Team 02 - Techincal Documentation

Limoney Technical Documentation

SW Engineering CSC648-848-05 Fall 2024

Limóney (Financial Budgeting WebApp)

Team 02

Name	Role
Emily Perez	Team Lead
Ishaank Zalpuri	Database Administrator
Andrew Brockenborough	Technical Writer
Dani Luna	Github Master, Database Administrator, Frontend Lead
Jonathan Gonzalez	Software Architect
Gene Orias	Backend Lead

Milestone 1

6/17/25

Milestone	Version	Date
Milestone 1	Version 1	6/17/25

Table of Contents

Executive Summary

Limóney is an AI-enhanced budgeting web application designed to help users manage their finances all in one platform. We live in a time where many people struggle with managing recurring expenses, forgotten subscriptions, and inconsistent saving habits. Because of this, there's a growing need for financial tools that are not just useful, but also simple and easy to use. Our motivation is to keep the application both functional and accessible so everyday users can stay on top of their finances without feeling overwhelmed by data or complex interfaces.

Inspired by *EveryDollar* and *Rocket Money*, what sets *Limóney* apart is that it combines the best features and usability of both apps into one platform, while also integrating AI to provide personalized, intelligent budgeting insights, which is something neither of those apps currently offer. The built-in assistant provides tailored spending suggestions, identifies unnecessary expenses, and recommends ways to save based on the user's financial habits. The result is a smarter, more intuitive budgeting experience that helps users improve their financial well-being over time.

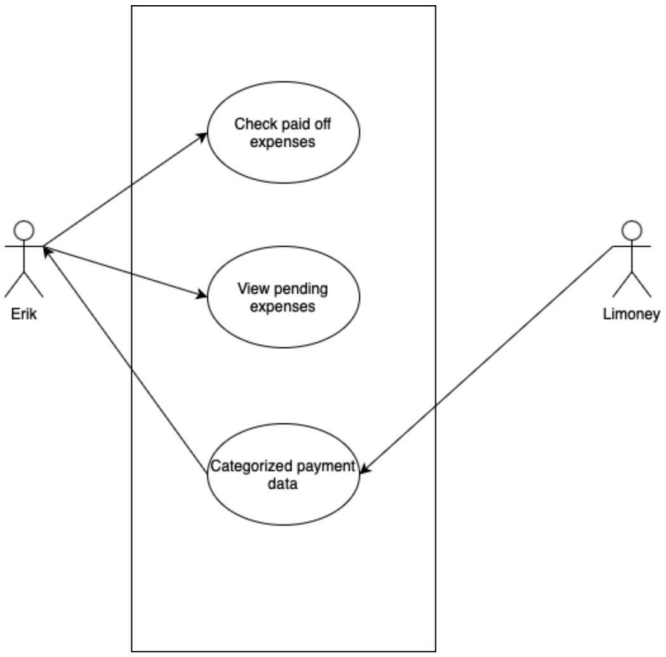
Main Use Cases

Main Use Case 1

Create Budget
The user defines one or more income sources and allocates funds into spending categories. If allocations exceed income, the system prompts the user to adjust the amounts.

1. Checking which specific credit card expenses got paid off

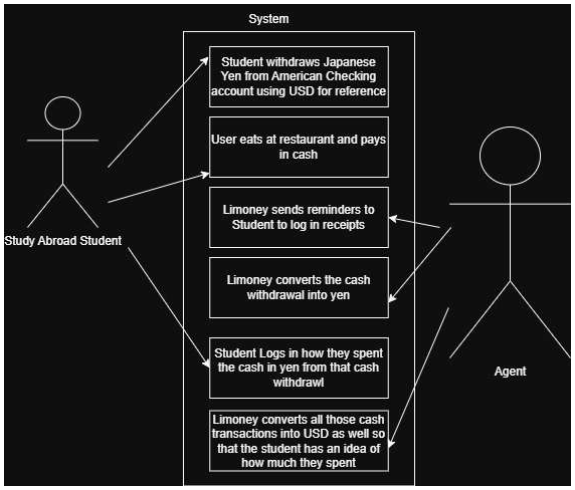
- Actors:
  - Erik (Student managing Credit Cards)
  - Limoney (Company)
- Assumptions:
  - Erik has multiple credit cards.
  - He has made recent payments, but is unsure of which expenses got paid.
  - Erik has connected his bank info to Limoney to keep track of his expenses.
- Use Case: Erik made a credit card payment, but is unsure of what was actually paid. He opens up Limoney, and the app provides a clear list of which charges were paid off. For ex: his textbooks, school supplies, and what is still needed. Being able to see which expenses have been paid has given Erik peace of mind and helps him plan out his next payments with confidence.
- Benefits for Erik :
  - Erik sees which expenses have been paid.
  - Helps Erik decide which expenses to pay off next.
  - He feels more in control of his credit and his spending.


