Coding Project Final Report
Money-Man

A Money Management Application to Track Expenses and Set Budgets

Group 1
Diego Bravo, Nick Filipov, Jose Bolanos, Alejandro Bravo
CS 440
University of Illinois at Chicago

December 2024

Table of Contents

List of Figures

List of Tables

I Project Description

1 Project Overview

Money-Man is an application that allows you to obtain access to a personal accountant in virtual form. Money-Man will be able to track real-time spending from bank accounts or any 3rd party payment methods. Additionally, it has features such as creating budgets, information on spending habits, notify for payment deadlines, and coupons/discounts.

2 Project Domain

This project focuses on personal finances and it falls within financial technology. The point is to optimize financial resources. The goal is to give users the ability to control their finances by giving tracking tools in order to be able to do this. For the prototype, users are able to manually enter what they are spending their money on. In regards to testing, test cases will focus on accuracy, security, and usability. The application is meant to be a smooth experience for users and very straightforward.

3 Relationship to Other Documents

"Money-Man" Development Project Final Report by Group 32 Shambhavi D., Mashel A., and Deep D. Spring 2022.

"Money Man" Development Project Final Presentation by Group 32 Shambhavi D., Mashel A., and Deep D. Spring 2022.

4 Naming Conventions and Definitions

4a Definitions of Key Terms

Accountant: analyzes financial accounts.

Budget: Estimate of spending funds available for the user,

Spending limit: a limit on certain items in order to minimize costs.

Real numbers: Values that are integers, decimals, or irrational numbers.

4b UML and Other Notation Used in This Document

UML Diagrams: Diagrams that help visualize designs, code architecture, and software systems.

Class Diagrams: UML diagram that helps represent relationships between classes/objects.

-> in class diagrams: shows where a page will take you next when a button is clicked.

4c Data Dictionary for Any Included Models

- Amount user can enter can only be real numbers, decimals and integers.
- Pie charts on the budgeting page will have $ values in real number form however, for simplicity only integers/decimals are used.
- Scale for each piece on pie charts is 0-100 where all parts end up adding to 100% and $ values will go as high as possible.
- Pie charts will be split between different categories depending on type of spending.

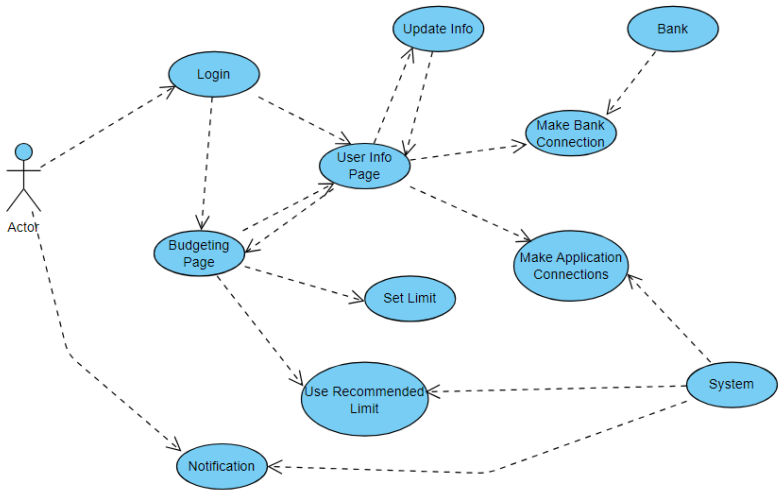
II Project Deliverables

Over the course of the semester we produced the money-man application with all of the money management aspects of the application that were described such as spending categories, spending limits, notifications, and more all using a real time database. We also added some of our own features such as the activity log and dark mode for the application.

1 First Release

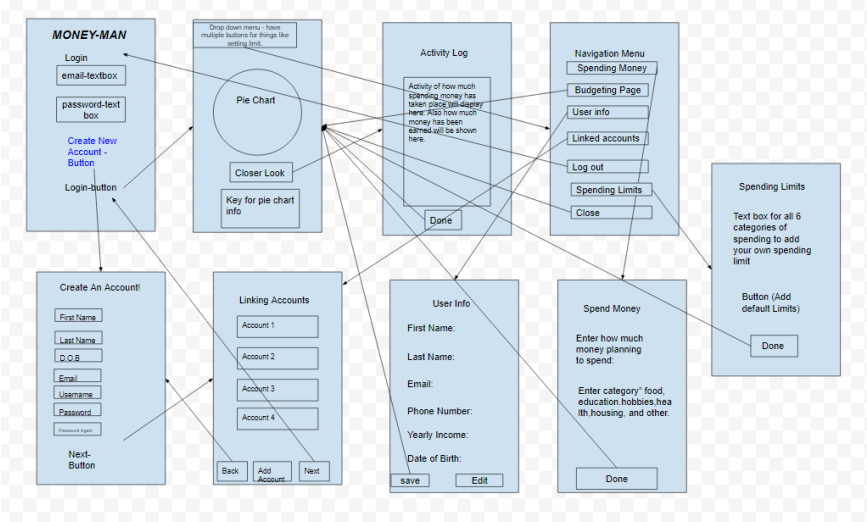
The first release was due October 11th. By that time we have created most of the screens such as the create account page, budget page, login page, categories page, and notifications page. We implemented a file to be able to hold all the information from the users login as well as their total money spending (later switched to database). We

planned out what the final project would look like and began working towards it.



