CPS731 Software Engineering

Team 8

Phase 2: Final Report

December 4th, 2021

Ayusha Adhikari, Syed Ali, Cindy Fang, Jenny Long

NOTES:

Changes from the previous phase:

1) Architecture Design
2) Interaction Model
3) Design Class Model

Table of Contents

Project Statement……..... **4**

Informal Requirements Description…….. **4**

Use Case Model…….... **5**

Use Case Diagram……. 5

Use Case ……...……... 6

Non-functional Requirements…… **15**

Domain Model…….…… **16**

User Interface Sketch/wireframe/prototype…….. **17**

Project Plan Model…….. **19**

Task Breakdown……...……. 19

Gantt Chart……. 20

Phase 1: Home Page……..… 21

Phase 2: Main Features……. 22

Phase 3: Resource & Guide……...… 23

Phase 4: Miscellaneous…….. 24

Architectural Design…… **25**

Detailed Design…… **27**

Interaction Model…….. 27

Design Class Model……... 29

State Machine Model……. 31

Test Design…….... **32**

Review of Project Plan…….. **44**

1 Project Statement

Personal money management and financial literacy are very important for everyone. Currently, there are many youth and individuals that have trouble managing their finances or do not know much about the topic. There should be an easily usable and reliable web application that not only aids users in budgeting their money but also teaches them. Users should be able to set up a unique profile to input and set up a budget plan, receive reminders to make payments, extra tools to aid payment and learn more about financial literacy.

2 Informal Requirements Description

Once a user has finished signing up, they have the option to browse through the following menus: User Profile Section, Payment Reminder Section, Budgeting Section, Investment Recommendation Section, Tutorials Section, Bill and Tip Splitter Section. Under the User Profile Section, the user can view and update their personal information. The Payment Reminder Section allows users to set their payment due dates so that they are reminded to pay for their taxes, loans, or any other bills. Users can budget their expenses through the Budgeting Section. A user can use the Expense Tracker section where they will input their expenses and how much they want to budget. This section also offers a built-in calculator where users can calculate their annual income, investments, retirement, loans, interest, and exchange rate conversions. There is also a section that will provide users with various recommendations on investment types based on questions that the user answers. From the Tutorials Section, users can get more information on how to manage their finances. This section will offer a variety of articles, videos, and infographics on investments. Through the Bill and Tip Splitter section, users can calculate their payments after splitting their bills with others. This function will help users in cases where they go to a restaurant and want to split the bill or other similar scenarios. This function also allows users to calculate tips in any percentage specified.

While the application is under construction, some of the expected constraints are budget, time, dependency on other applications, and complexity. If our application has many requirements and/or requires frequent updates, this could be very costly. We may also experience difficulties finishing the application on the scheduled due date. If the application is dependent on other applications that are not yet completed, this could be an issue too. Since the application includes various complex functionalities, complexity is another constraint.

4 Use Case Model

The use case model of the financial literacy web app has been established. The use cases and their relationships are summarized in Figure 1. The individual descriptions of each use case are listed below.

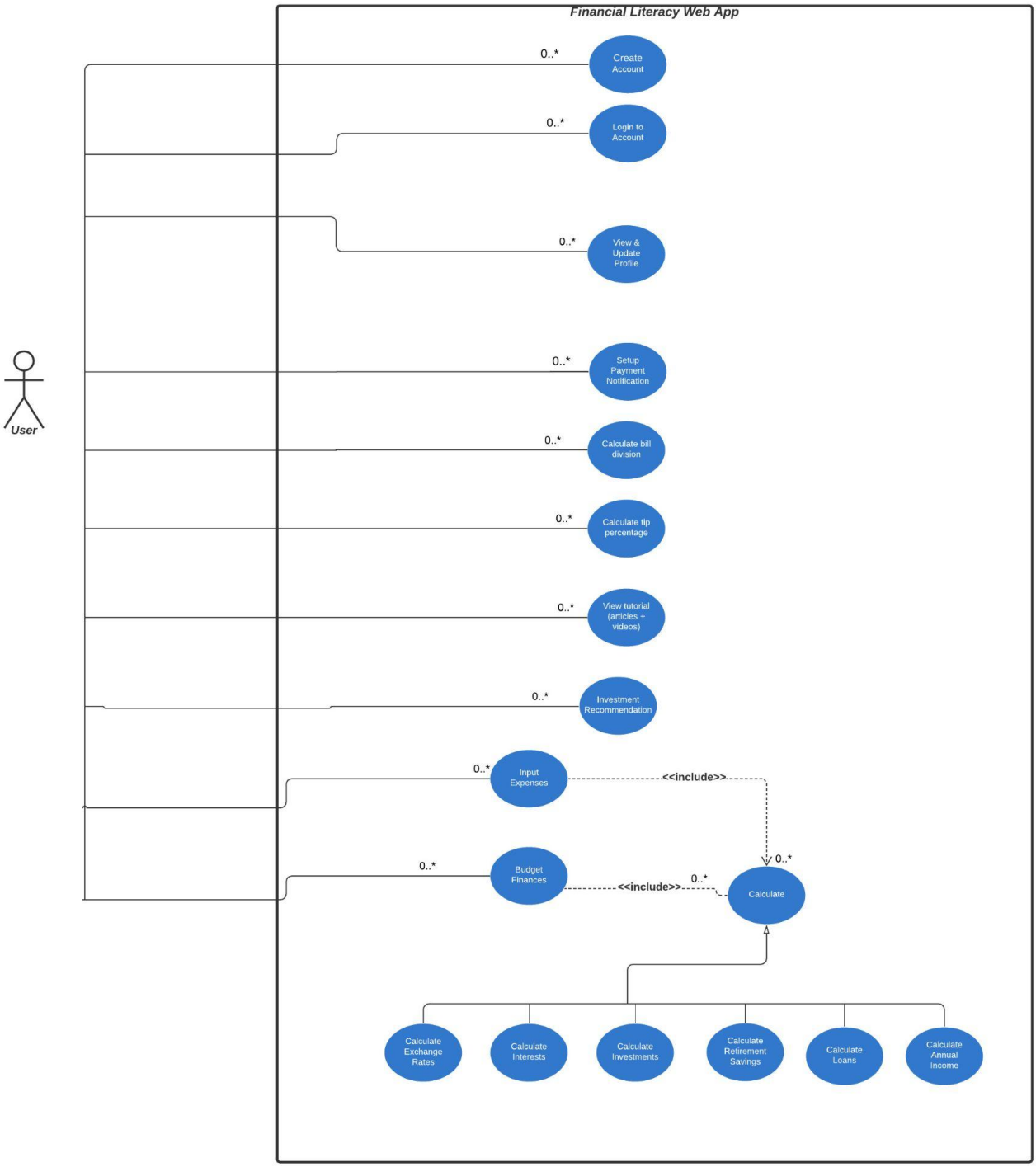


Figure 1: Use Case Diagram

Account Registration Use Case
Use Case: Create an Account
Scope: Account Registration Section
Level: User Goal
Intention in Context: The intention of the User is to register an account so that they can access the financial features.
Multiplicity: Many Users can register for one account.
Primary Actor: User
Main Success Scenario:

1. User enters their email address.
2. User enters the password
3. User agrees to the Terms and Conditions.
4. Users account is created.

Extensions:
(1-4) a. User informs the System that they want to cancel the registration process.
(1-4) a. 1. System redirects to home page; use case ends in failure.
3a. User does not agree to the Terms and Conditions; use case ends in failure.
4a. System ascertains that passwords do not match:
4a.1. System informs User and prompts them to retry; use case continues at step 2.
4b. System ascertains that email is invalid:
4b.1. System informs User and prompts them to retry; use case continues at step 1.

Login Use Case
Use Case: Users log in to their account
Scope: Login Section
Level: User Goal
Intention in Context: The intention of the User is to sign in to their account so that they can access the financial features.
Multiplicity: Many Users can log in to their account, but there's only one User per account.
Primary Actor: User
Main Success Scenario:

1. User inputs email and password.
2. System validates login information.
3. User gets access to the home page.

Extensions:
1a. User forgets email or password.
2a. System ascertains that login credentials are invalid.
2a.1 System outputs error message; use case continues at step 1.
3a. System is unable to connect with the home page.
3a.1 System informs User that it is out of service; use case ends in failure.

View and Update Profile Information Use Case
Use Case: View and Update Profile Information Use Case
Scope: User Profile Section
Level: User Goal
Intention in Context: The intention of the User is to view their personal profile and make updates to their information if wanted.
Multiplicity: Many users can be viewing or making updates to their own profiles at the same time.
Primary Actor: User
Main Success Scenario:
1. User enter their User Profile section.
2. System presents User's personal information and profile content to User.
3. User requests System to let them edit their profile, System enters User into editing mode.
4. User make edits to their profile and request System to save and update their changes.
5. System confirms changes and informs User that their updates were successful.
6. System presents User's updated profile to User.
Extensions:
4a. User requests System to cancel making updates to their information.
4a.1. System confirms update cancellation and exits editing mode; use case ends in failure.
5a. System ascertains that User left blanks in their new updates.
5a.1. System informs User and prompts them to enter their information; use case continues at step 4.
5b. System ascertains that User entered invalid information in their new updates.
5b.1. System informs User and prompts them to enter valid information; use case continues at step 4.

Input Expense Use Case
Use Case: Input Expenses
Scope: Expense Tracker Section
Level: User Goal
Intention in Context: The intention of the User is to input their expenses and store them in the database so that they can be used for the budget section.
Multiplicity: User can have multiple expenses.
Primary Actor: User
Secondary Actor: Database
Main Success Scenario:
1. User inputs expenses
2. Expenses gets stored in the database
Extensions:

1a. User input is invalid:

1a.1. System informs User that input is invalid; use case continues at step 1.

