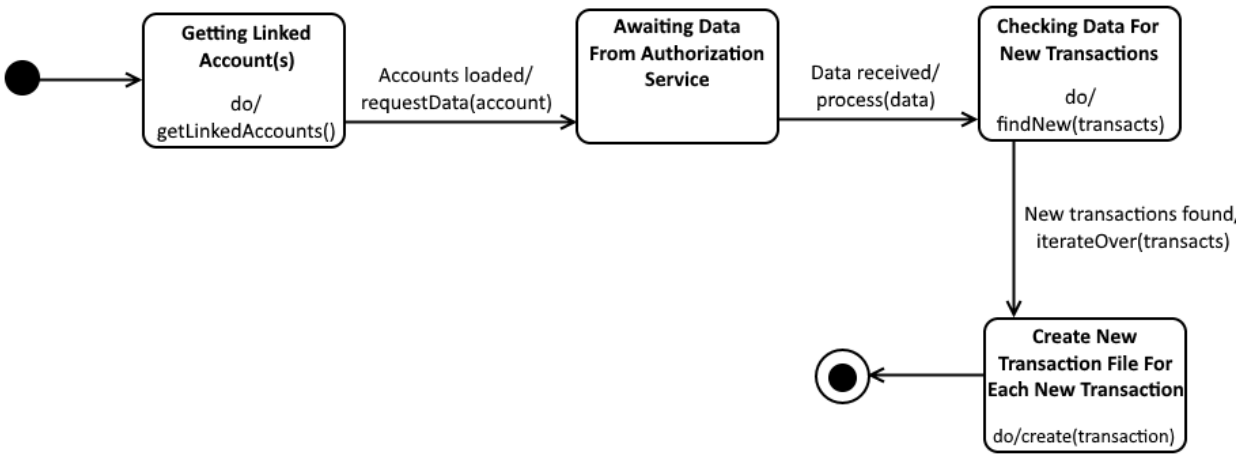
Export Data



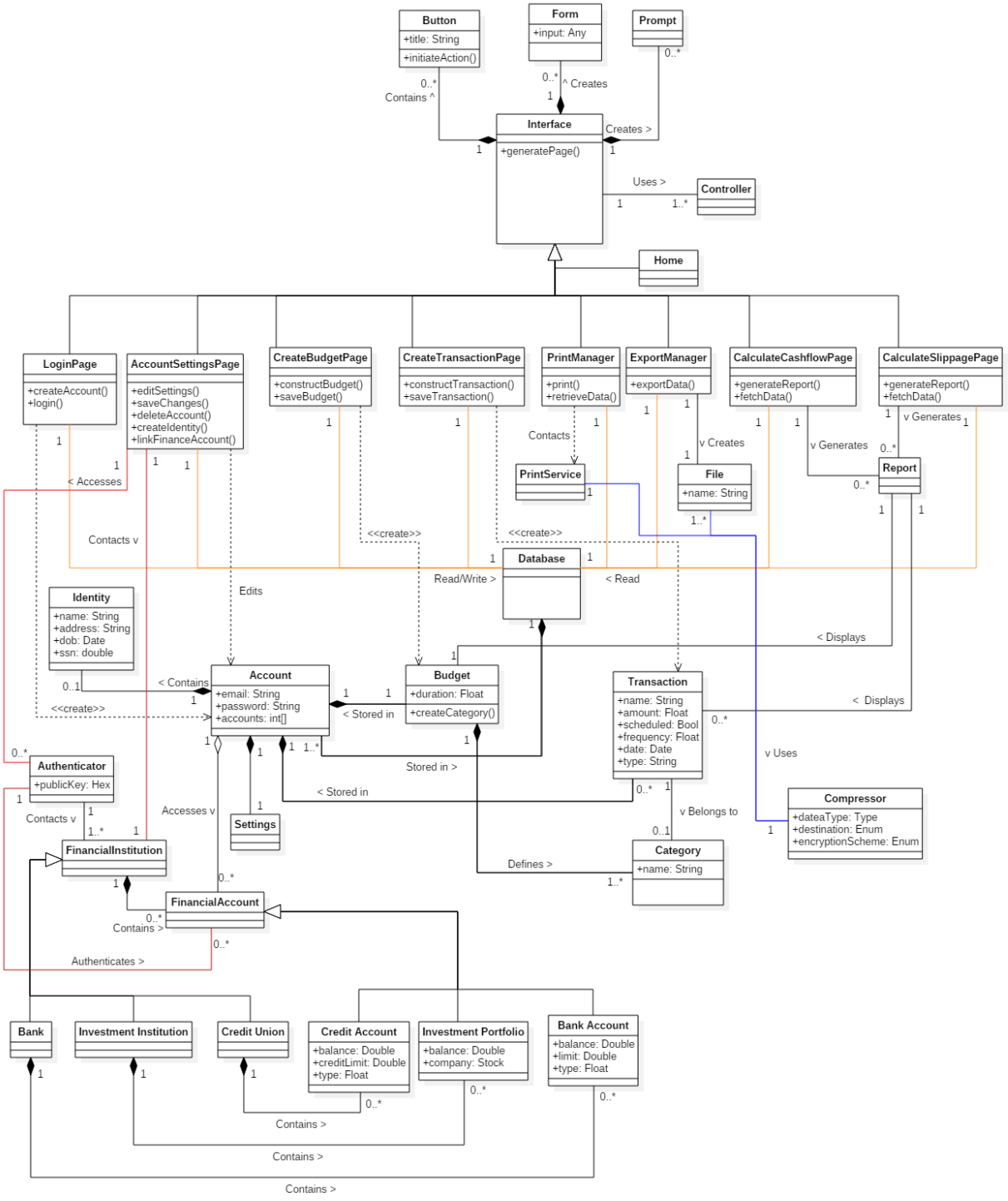
Print Data



Refresh Data



Consolidated Class Model



This document outlined the project analysis portion of a personal finance manager called Cash Stash using an Object-Oriented and UML design modeling process.

The Domain Analysis portion attempts to describe the Cash Stash application on a very general level and does not aim to elaborate on how the application functions internally. The Concept Statement is a generally explains the purpose of the application, who can use it, the features it provides and how it interacts with external systems. Using information from the concept statement, a Conceptual Domain Model (CDM) was generated to show the relationships between high-level systems that make up the applications logic. While most of these associations refer to constructs local to the program, some of them may interact with external systems such as a financial institution. Another type of association shows child-parent relationships, meaning one construct (the child) may inherit the ideas and attributes of another (the parent). The Domain State Model (DSM) elaborates the ideas presented in the CDM by showing their various states and behaviors. The associations between states in the DSM shows what initiates a change of state and these ideas where fabricated with the help of information found within the concept statement.

The Application Analysis portion aims to describe the Cash Stash application on more of an application specific level, meaning it shows in detail how each feature will function and how the internal components of the application interact with each other. The Application Interaction Modal (AIM) describes fifteen different use cases of the application. First, an essential use case will describe a step-by-step interaction between the user, the Cash Stash system and any external systems in general terms. These step-by-step interactions also show how to handle exceptions, which are unexpected actions that may occur during the process. Next, a scenario is written that shows the interaction previously described by the essential use case but using a third person point of view; such as, “John Doe enters his username”. After the scenario, a High-level System Sequence Diagram (HSSD) is created that shows the essential use case in graphical form. The