Check paid off expenses
View pending expenses
Categorized payment data
Erik
Limoney

Main Use Case 2

Tracking multiple reimbursements and predicting balances

- Actors: Student A, Student A's group of friends
- Assumptions: Student A is somewhat financially stable
- Use Case: Student A goes eating out with a group of friends once a month. Since they are a large group, it is usually difficult to split the bill among all of them. He pays the bill and expects his friends to reimburse him later, but everyone gets busy and can't keep track of how much needs to be paid back. The next time they go eating out, he puts the meal on his card. The student goes to the app right away and logs in the transaction. He also has the option of splitting the bill by a number of friends or splitting it by what they order and adding their names. These splits are called reimbursement requests, and the student can even send them to his friends as reminders. The request will be active until it is complete, and the money paid back will reflect on his account balances.
- Benefits for Student A: Student A can keep track of who owes money back to him
- Benefits for Student A's Friends: They are held accountable and become more financially responsible

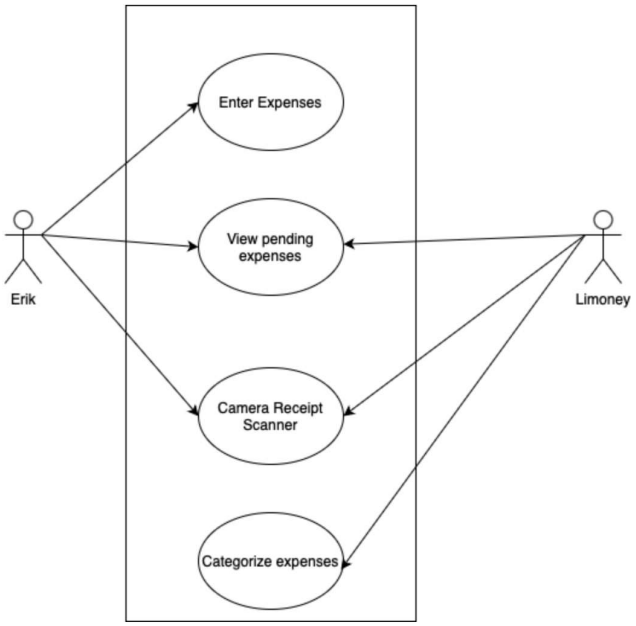


Main Use Case 3

Log Transactions
Users import or manually record transactions; the system auto-categorizes based on rules, with a user override option.

1. Making it a habit to log in the Receipt

- Actors:
  - Erik (Busy College Student)
  - Limoney (Company)
- Assumptions:
  - Erik juggles school and work on a packed schedule.
  - He stores physical receipts but forgets to log them at the end of the day.
  - Erik prefers to manually enter his expenses for privacy reasons.
  - He does not use updated bank tools.
- Use Case: Erik tries to keep track of his spending, but with work and school, he tends to forget about his receipts, which pile up. Logging the receipts by hand feels like too much. One day, Limoney sends out a reminder and suggests that he can scan the receipts via his camera instead. Erik tries it and is surprised by how quickly and efficiently it works. Erik is now logging expenses through Limoney because it provides a simple way to keep track of his receipts and has become a nightly routine.
- Benefits for Erik :
  - Logging receipts becomes fast and easy with the camera scanner.
  - Helpful reminders help Erik stay on track.
  - The process fits into his busy lifestyle and does not require advanced tech or skills.


Enter Expenses
View pending expenses
Camera Receipt Scanner
Categorize expenses
Erik
Limoney

Main Use Case 4

1. Security concern

Actors:

- Andrew
- Limóney

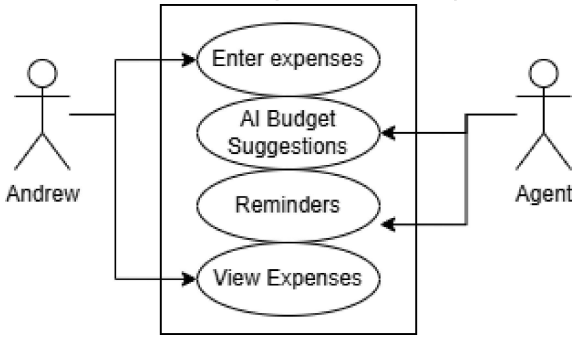
Assumptions:

- The user manually enters receipts
- They don’t have the latest technologies
- They don’t want to use apps that access their financial information
- Use Case: Andrew is serious about his privacy and refuses to use financial apps that require access to his bank account or store data externally. For years, he has been using spreadsheets to track his spending manually. The process has become overwhelming to do. He discovers Limóney, a secure, AI-enhanced budgeting app designed for users like him. Limóney allows users to enter transactions manually and get personalized budgeting insights without ever syncing financial accounts. Andrew starts using Limóney’s interface to log purchases. The built-in AI analyzes his patterns and suggests simple ways to save like avoiding frequent late-night food orders. Andrew gets skeptical on AI and has the choice of turning off the AI suggestions. What surprises Andrew most is how nice it feels: he stays in control, but still benefits from smart financial planning tools

Benefits for ... :

- Manages finances without linking sensitive financial accounts.
- Gets smart recommendations based on spending habits
- Fits into Andrew’s low-tech lifestyle; adaptable

- Knows his data stays local and private

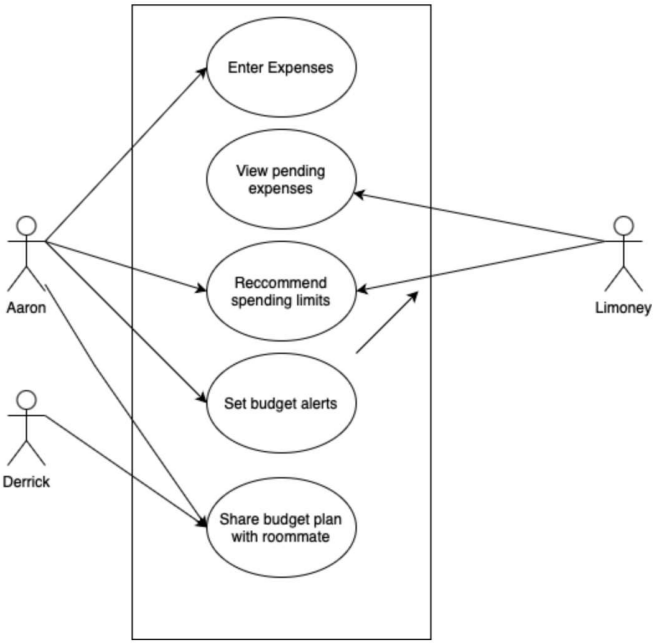


Main Use Case 5

Receive AI Suggestions
The system analyzes spending patterns daily and sends personalized tips, such as reducing recurring subscriptions or reallocating unused budget.

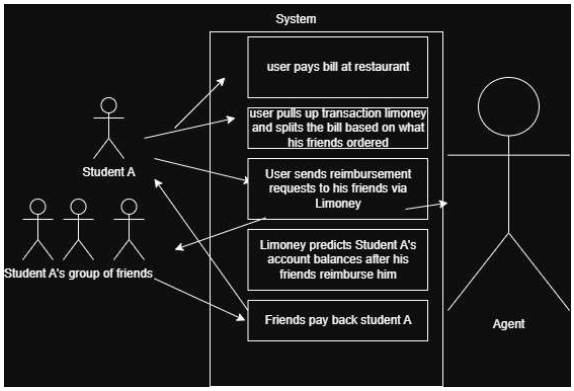
1. Budgeting: Cut down on expenses

- Actors:
  - Aaron (College Student)
  - Derrick (Aarons Roomate)
  - Limoney (Company)
- Assumptions:
  - Aaron and Derrick have separate bank accounts and budgets in Limoney
  - Aaron has connected his banking info to the Limoney App
- Use Case: On a friday evening, Aaron opens the Limoney App to figure out why he keeps running low on money every month. The Limoney app, quickly points out that his dining expenses are higher than usual and suggests a closer look. Aaron notices that he has several delivery charges that had been adding on. The Limoney App recommends that Aaron sets up a weekly limit which will allows him to see how much he can potentially be saving. Relieved by this, Aaron sets a spending alert and feels in control of his budget and shares it with Derrick. Aaron and Derrick decide that cutting back on dining expenses can save them a lot so they begin cooking meals at home together and decrease their spending.
- Benefits for Aaron:
  - He understand exactly where his money is going.
  - Aaron can set spending alerts and adjust his budget in real time.
  - Feels confident and is more proactive with his savings.