2a. Database is unable to save data:

2a.1. System informs User; use case ends in failure.

Calculate Annual Income Use Case

Use Case: Calculate Annual Income
Scope: Expense Tracker Section
Level: User Goal
Intention in Context: The intention of the User is to input all of their annual income, calculate the total and store it in the database.
Multiplicity: Users can have multiple sources of income.
Primary Actor: User.
Secondary Actor: Database
Main Success Scenario:

1. User input incomes
2. System outputs income total
3. Database store income total

Extensions:

1a. User input non-numeric value for income

1a.1. System informs Client that the value is invalid; use case continues at step 1.

2a. System outputs incorrect total; use case ends in failure.
3a. Database is unable to store income total; use case ends in failure.

Calculate investments Use Case

Use Case: Calculate investments
Scope: Expense Tracker Section
Level: User Goal
Intention in Context: The intention of the User is to calculate the investment growth.
Multiplicity: User can have multiple sources of investments.
Primary Actor: User
Secondary Actor: Database
Main Success Scenario:

1. User inputs initial investment amount.
2. User inputs rate of return.
3. User inputs time to grow.
4. User inputs additional contributions.
5. System calculates and outputs investment growth
6. Database stores investment values to the database.

Extensions:

(1- 4) a. User inputs non-numeric values:

(1- 4) a.1. System informs User and prompts for retry; use case continues in step (1 – 4).

5a. System outputs incorrect value; use case ends in failure.

6a. Databases are unable to store data; use case end in failure.

Calculate Retirement Savings Use Case

Use Case: Calculate Retirement Savings

Scope: Expense Tracker Section

Level: User Goal

Intention in Context: The intention of the User is to calculate the amount of money you need to save for retirement and save that to the database.

Multiplicity: Only one user per retirement savings.

Primary Actor: User

Secondary Actor: Database

Main Success Scenario:

1. User inputs current age and retirement age.
2. User inputs retirement savings and income.
3. System outputs the amount needed to save for retirement.
4. Database saves amount to the database.

Extensions:

1a. User input retirement age to be younger than current age:

1a.1. System informs User and prompts for retry; use case continues in step 1.

2a. User inputs non-numeric values for savings and income:

2a.1. System informs User and prompts for retry; use case continues in step 2.

3a. System outputs incorrect amount; use case ends in failure.

4a. Databases are unable to save data; use case end in failure.

Calculate Loans Use Case

Use Case: Calculate Loans

Scope: Expense Tracker Section

Level: User Goal

Intention in Context: The intention of the User is to calculate the total of loans and store it in the database.

Multiplicity: One user can have multiple loans.

Primary Actor: User

Secondary Actor: Database

Main Success Scenario:

1. User inputs loan amounts
2. System calculates the total of loans.
3. Database stores the total of the loans.

Extensions:
1a. User input non-numeric value for loan amount:
    1a.1. System informs User and prompts for retry; use case continues at step 1.
2a. System calculates the total of loans incorrectly; use case end in failure.
3a. Database is unable to store the loan total; use case end in failure.

Calculate Interests Use Case
Use Case: Calculate Interests
Scope: Expense Tracker Section
Level: User Goal
Intention in Context: The intention of the User is to calculate interest and final balances.
Multiplicity: User can calculate interest one at a time.
Primary Actor: User
Main Success Scenario:
1. User inputs interest rate
2. User inputs starting principal
3. User inputs Contribution
4. System outputs final balance and total interest
Extensions:
(1-3) a. User input non-numeric value:
    (1-3) a.1. 1a.1. System informs User and prompts for retry; use case continues at step 1.
4a. System outputs incorrect value; Use case system ends in failure.

Calculate Exchange Rate Conversions Use Case
Use Case: Calculate Exchange Rates
Scope: Expense Tracker Section
Level: User Goal
Intention in Context: The intention of the User is to calculate foreign exchange rates.
Multiplicity: User can calculate different currencies but one at a time.
Primary Actor: User
Main Success Scenario:
1. User select a country and input their home country currency.
2. System outputs foreign exchange rate.
Extensions:
1a. User enters an incorrect currency format.
    1a.1. System informs User and prompts User to retry; use case continues at step 1.
2a. System is unable to convert exchange rate
    2a.1. System informs User; use case ends in failure.

Budgeting Use Case
Use Case: Budget Finances
Scope: Budgeting Section
Level: User Goal
Intention in Context: The intention of the User is to input and calculate expenses so that they can budget their finances.
Multiplicity: Many Users can input their expenses at the same time.
Primary Actor: User
Secondary Actor: Database
Main Success Scenario:
1. User input their expenses.
2. System asks the user to input a limit on how much the user would like to spend for the month.
3. System asks User to confirm the information inputted.
4. System updates the information in the database.
Extensions:
1a. Users can Calculate expenses (annual income, investments, retirement, loans, interest, and exchange rate conversions) using the data from the Expense Tracker section.
3a. User does not confirm; User will have to input the amount again; use case continues at step 2.
3b. System does not allow the inputted amount if the budget exceeds the total amount of money the user has; use case continues at step 2.

Payment Notifications Setup Use Case
Use Case: Payment Notifications Setup
Scope: Payment Reminder Section
Level: User Goal
Intention in Context: The intention of the User is to set up their payment notification preferences.
Multiplicity: Many users can be setting their notification preferences to their own accounts at the same time.
Primary Actor: User
Main Success Scenario:
1. User enters their Payment Reminder Section.
2. System presents an empty form for users to fill out their time and frequency preferences.
3. User enters their time and frequency preferences, and requests System to save these preferences.
4. System confirms set preferences and informs User their preferences have been set.
5. System presents User's notification timing and frequency preferences to User.
Extensions:
3a. User requests System to cancel setting up payment notification preferences.

3a.1. System confirms cancellation and exits the preferences form; use case ends in failure.
4a. System ascertains that User left blanks in their new preferences.
4a.1. System informs User and prompts them to enter valid input in blanks; use case continues at step 3.
4b. System ascertains that User entered invalid information in their new updates.
4b.1. System informs User and prompts them to enter valid information; use case continues at step 3.

Payment Reminder Notifications Use Case
Use Case: Payment Reminder Notifications
Scope: Payment Reminder Section
Level: System Goal
Intention in Context: The intention of the System is to send payment reminder notifications to User.
Multiplicity: System can send out multiple reminder notifications to different Users.
Primary Actor: System
Secondary Actor: User
Main Success Scenario:
1. System ascertains the current date and time is the same as the set preferences by User.
2. System sends the User a notification of the payment reminder.
3. User responds to the payment reminders and makes the payment.

Extensions:
2a. System is unable to send notifications to User; the use case ends in failure.
3a. User does not respond to the payment reminder.
3.a.1. System snoozes for 1 hour before sending another notification; use case continues at step 2.

Calculate Tip Use Case
Use Case: Calculate tip percentage
Scope: Bill + Tip Splitter Section
Level: User Goal
Intention in Context: The intention of the User is to receive the amount of money that they should tip after a bill.
Multiplicity: Users can calculate tip for multiple bills
Primary Actor: User
Main Success Scenario:
1. User inputs total amount of money for bill
2. System offers recommended tip percentage

3. User inputs how much percentage they would like to tip
4. System calculates total tip amount

Extensions:

1a. User inputs non-numeric value for bill amount
    1a.1. System informs User that value is invalid; use case continues at Step 1.
2a. System outputs incorrect value; Use case ends in failure.
3a. User inputs non-numeric value for tip percentage
    3a.1. System informs User that value is invalid; use case continues at Step 3.
4a. System outputs incorrect value; Use case ends in failure.

Calculate Splitting Bills Use Case

Use Case: Calculate Bill Division
Scope: Bill + Tip Splitter Section
Level: User Goal
Intention in Context: The intention of the User is to receive the amount of money that they should pay after dividing up a bill among a group of people.
Multiplicity: Users can calculate their individual payment for multiple bills
Primary Actor: User
Main Success Scenario:

1. User inputs total amount of money for bill
2. User inputs total amount of people sharing the bill
3. System calculates bill cost for each individual

Extensions:

1a. User inputs non-numeric value for bill amount:
    1a.1. System informs User that value is invalid; use case continues at Step 1.
2a. User inputs non-numeric value for ‘amount of people’:
    2a.1. System informs User that value is invalid; use case continues at Step 2.
4a. System outputs incorrect value; Use case ends in failure.

View Tutorial Articles & Videos Use Case

Use Case: View tutorial articles and videos
Scope: Tutorials Section
Level: User Goal
Intention in Context: The intention of the User is to view helpful articles and videos on how to manage their finances.
Multiplicity: Users can access a variety of articles and videos
Primary Actor: User
Main Success Scenario:

1. User selects the ‘View Tutorials’ option on the website
2. User selects which tutorial they would like to view

3. System displays the tutorial in either article or video form

Extensions:

1a. System is unable to provide ‘View Tutorials’ option; Use case ends in failure.

2a. User selects a non-valid option for their tutorial:

2a.1. System informs User that the option is invalid; use case continues at Step 2.

3a. System is unable to display the correct tutorial; Use case ends in failure.

Investments Recommendation Use Case

Use Case: Investment Recommendation

Scope: Investment Recommendations Section

Level: User Goal

Intention in Context: The intention of the system is to recommend different investment options to the user.

Multiplicity: Many users can take the same survey at the same time.

Primary Actor: User

Main Success Scenario:

1. User answers a series of questions asked by the system.
2. System recommends different investment options based on the answers.
3. Users can click on the “View Tutorial Articles & Videos Use Case” link to learn more about investments they are interested in.

Extensions:

1a. User does not answer a question; the system asks the user to answer again; Use case continues at step 1.

2a. System cannot generate any recommendations; Use case end in failure.