graphical form is easier to understand because it shows a clear interaction between the user and the system with arrows pointing in the direction of the interaction and a message that shows the action being performed. Some internal actions are also represented as additional text on the system side of the chart. These three concepts (essential use case, scenario, HSSD) explain the application use cases in a quick and concise form. Now the concrete use cases will expand the concepts in essential use cases to provide more detail about what is happening during these interactions. For example, where an essential use case may say “System validates information”, the concrete use case will say “The system checks to make sure the email address provided does not belong to an existing account”. Furthermore, the concrete use cases are also converted into a graphical form called the Detailed System Sequence Diagram (DSSD). The DSSD has a similar style to the HSSD but it shows more information than just the interactions between the user and the system. The DSSD also contains boundary objects that represent the interface with which the user interacts to initiate a process, control objects which manage the rest of the interaction within a use case and entity objects which are representations of the data being created by a process. For each DSSD, a control object is created due to the interaction between a user and a boundary object, then the control object may or may not request further interaction from the user throughout the process. Similar to the CDM and DSM from the Domain Analysis portion, an Application Class Model (ACM) shows the relationships between the different objects found in the DSSD and an Application State Model (ASM) shows the various states and behaviors of each application use case. While the ACM shows associations of

objects among all use cases, the ASM will show the state changes of each use case based on the incoming and outgoing interactions with their respective control objects found in the DSSD.

The Consolidated Class Model aims to combine the CDM with the ACM to show how the individual processes of the application interact with the more general systems. By taking the CDM and comparing it with the ideas in the ACM, some of the concepts were similar enough that they could be consolidated into one construct which helped simplify the system.