Enter Expenses
View pending expenses
Reccommend spending limits
Set budget alerts
Share budget plan with roommate
Aaron
Derrick
Limoney

Main Use Case 6

- Actors: Study Abroad Students
- Assumptions: Some Students struggle to adjust to foreign currency and want to get used to the system
- Use Case: Study abroad student goes to Japan. The country paper based, most restaurants only take cash. Whether the student opts to manually track expenses or link his bank account, he needs to withdraw cash and keep a track of how he spends it. It gets confusing however as he collects receipts and the currency rate changes every day. By the time he sits down to enter his cash transactions, he has to spend extra time converting currency. Instead of stressing out, he chooses to switch the currency on the app to Japanese Yen so he can get used to the currency abroad and accurately log how he used his cash withdrawal. At the end, he also converts all those transactions back into USD to get an idea of how much he is spending and what the Japanese economy is like
- Benefits for study abroad Student A: The Student keeps a track of his expenses abroad and gets used to the Japanese currency



Main Use Case 7

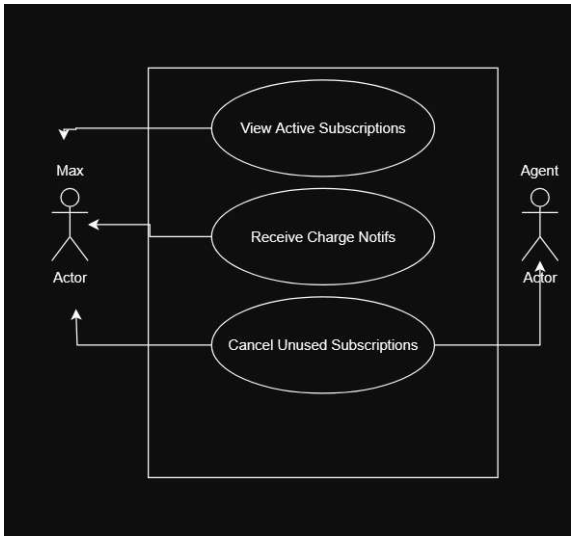
1. Tracking ongoing expense

- Actors:
  - Max who has a lot of active subscriptions
  - Max doesn't really know much about the digital landscape

- Assumptions
  - Max has a lot of active subscriptions

- Use Case:
  - Max is a busy college student juggling classes, a part time job, and taking care of his siblings. Lately he has been feeling very overwhelmed and doesn't have time to check his bank account. Max suddenly got a notification on his phone that he has been charged 20$ for a subscription he didn't even know he had. Max then downloaded the Limoney app which can help him manage his subscriptions. Instantly he gets greeted by the active subscriptions he has on right now! With a few taps those useless subscriptions have been cancelled!

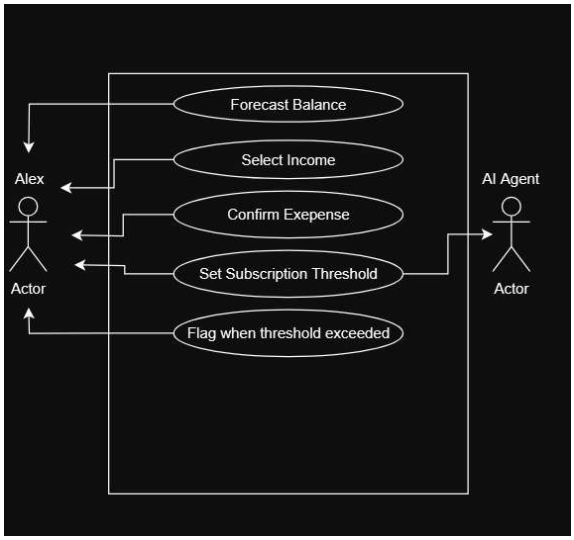
- Benefits
  - Prevents ongoing subscriptions as to easily see everything
  - Ability to see unwanted or unused subscriptions