5 Non-functional requirements

Product related requirements:

1. Security: System and its data must be protected against unauthorized users and malicious attacks.
2. Capacity: System must be able to store and load all information inputted by its users.
3. Reliability: System must not experience critical failure at a probability of at least 80% per week.
4. Usability: Users must be able to use the application without prior knowledge of money management and/or budgeting.

Organizational or external requirements:

1. Delivery: the final web application and its documentation must be delivered by December 4, 2021.
2. Ethical: System must create achievable budgeting goals for users and provide accurate information in tutorials.

6 Domain Model

Figure 2 shows the domain model that has been established during requirements elicitation to accompany the use case model.

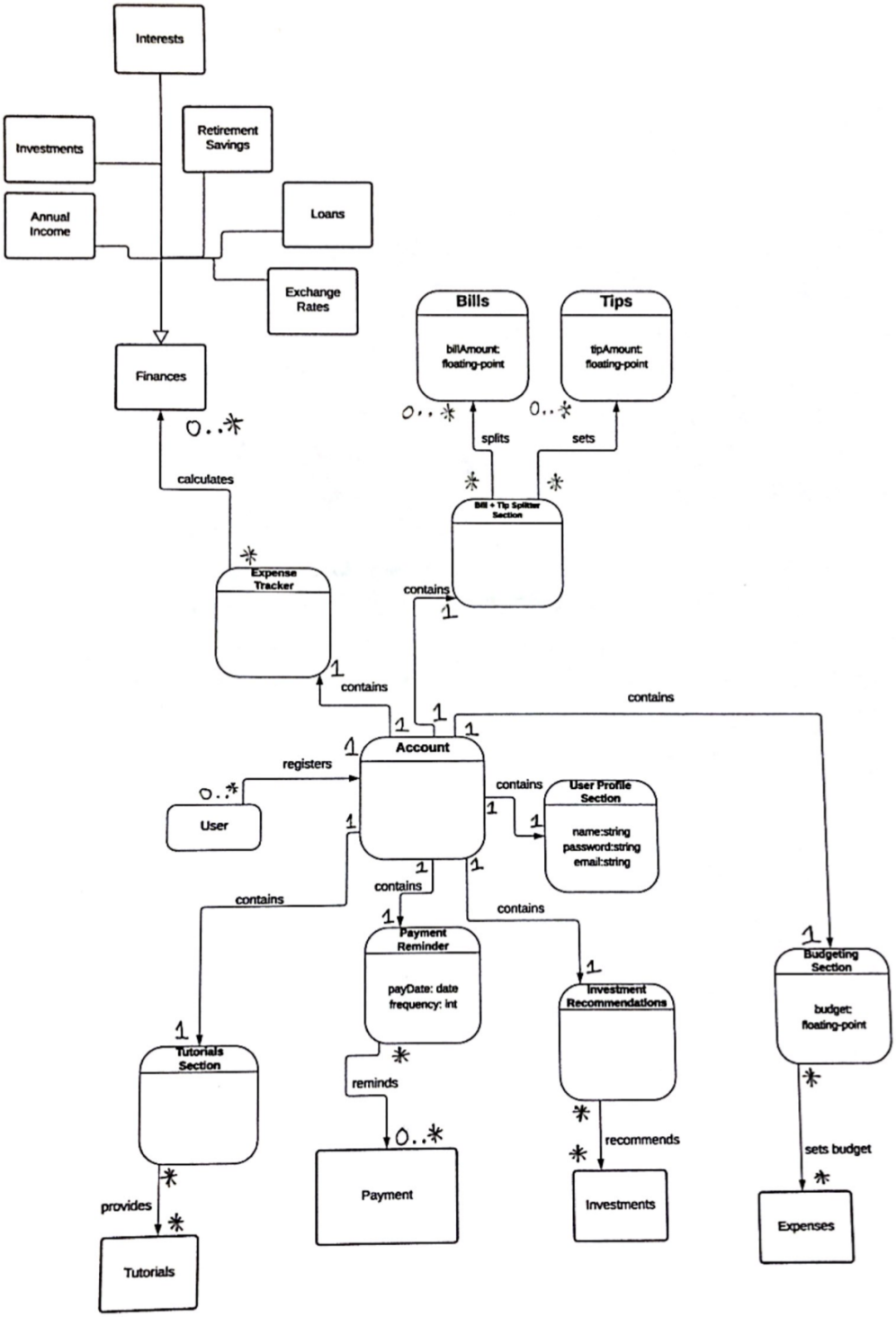


Figure 2: Financial literacy Web app Domain Model

7 User interface sketch / wireframe / prototype

Figures 3 to 9 shows the user interface sketch of all the main components of the financial literacy web application.

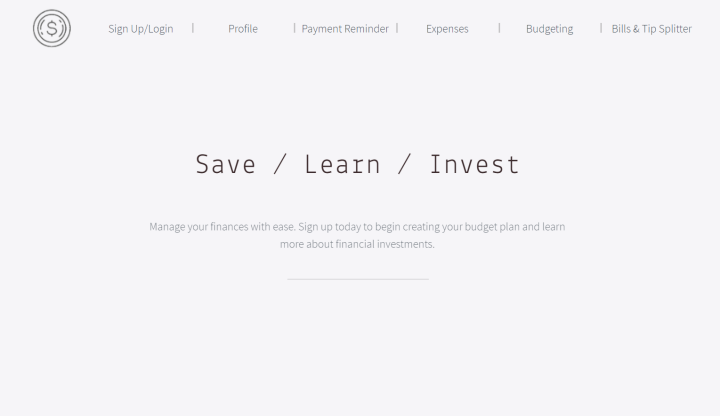


Figure 3: landing page

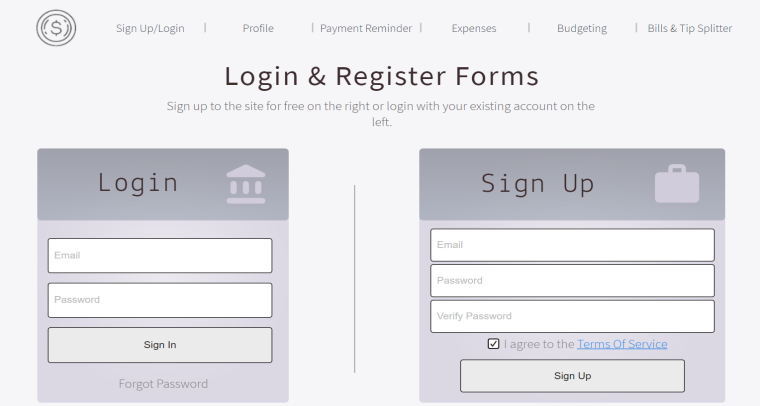


Figure 4: Login and registration page

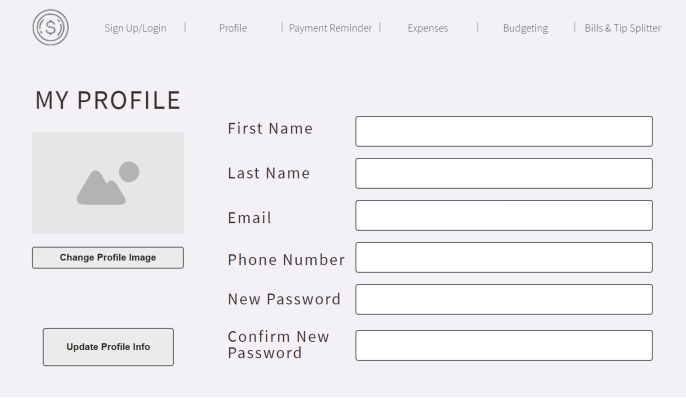


Figure 5: Profile page

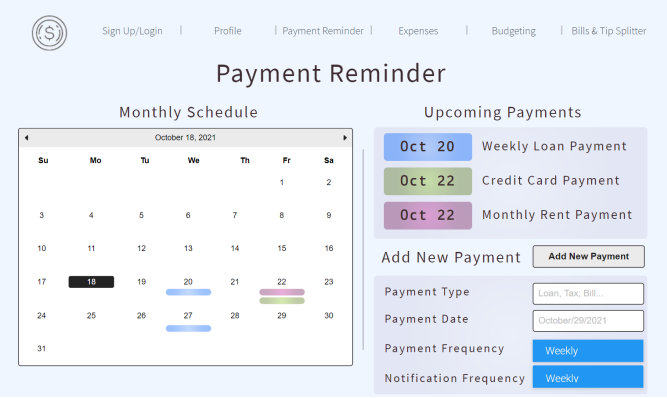


Figure 6: Payment reminder page

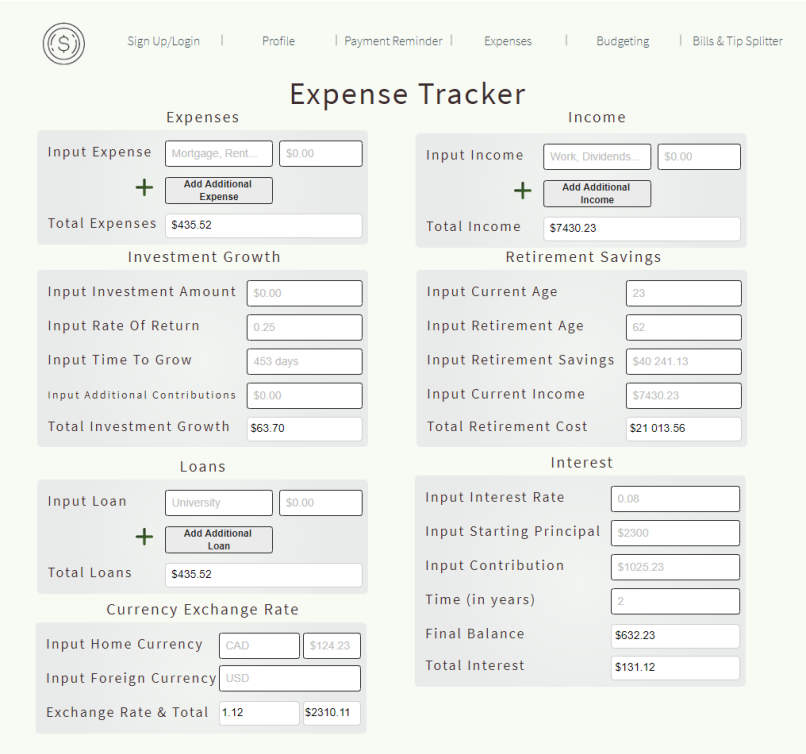


Figure 7: Expenses page

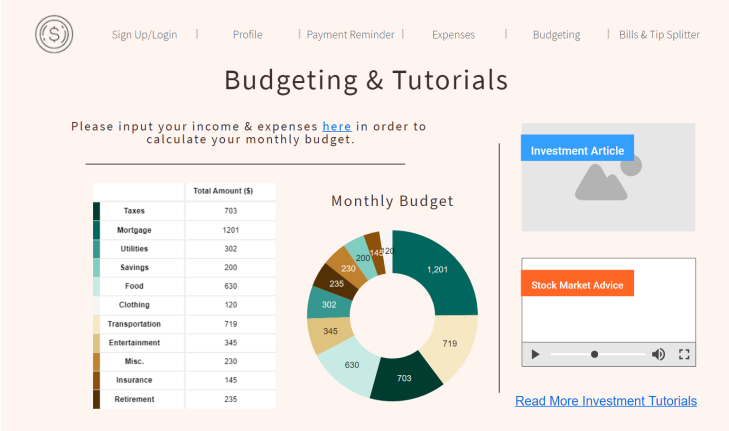


Figure 8: Budgeting and Tutorials page

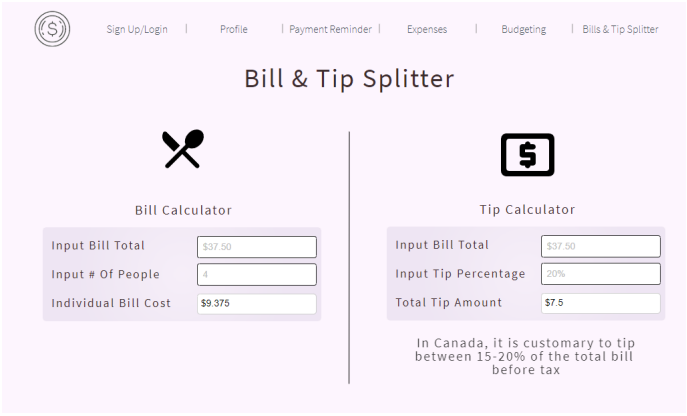


Figure 9: Bill and Tip splitter page

8 Project Plan Model

This section includes the breakdown of the phases and activities, Gantt chart as well as the milestones, time estimate per tasks, slack times and critical paths.

Phase 1: Home Page	**Phase 2: Main Features**	**Phase 3:Resource + Guide**	**Phase: 4: Miscellaneous**
Step 1: Home page	**Step 1:** Prepare budget section	**Step 1:** Investment recommendations	**Step 1:** Payment reminder section
Activity 1.1: Design home page layout	Activity 1.1: Design Budget section	Activity 1.1: Design investment recommendations page	Activity 1.1: Create Notification setting form
Activity 1.2: Code home page	Activity 1.2: Code budget page	Activity 1.2: Code investment recommendations page	Activity 1.2: Implement Reminder alerts
Step 2: Registration	**Step 2:** Expense Tracker Section	Activity 1.3: Set up questions for investments	**Step 2:** Bill + tips splitter
Activity 2.1: Create registration form	Activity 2.1: Create expense form	Activity 1.4: Implement algorithm for matching investment	Activity 2.1: Implement tips calculator
Activity 2.2: Set up Terms + conditions	Activity 2.2: Implement annual income calculator	Activity 1.5: Set up summary of investments	Activity 2.2: Implement splitting bills calculator
Step 3: Login Page	Activity 2.4: Implement retirement calculator	Activity 1.6: add hyperlink to investment summary	Activity 2.3: Implement payment calculator
Activity 3.1: Create login form	Activity 2.5: Implement investment calculator	**Step 2:** Tutorials Section	Activity 2.4: Testing
Activity 3.2: Set up login validation	Activity 2.6: Implement exchange rate calculator	Activity 2.1: Design tutorials layout	
Step 4: User Profile	Activity 2.7: Store data	Activity 2.2: Code tutorials layout	
Activity 4.1: Design profile interface	**Step 3:** Budgeting section	Activity 2.3: Insert content	
Activity 4.2: Set up user profile form	Activity 3.1: Create budget form	Activity 2.4: Testing	
Activity 4.3: Testing	Activity 3.2: Implement budget limit function		
	Activity 3.3: Testing		

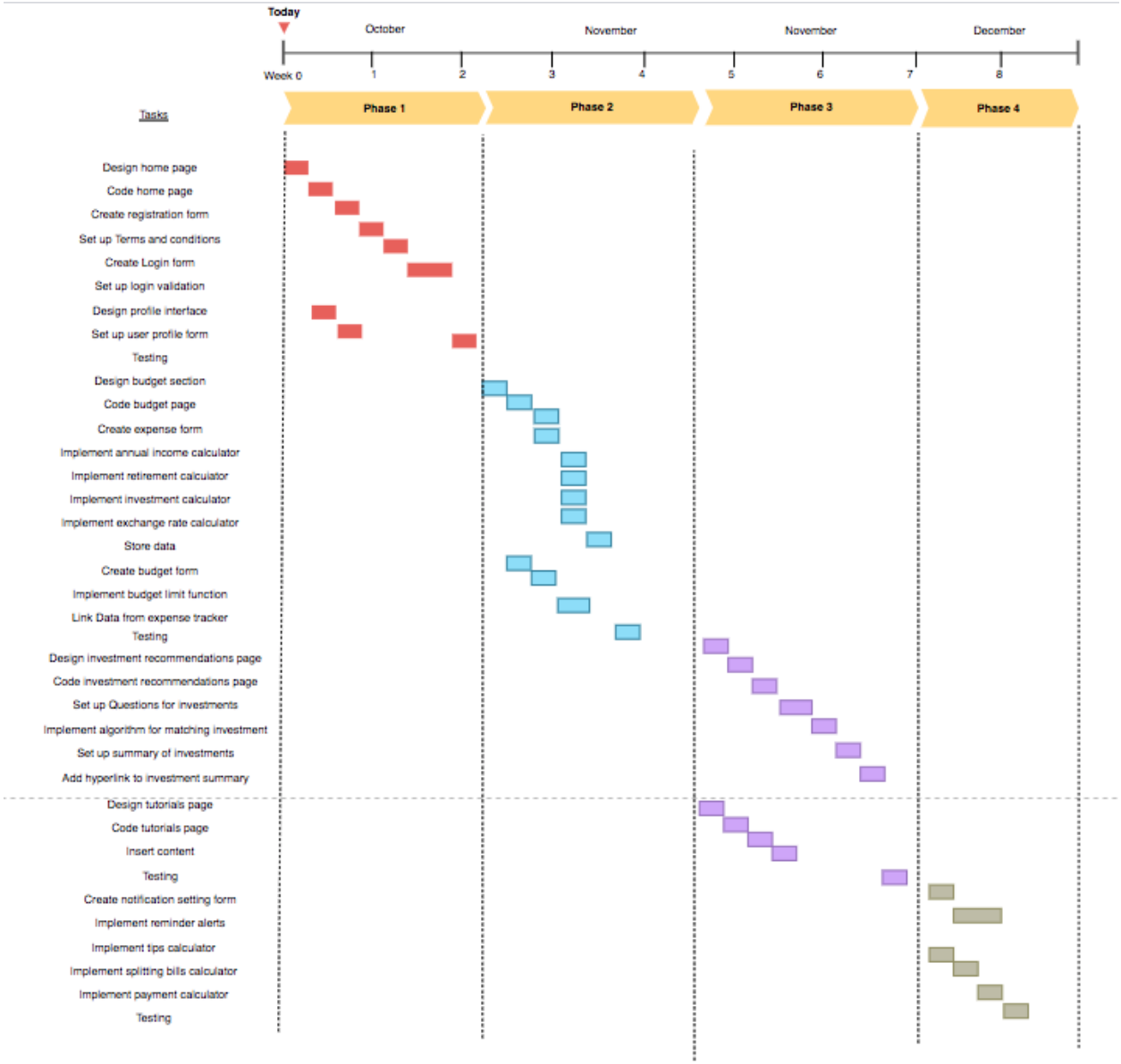
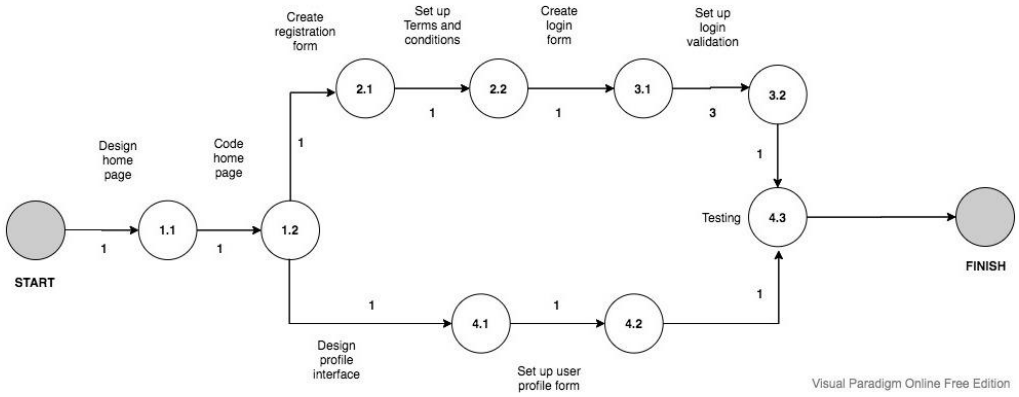