Main Use Case 8

1. Predicting balances end of month

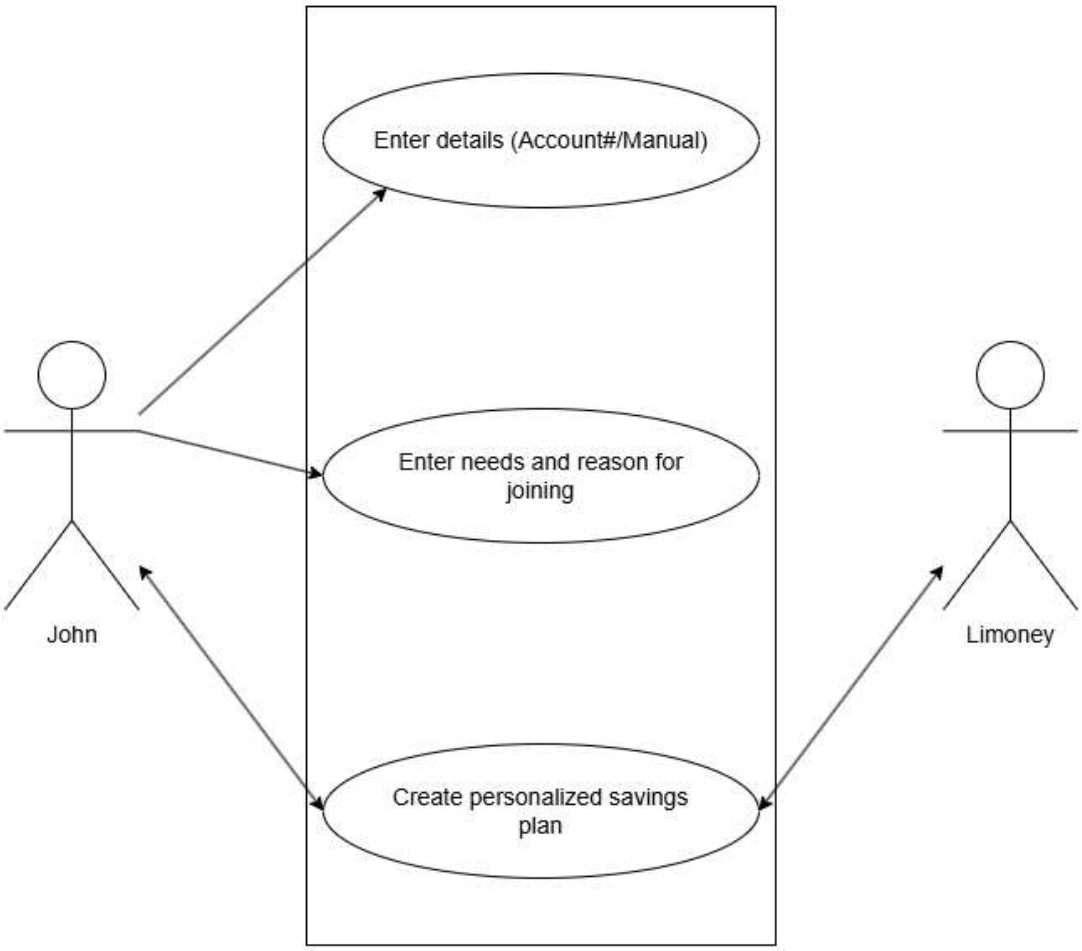
- Actors:
  - Alex who loves planning ahead
- Assumptions:
  - Alex has stability within expense and income
  - Alex updates their transactions regularly
- Use Case:
  - Alex, who is a planner by nature, wants to know if he will be financially comfortable by the end of the month. He opens the Limoney app and selects “Forecast Balance”. The app prompts him to select his income and confirms his current expenses like subscriptions. Alex then sets a threshold of 100$ within subscriptions so the app can flag Alex whenever he goes above that mark.
- Benefits
  - It completely avoids disturbance within the stability of expenses
  - If there are balance surprises you can see your subscription bills easily and adjust accordingly.



Main Use Case 9

Use Case 9: Signing up/Intro UI

- Actors: John(User) and Limoney(Company)
- Assumptions:
  - John wants help with budgeting his expenses
  - John finds Limoney ad interesting and clicks
- Use Case:
  - John is worried about his finances. He is concerned about going broke. Enter Limoney, a budgeting app he found in an ad. He is now signed up, and it is now asking him questions. It begins with why John is on the app. What does he need help with? (budgeting, managing expenses, investment, tracking subscriptions) Afterwards, it asks if John wants to share his data using his SS and banking info, or if he prefers to manually enter his info. If he chooses the latter, it takes him to fill out the information he needs help with (income, average expenses, current bills, etc.). He then receives a message welcoming him to Limoney and a well-wishing of “Happy Savings!”, with an immediate option to work with Limoney to start saving money.
- Benefits for … :
  - Users
    - Users feel welcomed and impressed by the application, giving them confidence that they will be taken care of
  - Company
    - The application will get more traffic through word of mouth, the more impressed the users are with its features
- Diagram:
  -