Figure 10: Gantt Chart

Phase 1: Home page



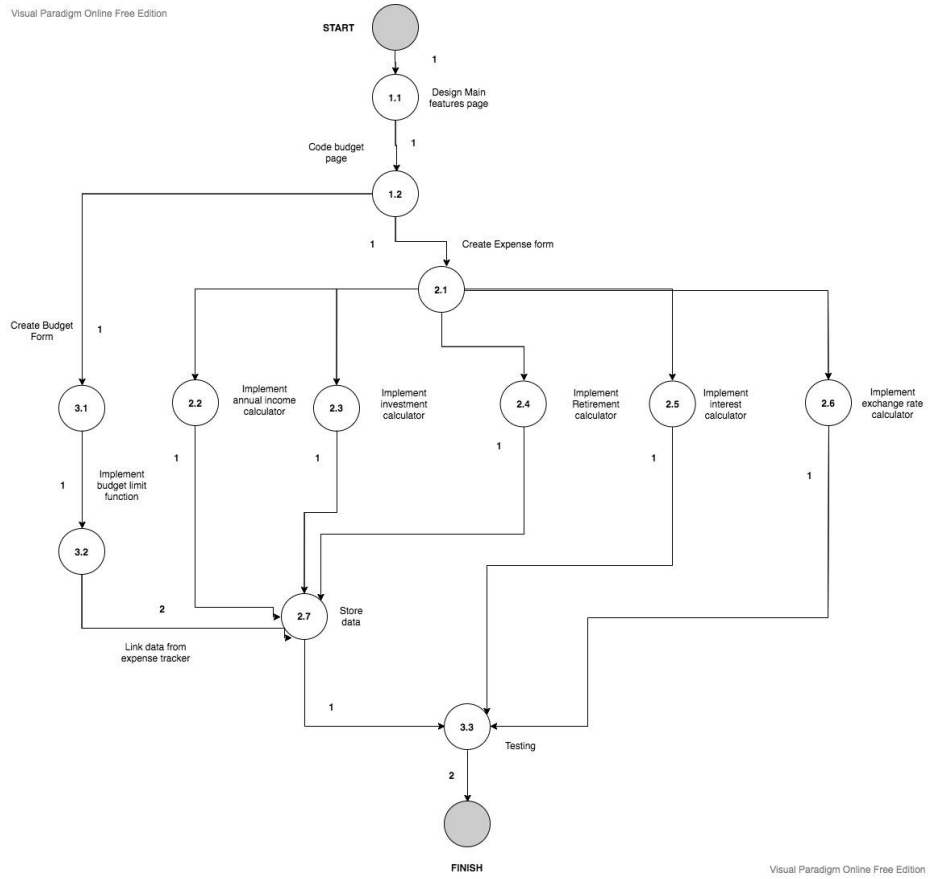
Activity	Time estimate (in days)	Earliest start time	Latest start time	Slack
1.1	1	0	0	0
1.2	1	1	1	0
2.1	1	2	2	0
2.2	1	3	3	0
3.1	1	4	4	0
3.2	3	5	5	0
4.1	1	2	6	4
4.2	1	3	7	4
4.3	1	8	8	0
Finish		9	9	0

<u>Critical Path:</u> 1.1, 1.2, 2.1,2.2,3.1,3.2,4.3
Takes 9 days to complete

Milestones

1.1 Designed home page	3.2. Login validation set up completed
1.2. Coded home page	4.1. Designed profile interface
2.1. Created registration form	4.2. User profile form completed
2.2. Terms & Conditions Setup completed	4.3. Testing completed
3.1. Created login form	3.2. Login validation set up completed

Phase 2: Main Features



Activity	Time estimate (in days)	Earliest start time	Latest start time	Slack
1.1	1	0	0	0
1.2	1	1	1	0
2.1	1	2	4	2
2.2	1	3	5	2
2.3	1	3	5	2
2.4	1	3	5	2
2.5	1	3	6	3
2.6	1	3	6	3
2.7	1	6	6	0
3.1	1	2	2	0
3.2	2	3	3	0
3.3	2	6	6	0

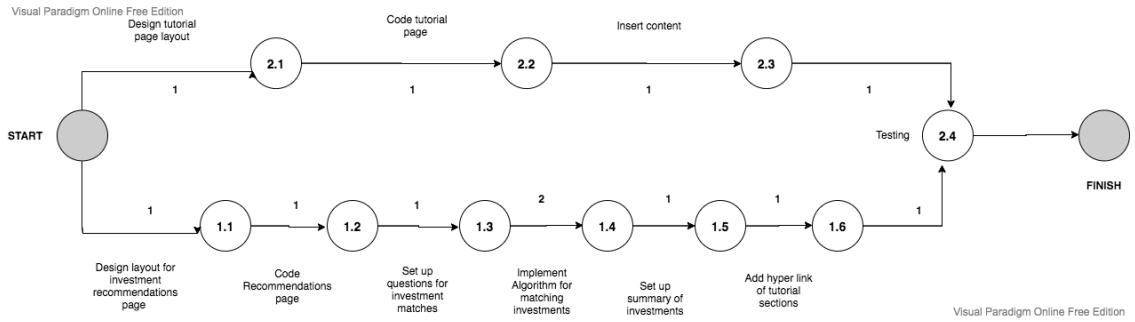
Finish	2	7	7	0

Critical Path: 1.1, 1.2,3.1,3.2,2.7,3.3
Takes 9 days to complete

Milestones

1.1. Designed Main features page	2.5. Implemented interest calculator
1.2. Coded budget page	2.6. Implemented exchange rate calculator
2.1. Created budget form	2.7. Stored data
2.2. Implemented annual income calculator	3.1. Created budget form
2.3. Implemented investment calculator	3.2. Implemented budget limit function
2.4. Implemented retirement calculator	3.3. Testing completed

Phase 3: Resource and Guide



Activity	Time estimate (in days)	Earliest start time	Latest start time	Slack
1.1	1	0	0	0
1.2	1	1	1	0
1.3	2	2	2	0
1.4	1	3	3	0
1.5	1	5	5	0
1.6	1	6	6	0
2.1	1	0	4	4
2.2	1	1	5	4
2.3	1	2	6	4

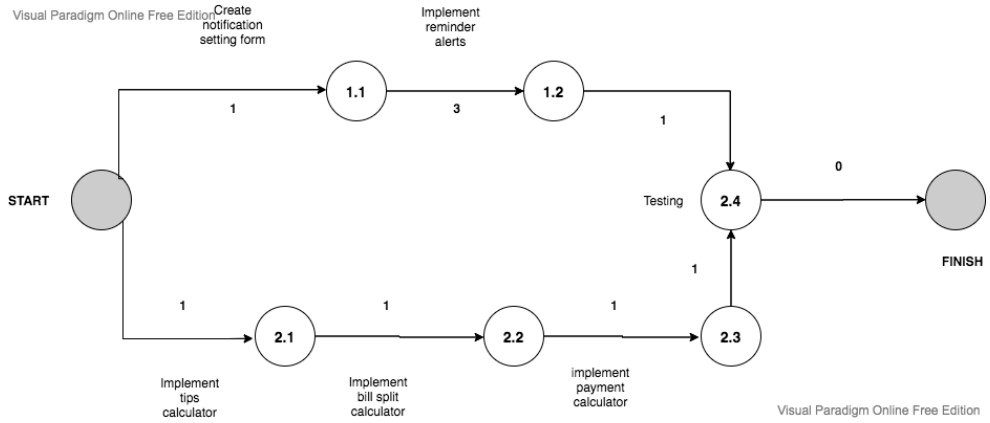
2.4	1	7	7	0
Finish	1	8	8	0

Critical Path: 1.1,1.2,1.3,1.4,1.5,1.6,2.4
Takes 9 days to complete

Milestones

1.1. Designed layout for investment recommendation page	1.6. Hyper links connected to tutorial section added
1.2. Coded recommendation page	2.1. Designed tutorial page layout
1.3. Questions for investment matches completed	2.2. Coded tutorial page
1.4. Implemented algorithm for matching investments	2.3. Content inserted
1.5. Investment summary setup completed	2.4. Testing completed

Phase 4: Miscellaneous



Activity	Time estimate (in days)	Earliest start time	Latest start time	Slack
1.1	1	0	0	0
1.2	3	1	1	0
2.1	1	0	1	1
2.2	1	1	2	1
2.3	1	2	3	1
2.4	1	4	4	0

Finish		5	5	0

Critical Path: 1.1,1.2,2.4
Takes 5 days to complete.

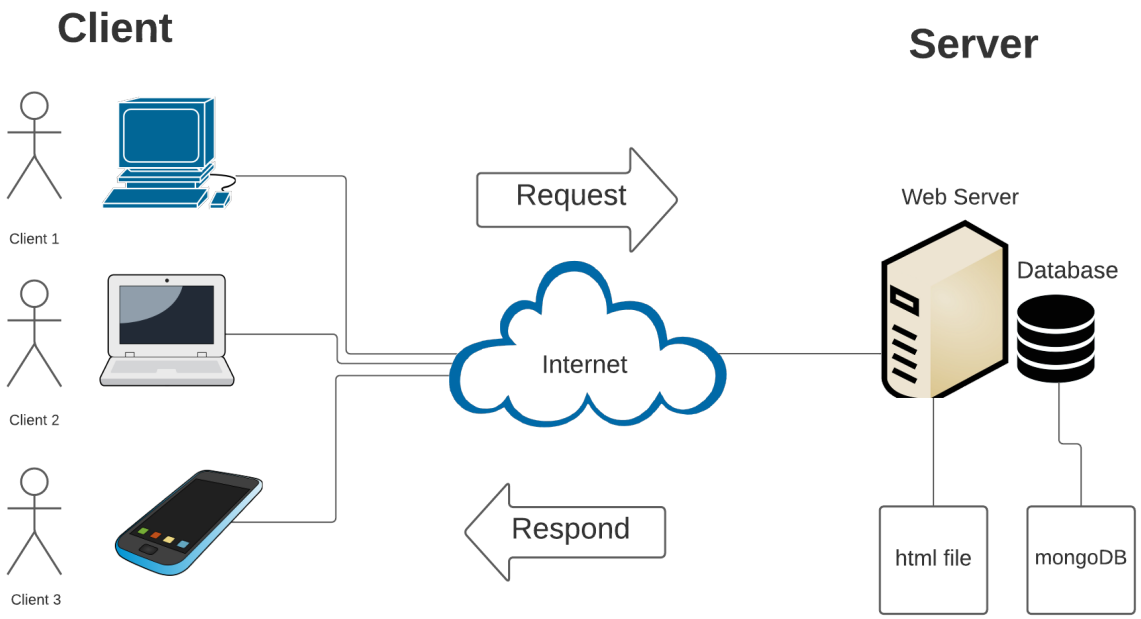
Milestones

1.1. Created notification setting form	2.2. Implemented bill split calculator
1.2. Implemented reminder alerts	2.3. Implemented payment calculator
2.1. Implemented tips calculator	2.4. Testing completed

9 Architectural Design (Changes made)

The primary architectural style that we will be using is the Client-Server model. We will have a web server and database, which the client can request access to in order to update it. A client can be defined by various devices (computer, laptop, phone) that are currently accessing the web app. Whenever information is updated, the client will send it to the server and if everything is in order, the server will then respond and update the information on the website to confirm a successful request. The Web server is represented by the HTML file of the page that the user is on. The Database is represented by MongoDB, which is what we are using to store all of our information.

We decided to go with this model because it works well for the type of application that we are developing. It allows us to have multiple clients, who can each request access to specific servers in order to do various tasks (add income source, add payment reminder, etc). For example, if the user wanted to add an additional expense to their budgeting section, they would have to access the budgeting.html file in order to view the user interface. From there, they would be able to input their new expenses and press a button to confirm them. This would then send a request to MongoDB, where that information would be stored. The server would then respond by updating the user interface on the budgeting.html to reflect the updated information.

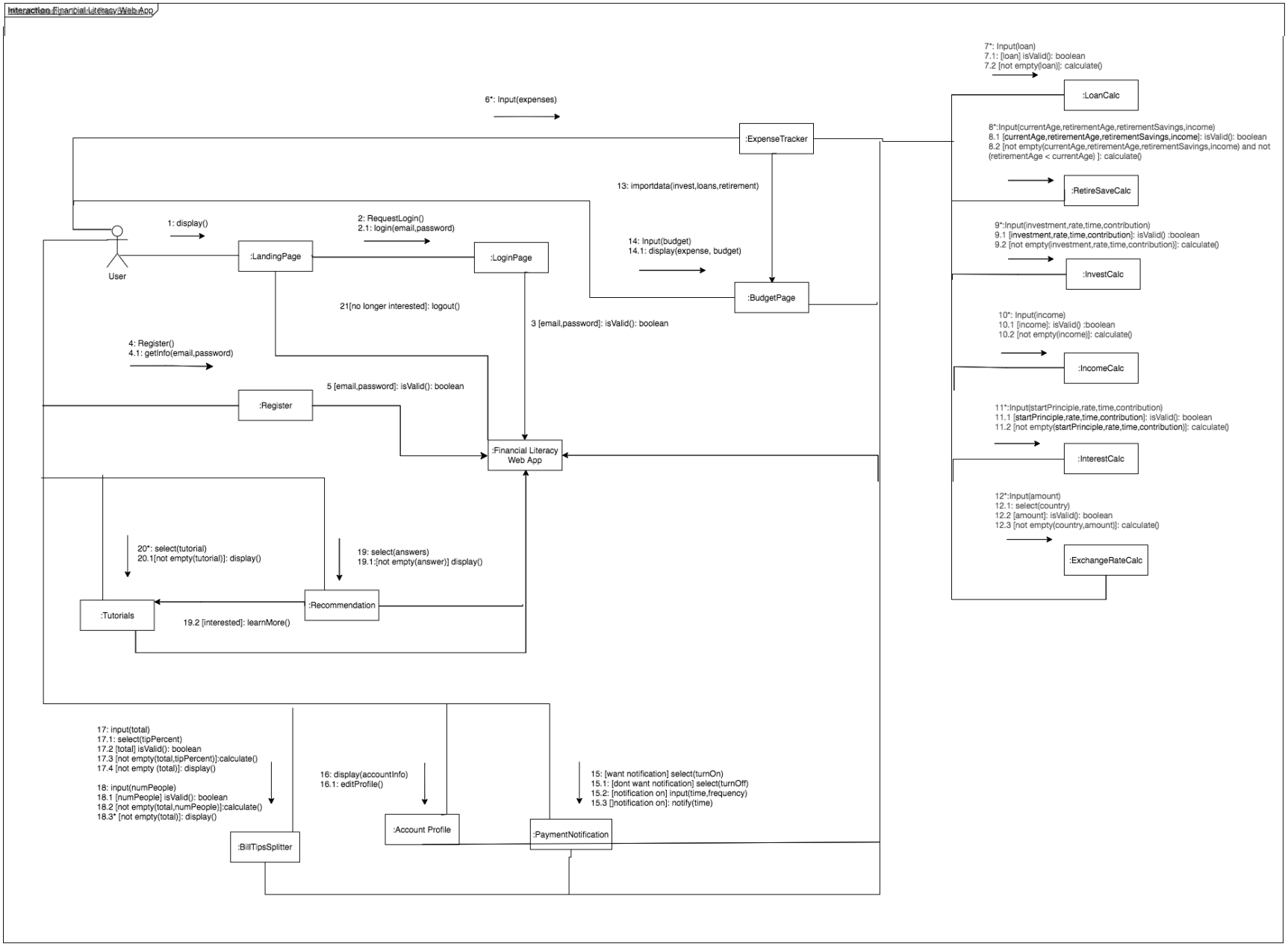


Changes were made to this section, compared to the original report. Before we were using a layering model for our architecture design but we decided to change that to a client-server model. This design better illustrates how our system is to be architectured by representing the interactions between the clients and the server more clearly. The written portion of this section was also updated to match the new design diagram.

In the architecture pattern, the users are the client. Users can use their desktops, laptops, or mobile devices to access the Money Ωmega web page. Clients can access the web pages and send requests to the web page via the internet. The two servers, the webserver and the database, then fetch these instructions and respond back to the client. The Web server is the Html files and the database server is MongoDB.