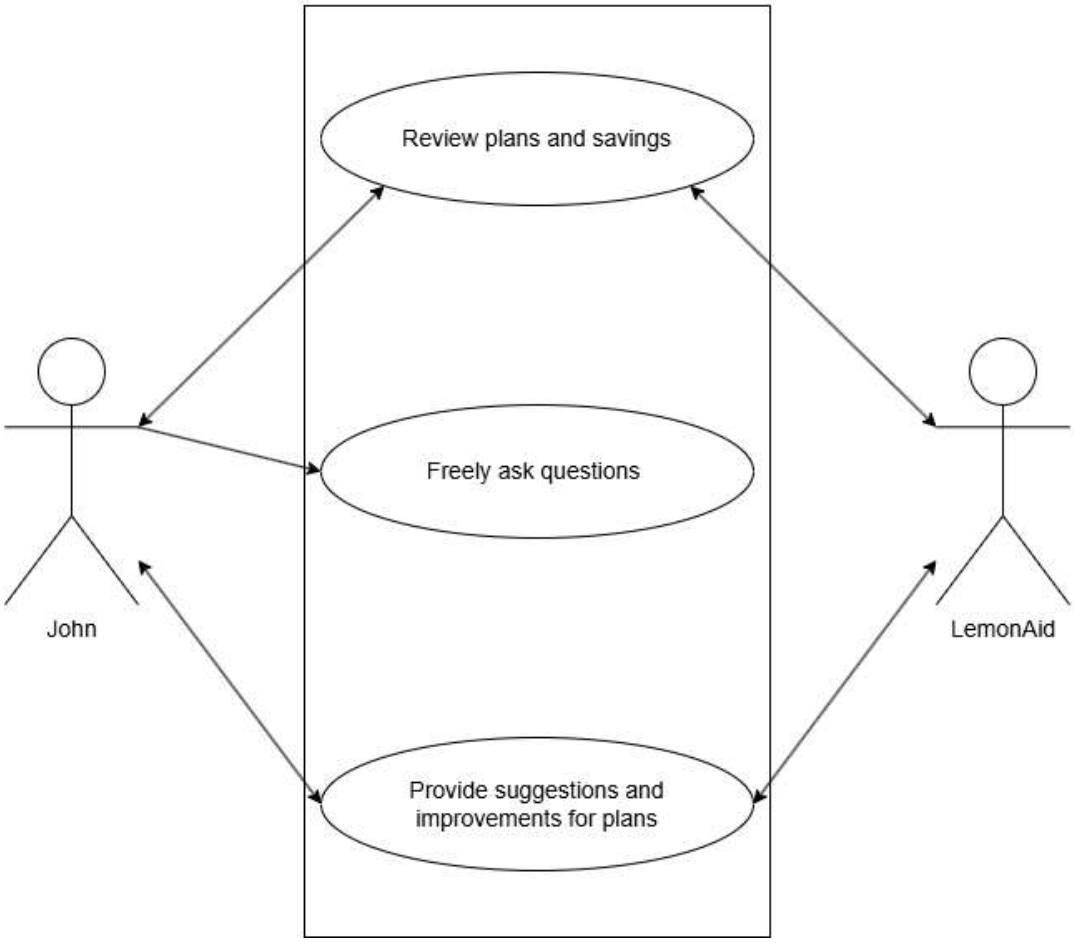


Modeling Diagram for Use Case 9

Main Use Case 10

Use Case 10: Chatbot/AI assistance

- Actors: John (User) and LemonAid(Chatbot)
- Assumptions:
  - John is familiar with the concept of a Chatbot and feels comfortable using it, understanding it is only for reference and will not be 100% accurate
  - John likes the idea of having his entries analyzed and would like to have an outside perspective on his savings without sacrificing his privacy
- Use Case:
  - John has successfully signed up and is enjoying using Limoney. However, all the technical jargon and the routine of inputting and reading numbers gets tiring. He wants to take a bit of a break and wants to be able to discuss his finances with others. He opens LemonAid, the company's financial aid assistance, and asks it for help. To streamline the process, LemonAid asks John how exactly it can help him today, while providing several recommendations that he can click on or directing him to enter his own question in the text field.
- Benefits for ... :
  - Users
    - Users shall be able to discuss their finances with an AI assistant, allowing them not only to receive feedback from it, but also allowing them to think more deeply on the subject matter they are asking about. All while maintaining their privacy and not being required to share their finances with others.
  - Company
    - The company shall be able to streamline the feedback process and make communication much easier. If users choose to share their chat information, the company can use it to further develop the AI and train it to be prepared for more situations.
- Diagram:
  -



Modeling Diagram for Use Case 10

List of Main Data Items and Entities

Hierarchy: Blue - Main Items > Purple - Sub-Items > Orange - Item Filters

- Users - Defines all the different types of users of the app
  - Customers - The basic user type, the average person, part of the target audience
  - Premium - Possible special customers with more privileges or access to benefits
  - Testers - Special users and customers that are given certain privileges and compensation in exchange for trying to break the site in every way they can think of
  - Administrators - Pinnacle of users that have the capability to directly change things site-wide or quickly take down the site in order to perform maintenance, prevent leaks, etc.
- Banks - Defines all the banks associated with users
  - Credit - The amount of money a user currently owes to their bank
  - Total - Total amount of money a user owes to all of their banks
  - Balance - Amount of money a user currently has stored at a bank
    - Total - Total money stored at all of their banks
    - Account Type - Special balance property that filters in which kind of account the user has their money stored in, checking, savings, cd, etc.
- Tasks - Certain tasks assigned to each user account
  - Recommended - The next action that a user should take in order to advance their account
  - Type - Defines certain filters for tasks
    - Savings - Tasks related to saving money, like cutting a subscription, paying off a bill etc.
    - Investing - Related to moving money and growing it, such as using an investment app or opening a cd account
    - Setup - Related to setting up the account for more accurate results
    - Testing - Special tasks for testers to complete, allows for mass testing of certain unstable builds and features