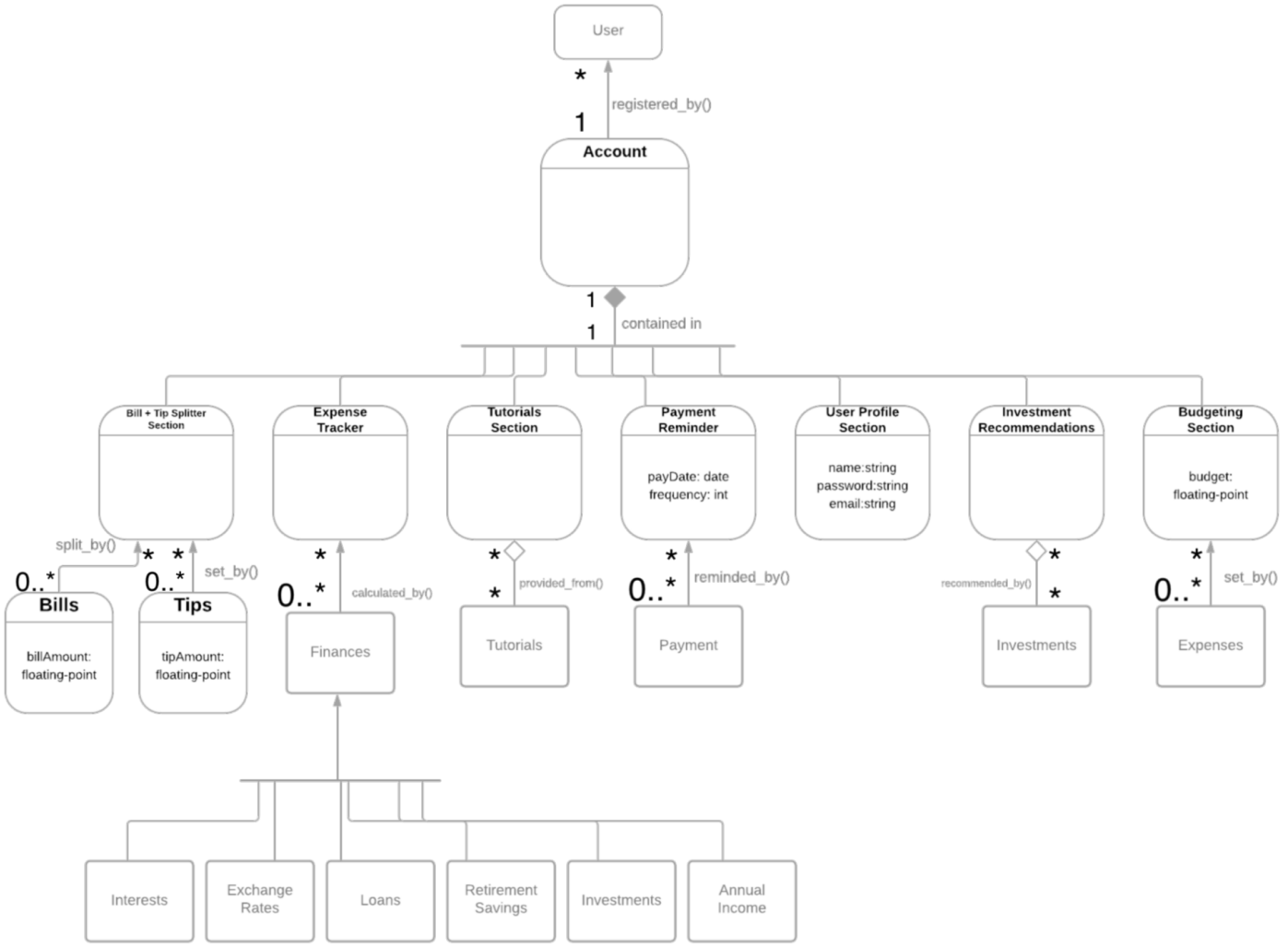
10 Detailed Design

10.1 Interaction model (Changes made)



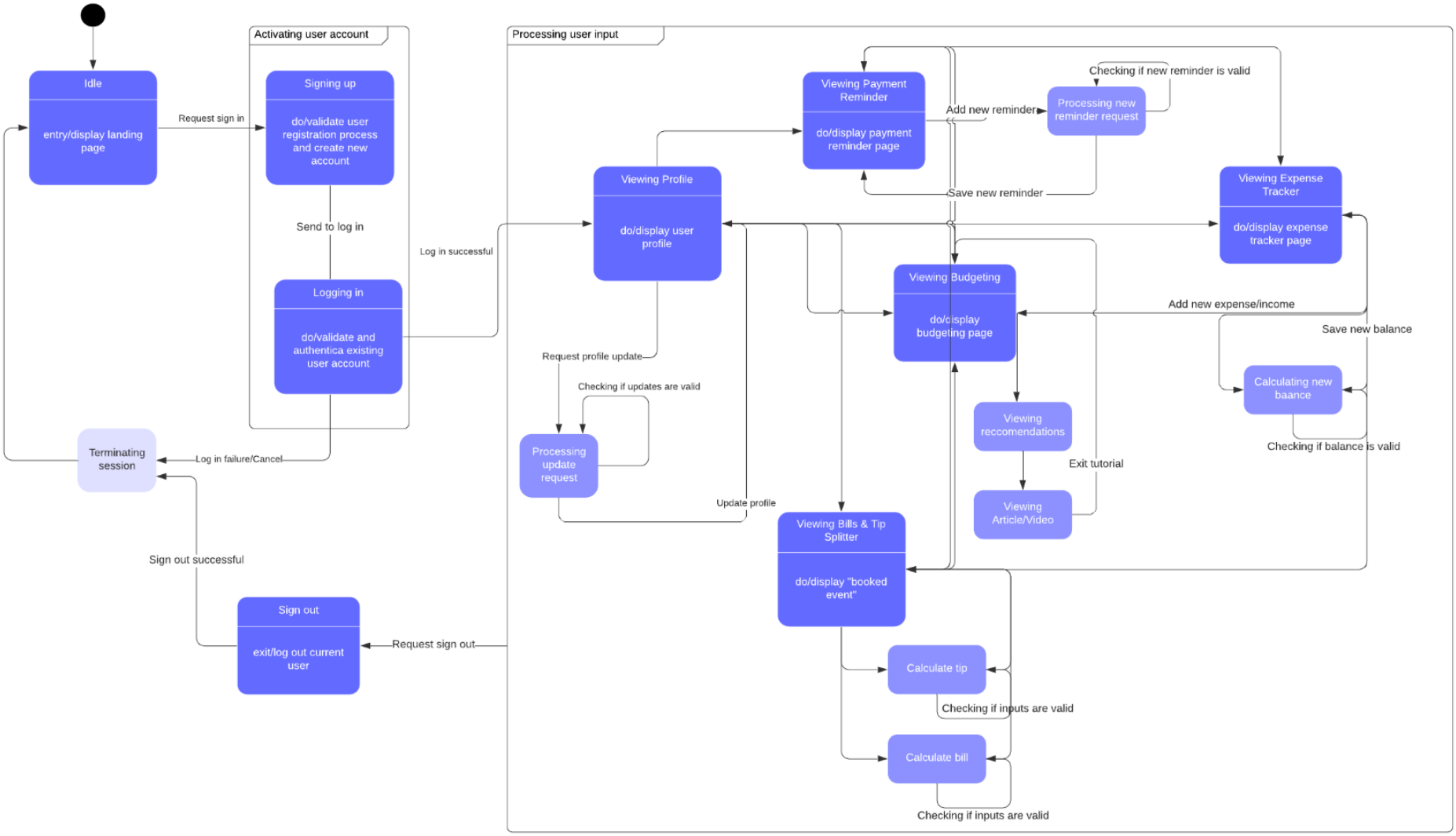
Changes were made to the interaction model as suggested by the feedback given. The flow of interactions such as registration is now starting from the actor and continues on to the “Financial Literacy Web App” object. Step ordering is also rearranged so that inputting into “LoanCalc”, “RetireSaveCalc”,”InvestCalc”,”IncomeCalc”,”InterestCalc” and “ExchangeRateCalc” can be done separately and not in any particular order

10.2 Design class model (Changes made)



Changes to the class diagram were made according to the feedback. Before, the class diagram did not reflect the register() method in the interaction model. In the updated version of the class diagram, the registered_by() node was added between User and Profile so that it could reflect the register() method in the interaction model.

10.3 State machine model



11 Test Design

Test Cases	Steps	Input	Expected Result
1	Register Account		
	Enter account information	email:name@email.ca Password: secret123	User can enter email and password
	Read Terms and Conditions	Click checkbox	User agreed with Terms and Conditions
	Click submit button		User account is created and redirected to home page

Test Cases	Steps	Input	Expected Result
2	Register Account - invalid information		
	Enter account information	Email:nonexist@email.ca Password:	Users can enter (incorrect) details or leave details empty
	Read Terms and Conditions	Do not click the checkbox	User does not agree to the terms and condition
	Click submit button		View Error Message, remain on the page

Both of these cases were tested manually by making registration as well as tested similarly to the labs with unit testing (jest and cross-env).
When testing the successive case we inserted User information such as name, email, username and password. We clicked the checkbox and the submit button and were able to be redirected to the home page.
We also tested the failure case where the information was missing such as the password or the checkbox for the terms and conditions was not checked. In the form we set required inside the input tags, this prevented the page from redirecting to another page. Additionally, we tested for if a username was already taken, the page gets directed to an error message page.
Screenshots of the unit test case will be included with the login test cases.

Test Cases	Steps	Input	Expected Result
3	Login		
	Enter login information	Username: User Password: secret123	Users can enter username and password
	Click submit button		Redirected to home page

Test Cases	Steps	Input	Expected Result
4	Login - Invalid information		
	Enter login information	Username: User Password: incorrect	Users can enter (incorrect) details or leave details empty
	Click submit button		View Error Message, remain on the page

These test cases were tested manually as well as unit testing. First, when we entered the username and passwords and clicked submit, we were able to redirect to the home page. When testing for the failure case we tried entering the incorrect password and redirection was impossible, the same goes for entering a non-existing username.

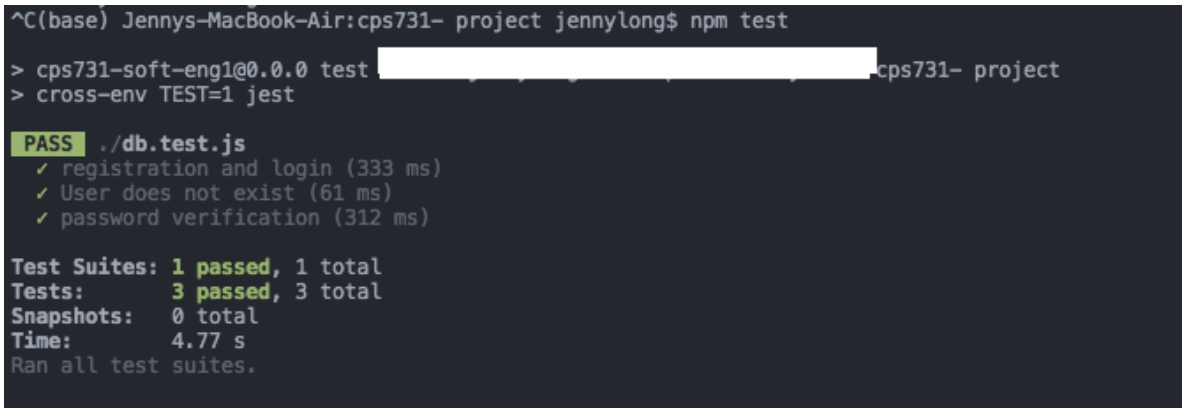


Figure 11: Unit testing for login and registration

Test Cases	Steps	Input	Expected Result
5	View and update profile information		
	Enter new profile changes	Email:nonexist@email.ca First Name: Bob Last Name: Long	Users can enter a new name, or email,
	Click submit button		Updated info is displayed

Test Cases	Steps	Input	Expected Result
6	View and update profile information - invalid input		
	Enter new profile changes - invalid information	Email:nonexist@email.ca	Users can enter (incorrect) details or leave details empty
	Click submit button		View Error Message, remain on the page

For the successive case, we tested this by entering a new email, first name and last name and clicking the submit button. The page automatically refreshes and the updated information appears.

For the failed case, we just remain on the same page.

Test Cases	Steps	Input	Expected Result
7	Input Expense		
	Enter expenses	Phone bill: 40 Transportation: 100 Rent: 1200 Food:200	Users can insert expenses
	Click submit button		Pie chart is displayed with all of the expenses inputted

For this case, we input values and the pie chart was able to display each expense in different colours.

Test Cases 8-9 cover calculating annual income, investments, retirement savings, loans, interests, exchange rates, from the Textural Use cases

Test Cases	Steps	Input	Expected Result
8	Calculate		
	Enter calculation variables	Income: 2000 Income: 300	User insert values needed for calculation
	Click submit button		Correct calculation outputted I.e. Income: 2300

Test Cases	Steps	Input	Expected Result
9	Calculate - Invalid values		
	Enter calculation variables	Income: 2000 Income: three	User insert values needed for calculation
	Click submit button		Error message, Incorrect value outputted or unable to calculate. I.e. Income: 2000

We inputted some test values and were able to get the correct values, for failure when we leave an input blank, a message will prompt us to retry.

Test Cases	Steps	Input	Expected Result
10	Payment Notification Set up		
	Enter payment and notification information	Payment type: loan/tax/bill Payment date: M/D/Y	Users can enter payment type, payment date
	Click add new payment button		View setup success message, the page refreshed (form cleared)

Test Cases	Steps	Input	Expected Result
11	Payment Notification Set up - invalid input		
	Enter payment and notification information	Payment type: (empty) Payment date: invalid	Users can enter (incorrect) details or leave details empty
	Click add a new payment button		View Error Message, remain on the uncleared form

We tested this by entering the dates and names for the payment notifications and when we click the save button the notification was displayed on the left-hand side.

For failed cases, we left some of the options blank and that resulted in a redirection to an error page.

Test Cases	Steps	Input	Expected Result
12	Payment Reminder Notification		
	The system confirms notification set date		Notification date matching payment reminder is found in database
	The system sends notification to user's device	User responds and makes payment	View payment paid successful message, return to home page

Test Cases	Steps	Input	Expected Result
13	Payment Reminder Notification - failure to receive response from user		
	System confirms notification set date		Notification date matching payment reminder is found in database
	System sends notification to user's device	None	Send notification again after 1 hour. Repeat until the payment deadline.

Test Cases	Steps	Input	Expected Result
15	View Tutorial Articles & videos		
	Select an article or video	User clicks on article or video	User chooses a valid article or video
	System displays article or video		User is redirected to the article or video that they selected

Test Cases	Steps	Input	Expected Result
16	View Tutorial Articles & videos - Failure		
	Select an article or video	User clicks on article or video	User selects an invalid article or video
	System displays error message		User is displayed an error message and is asked to select a valid article or video

Both of these cases were manually tested by clicking on the articles and videos in order to ensure proper functionality. For the articles, the buttons that opened them were pushed to see if they linked to their respective articles. For the videos, the video players were tested to see if they played the proper videos and if video functionality (video speed, pause button, fullscreen) worked.

For the failure case, the articles were tested by trying to select a non-article element on the page. For the videos, we used a web browser that does not support HTML video in order to test that the error message was properly displayed.

Test Cases	Steps	Input	Expected Result
17	Investment Recommendation		
	Select the answer to the questionnaire	Answer: *selected answer from multi-answer*	
	Submit answer		System displays the recommended investment and additional link to tutorial/articles

Test Cases	Steps	Input	Expected Result
18	Investment Recommendation - Failure		
	Select the answer to the	Answer:	

	questionnaire	*selected answer from multi-answer*	
	Submit answer		System is unable to display the recommended investment and additional link to tutorial/articles

For the successive case, some of the questions were selected and an investment product was generated with a link that goes to the tutorials page.

For the failure case, when the system is unable to recommend any investment product an alert pops up that says "No Matches were found! Please try again!"

Test Cases	Steps	Input	Expected Result
19	Bill Calculator		
	Enter Bill total	User inputs total bill cost	Users can input their total cost in integers
	Enter number of people	User inputs total number of people	Users can input their total number of people in integers
	Individual Bill Cost		System outputs individual bill cost properly

Test Cases	Steps	Input	Expected Result
20	Bill Calculator - Failure		
	Enter Bill total	User inputs total bill amount	User inputs an invalid response for the bill total
	Enter number of people	User inputs total number of people	User inputs an invalid response for the number of people

	Individual Bill Cost		System outputs an error message and prompts User to input valid values

The testing for Bill Calculator is in combination with the tip splitter, it will be in the next pages

Test Cases	Steps	Input	Expected Result
21	Tip Calculator		
	Enter tip total	User inputs bill total	Users can input their total bill in integers
	Enter tip percentage	User inputs tip percentage	Users can input their tip percentage in integers
	Total Tip Amount		System outputs total tip amount properly

Test Cases	Steps	Input	Expected Result
22	Tip Calculator - Failure		
	Enter tip total	User inputs bill total	User inputs an invalid response for the bill total
	Enter tip percentage	User inputs tip percentage	User inputs an invalid response for the tip percentage
	Total Tip Amount		System outputs an error message and prompts User to input valid values

The following 4 test cases were all tested at the same time. For both calculators, if the proper input is entered, then they will display the expected results. This was done manually by using each input box to enter some example numbers to test the calculators. For the failure cases, we would leave certain input boxes blank and the system would display $NAN for the output, indicating to the user that they must fill out all the remaining fields before they can get their answer.

Test Cases	Steps	Input	Expected Result
23	Calculate interest		
	Enter calculation variables	Interest Rate: 2% Principal: $4000 Periods: 2 Time (y): 4	User insert values needed for interest calculation
	Display result		Correct calculation outputted Ex. Total interest is: $213 Total amount is: $23023

Test Cases	Steps	Input	Expected Result
24	Calculate - Invalid values		
	Enter calculation variables	Interest Rate: 2% Principal: $4000 Periods: Time (y): adx3	User insert values needed for interest calculation
	Display result		Calculator will display an error message or refuse to provide a proper output. Ex. Total interest is: $NAN Total amount is: $NAN

The 2 cases above were tested together manually. The inputs for the interest calculator would be entered and the web app would display the expected output if the user entered the appropriate inputs. For the failure case, it was manually tested by leaving certain input boxes blank and the result was that the display would show a $NAN for the output, indicating that the inputs were invalid.

NFR TEST CASES

- **Capacity:** System must be able to store and load all information inputted by its users.
- **Usability:** Users must be able to use the application without prior knowledge of money management and/or budgeting.

Test Cases	Steps	Input	Expected Result
1	Capacity		
	Enter large amount of user information into system	100 payment reminder notifications, 100 new expenses, 100 new income, and 100 new loans	System successfully saved all inputs
	View and load information		System loads all saved information

Test Cases	Steps	Input	Expected Result
2	Capacity - fail		
	Enter large amount of user information into system	100 payment reminder notifications, 100 new expenses, 100 new income, and 0 new loans - database storage limit reached	System data storage limit reached, did not save some or all inputs
	View and load information		System partially loads some information

For the success case, we tried making several accounts and had no issues in doing so. Obviously, due to time constraints, it is not feasible for us to create 100 accounts.
Notifications wise we are able to create as many notifications as we want. The only thing is that the top 3 notifications will be displayed rather than the entire notifications from the storage. Only after removing them will you see the others in order.

Test Cases	Steps	Input	Expected Result
3	Usability		
*10 random volunteers will be test users	Attempt user sign up	Email: nonexist@email.ca Password: 12345	Sign up successful, account created
	Navigate the web application		User successfully navigated and used the system

Test Cases	Steps	Input	Expected Result
4	Usability - fail		
	Attempt user sign up	Email: nonexist@email.ca Password: 12345	Sign up successful, account created
	Navigate the web application		User failed to navigate and use the system correctly

For usability the fonts are large enough to read and signing up is pretty straightforward. If the user inputs invalid information for registration there will be a small message that indicates that the inputs are required. After signing up, the user will be redirected to the homepage. They can easily navigate to other pages by clicking on the navigation bar.

12 Review of Project Plan

The project plan that was proposed at the start involved splitting the site into various sections (User Profile, Payment Reminder Section, Budgeting Section, Tutorials Sections and Bill & Tip Splitter Section). All of these remained in the final version of our web application, however, some additional sections were also added. Expense Tracker is a new section that was originally a part of Budgeting and allows the user to manually input their income/expenses as well as use various calculators to help them manage their finances. The other new section that was added is called Recommendations, which gives the user investment recommendations based on their current finances and risk tolerance. This section was originally a part of the Tutorials section but we decided to expand it into its own section, as its functionality increased.

Due to time constraints, a smaller team size compared to the rest of the class and a lack of prior web-development experience, there were several features that had to be removed for the final application. The Payment Reminder section was originally supposed to notify you via a web browser or email, whenever you had a payment approach. This feature was not implemented and instead, the “upcoming payments” section is used to show users their approaching payments. The account profile was originally going to be capable of updating the user information but this functionality was removed for the final version. One last change made was to the Budgeting section. The pie graph that shows the different expenses was supposed to show the names of those expenses but now it only shows the exact value of the various expenses, since it was easier to implement this way.

The process model that we decided to use for this project was the incremental development model. We originally chose this model because it allowed for the application to be broken down into several sub-projects that we worked on overtime. Our plan was to first work on the User Profile and Registration, then the Budgeting section and then whatever was remaining after that. This suited us well because it ensured that we would have a working site very early into the project, with the remaining features being added to the site as the deadline approached.

At the start, we were able to adhere to this model and focused on developing the basics of the site first such as the landing page and login screen. As the project progressed, we decided to deviate from this plan and instead split up the tasks between our group members. We decided to do this so we could quickly finish up all functionality on the site before the deadline. Each person was assigned 2 or 3 different sections of the site such as the Expense Tracker or the Payment Notification, with everyone working on these sections at their own pace. Instead of features being added incrementally and on a set order, they were simply added whenever someone finished their specific task(s).