        - Admin - Rare tasks for administrators to act on in order to improve the app or change things site wide
    - Completed - List of tasks completed by an account to be used in calculating rewards and levels, etc.
- Rewards - An extra incentive to make saving money more interactive and rewarding. Rather than waiting, receive immediate gratification for actions.
    - Level - A tracker of sorts that allows users to feel and see their growth and progress
    - Rating/Ranking - Potentially add a visually competitive aspect to making money
    - Redeem - Rewards for using the app and completing tasks, increasing user interaction
- Algorithms - Processes that can be run to fulfill user needs
    - Calculate - Simple processes that can be used to calculate different values, such as total expenses, savings, loss, etc. Can be used to further determine user tasks
- Limoney - Filter for all things related to money
    - Income - Value representing how much a user earns and where from
        - Total Income - Value representing the total amount a user is making, can be filtered to show daily, monthly, and yearly
    - Expenses - Value representing how much a user is spending and where
        - Total Expenses - Represents the total amount of money the user is spending, once again, daily, weekly, monthly, and yearly
    - Savings - A value that tracks how much users have saved since joining and using Limoney
    - Investments - An advanced tracker that organizes non-liquid assets, cannot provide in-depth feedback because web-scraping is illegal

Initial List of Functional Requirements

End User Functionality -

1.1 - Register a new account (email/password)

1.2 - Log in securely to account

1.3 - Recover / Reset password

1.4 - Connect bank account for transactions

1.5 - View dashboard (Overview of budget / goals)

1.6 - Add / Edit / Delete income sources

1.7 - Add / Edit / Delete expenses

1.8 - Create monthly budgets for difference spending categories

1.9 - View real time budget strategy

1.10 - Set saving goals with specific (amount / deadlines)

1.11 - Track progress against goals

1.12 - Receive personal AI-generated advice

1.13 - View spending history (weekly / monthly / yearly)

1.14 - Set spending limits (specific categories)

1.15 - Receive notifications when reaching spending limits

1.16 - Receive notifications for upcoming (bills / subscriptions)

1.17 - Categorize transactions manually

1.18 - Modify transaction category

1.19 - View all subscriptions on one place

1.20 - Set notification type and frequency

1.21 - AI assistance to reduce spending

1.22 - Customize interface themes (dark mode)

Limoney AI assistant Functionality -

1.1 - Analyze user transactions and identify spending patterns

1.2 - Recommend budget adjustments

1.3 - Suggest expense categories for unknown transactions

1.4 - Identify / flag unusual transactions

1.5 - Suggest specific ways to save based on habits

1.6 - Suggest saving opportunities such as (rounding up)

1.7 - Forecast upcoming bills based on history

1.8 - Simulate financial outcomes

1.9 - Notify users when goals are not on track

System (Backend) Functionality -

1.1 - Encrypt user data at rest

1.2 - Store all user budgets and transactions securely

1.3 - Sync transactions daily from linked bank accounts

1.4 - Maintain user session / authorized tokens

1.5 - Log / track user interactions for future improvements

1.6 - Automatically back up user data

1.7 - Track user engagement metrics

1.8 - Allows admins to manage / moderate accounts

1.9 - Generate monthly usage reports

1.10 - Enable continuous integration for future features

Admin Staff Functionality -

1.1 - Access / manage flagged transactions

1.2 - View support tickets

1.3 - Respond to user questions / AI errors

1.4 - View / manage platform usage

Notification System Functionality -

1.1 - Send daily summaries of spending activity

1.2 - Send weekly/monthly budget recaps

1.3 - Alert user of any missed saving goals

1.4 - Notify users about upcoming subscriptions charges

1.5 - Remind users to update their budgets every month

List of Non-Functional Requirements

Security -

- All data must be encrypted using HTTPS protocols

- All sensitive data (user credentials) must be encrypted at rest (AES-256)

- User auth must be protected via secured password (Firebase / Supabase auth)

- Common web vulnerability prevention (SQL injection)

- Role-based access control for restriction (admin / user privileges)

Performance -

- The app must laid the user dashboard within 3 seconds on stable connection

- AI recommendations should be generated / displayed within 5 second of user input

- All backend APIs must respond within 200ms

- The app must support real-time syncing of transactions

Usability -

- The UI must be clear and intuitive, allowing new users to complete key tasks with no documentation (budget setup / expense tracking)

- The system should support (dark / light) modes

- The AI assistant should use natural language for non-technical users

Reliability -

- The system should maintain ~ 99.5% uptime over a month period

- Transaction data should be backed up at least every 12 hours

- The system should automatically retry failed background syncs

Maintainability -

- Code must follow industry standard JS linting rules (ESLint + Prettier)

- Docker must be used to ensure local / production environments match

- Codebase must be modular which allows changes in one module without affecting others

- All critical modules must have at least ~ 70% test coverage

Portability -

- The app must run identical (cross browser)

- Docker containers must allow deployment across local and cloud environments

Scalability -

- The system should handle at least 500 concurrent users without performance degradation

- The architecture should support horizontal scaling via Docker / AWS

- Database must be optimized for read-heavy workloads

Compatibility -

- The system should integrate seamlessly with popular banks
- The AI engine must be pluggable and upgradable without affecting core functionality

Compliance -

- The app must comply with relevant data privacy laws (GDPR)
- The app must provide users with options to export / delete their data

Supportability -

- System logs must be accessible for debugging and must track user errors with AI assistance
- Admins must have a dashboard for managing / reviewing flagged content

Efficiency -

- System should utilize 60% CPU usage on average under normal load
- Memory usage should not exceed 75% memory allocation
- Efficient logging must be implemented to avoid log flooding

Competitive Analysis

Detailed Feature Comparison

Product	Platform	Income Tracking	Bill Alerts	AI Insights	Subscription Auto-Detect	Mo Ap
Rocket Money	Web, iOS, Android	✓	✓	Limited	✓	✓
EveryDollar	Web, iOS, Android	✓	✓	None	✗	✓
YNAB	Web, iOS, Android	✓	✓	None	✗	✓
Mint	Web, iOS, Android	✓	✓	Basic Insights	✗	✓
PocketGuard	Web, iOS, Android	✓	✓	Basic Tips	✓	✓

High-Level Comparison (Key Features)

Feature	Rocket Money	EveryDollar	YNAB	Mint	PocketGuard	Buc e
AI-Generated Suggestions	No	No	No	No	No	✓ (ML
Automated Subscription Tracking	✓	✗	✗	✗	✓	✓ (En)
Bank Integration	✓	✓	✓	✓	✓	✓ (pla
Customizable Alerts	Limited	Limited	Limited	Basic	Basic	✓ Ad

Exportable Reports	✓	✓	✓	✗	✗	✓ (

Our app offers personalized budgeting advice, superior subscription auto-detection, advanced alert customization, and dual-format report exports—features unavailable or limited in existing products.

Checklist

- The team needs to find timeslots outside of class time to meet
  - Done - Met on 06/07/25
- GitHub Master has been chosen.
  - Done - The GitHub master is Dani
- The team has collectively decided on and agreed to use the listed software tools and deployment server.
  - Done
- The team is ready to use the chosen front-end and back-end frameworks, and those who need to learn are actively working on it.
  - Done - Team is mostly familiar with things we are using, currently learning/reviewing React and Sequel
- The Team Lead has ensured that all members have read and understand the final M1 before submission.
  - On-track
  - Will be done as tech doc is turned in and explained to everyone
- GitHub is organized as discussed in class (e.g., master branch, development branch, folder for milestone documents, etc.).
  - ~~Issue - Need to speak with professor regarding how to submit work and how credit is looked at.~~
  - ~~On-track - Met with professor, have discussed how it works, currently working on cohesion~~
  - Done - Have determined how to pull and create branches for the project, as well as settled on using Gitbook

High-Level System Architecture and Technologies Used

- CIoud Server Host: AWS EC2 – t2.micro (1 vCPU, 1 GB RAM)
- Operating System: Ubuntu 22.04 LTS
- Database: MySQL 8.0.42
- Web Server Framework: Express.js 4.19.2
- Backend Language: JavaScript
  - Backend Framework: Node.js
- Frontend Language: HTML, CSS, JavaScript
  - Frontend Framework: React
- Containerization: Docker
- SSL Certificate: Let’s Encrypt with Certbot
- Version Control & Collaboration: Git, GitHub
- Optional Tools:
  - Firebase
  - AI-Based Budgeting Assistant: Powered by ChatGPT API or Gemini API for personalized financial suggestions

List of Team Contributions

Name	Contributions (for C2&3)	Score (1-10)
Emily Perez	Wrote executive summary, wrote high level system..., set up cloud server, helped with credentials md, helped with main use cases	10
Ishaank Zalpuri	Wrote list of main data...,wrote main use cases, wrote checklist, helped set up permissions with cloud server, did what was assigned	10
Andrew Brockenborough	wrote competitive analysis,	7.5
Dani Luna	made the title page, wrote main use cases, set up aboutus page, did what was assigned	10
Jonathan Gonzalez	wrote main use cases, wrote functional req, wrote non functional req, did what was assigned	10
Gene Orias	wrote table of contents, wrote main use cases, helped with credentials md, did what was assigned	10