2 Second Release

The second release was due on November 9th. By this point we created a database which will have sections for every single person automatically when they create an account. Since the database was running we were able to finish the add categories, create accounts, and spending pages by making all information be saved into the database. The budget page was almost completed with its pie chart being filled in with live data from the database and other actions to bring people to other screens. The notification page and activity page were beginning to be created and also added to the database. Most of the pages functionality has been completed, just some finishing touches to them and their UI were needed and would be completed for the final product.



3 Comparison with Original Project Design Document

Differences:

- As stated on page 7 of the money-man report in the “Project Overview” they wanted to use people's credit card and bank information to be able to run the application without any user input. But since that would be a security issue and not possible to do at the current time, we implemented our own page to let users enter the category of spending, the amount, and a description of what they spent money on to track their total spending and then saved it into a database.
- We also added a page to let the users make their own categories and spending limits to let the user decide what they want to categorize stuff as on their own because we don’t have access to their credit card information. Which means there is also no contactless payment.
- Our application doesn’t get coupon information to give to the user or have an AI assistant that the document stated.
- Finally we built our application to run on android phones while the original project design document had android and ios.

Similarities:

- We implemented spending limits to let users track how much money they are spending vs how much they want to spend in total.
- We added the total usage of the user which is displayed on the budget page and also showed separate purchases made on the activity log screen.
- A database was used as stated in the original project design document to hold all of the user information.
- A Notifications page was made and also saved into the database to notify the user when they are at their spending limit for a specific category that they created.

III Testing

1 Items to be Tested

Practically every item in the program was to be tested by all of the members of this team. We each had a synchronized copy of the application pulled from a collaborative repository. Items included, but were not limited to:

- Login
- Account Creation
- User Information
- Notifications
- Budgeting page
- Category addition/removal
- Spending entries
- Activity log
- Navigation

2 Test Specifications

#0 - Account Creation and Login

Description: The user should be able to create an account successfully and login with that same account that was stored to the database.

Items covered by this test: Account Creation and Login

Requirements addressed by this test: N/A

Environmental needs: Android Emulator or Phone

Intercase Dependencies: N/A

Test Procedures: Create an account by clicking "Create Account" on login screen, fill out all required fields on Create Account page, attempt to login to the app with newly created account info (username and password).

Input Specification: Any alphanumeric username and password.

Output Specifications: User should be able to login to the account successfully and see the home screen of the application. The account should be added to the database.

Pass/Fail Criteria:

Pass

- User sees budgeting page after logging in with new account.
- Account has been added to the database.

Fail

- User cannot login.
- User cannot create an account.
- The account was not added to the database.

#1 - User Info Screen

Description: The user should be able to view and edit their account details on the user info screen.

Items covered by this test: User Info and Navigation

Requirements addressed by this test: N/A

Environmental needs: Android Emulator or Phone

Intercase Dependencies: #0

Test Procedures: Open navigation drawer by clicking on the top-left menu button, click on the "User Info" option in the navigation menu. Once the user is on the user info page, the user should be able to see their account information. The tester must attempt to edit values in each field of user information.

Input Specification: Any alphanumeric piece of information or empty information.

Output Specifications: The user information should always be displayed in each field on the user info page. Toast messages from android should give feedback on saving new user information.

Pass/Fail Criteria:

Pass

- User can see all user information on the user info page.
- Text fields can be edited and saved successfully.

Fail

- User cannot see all user information on the user info page.
- Text fields cannot be edited.
- New user information does not save.

#2 - Budgeting Test

Description: The user should be able to see the pie chart that visualizes spending activity or see spending entries in the activity log.

Items covered by this test: Budgeting page, Activity log, Spending entries, Category additions/removals.

Requirements addressed by this test: N/A

Environmental needs: Android Emulator or Phone

Intercase Dependencies: #0

Test Procedures: Tester must add category entries by first pressing the "Add Categories" button on the budgeting page or on the navigation drawer menu. In the categories page, tester must fill the text fields with the category name, description, and budget limit (float). Tester now must go back to the budgeting page and press "Add Spending" to add entries to the new category. After adding entries, tester can go back to the budgeting page to check if the pie chart contains the total of entries in that category. Next, the tester must check if the entries were recorded in the activity log by pressing "Closer Look". Next, the tester should

remove the category in “Add Categories” to ensure that the pie chart is empty when all categories are removed. Tester can add more categories and spending entries for thorough testing.

Input Specification: Any alphanumeric name and description. Any float for the spending entries and limits.

Output Specifications: Pie chart should be visible or invisible depending on the number of categories added. Activity log should have each spending entry recorded and displayed for the user. Toast messages will give feedback to the user when entries and categories are saved.

Pass/Fail Criteria:

Pass

- User sees pie chart visualization of spending data.
- Pie chart visualization is dynamic with added entries and category additions/removals.
- Activity log displays each spending entry.
- Spending entries and category changes are saved successfully.

Fail

- User cannot see the pie chart or pie chart visualization is displaying incorrectly.
- Activity log does not display spending entries.
- Spending entries and categories do not save successfully or save incorrectly.

#3 - Notifications

Description: Notifications send to the user’s device when a limit has been reached or exceeded.

Items covered by this test: Spending entries, Notifications, Activity log.

Requirements addressed by this test: N/A

Environmental needs: Android Emulator or Phone

Intercase Dependencies: #0 and #2

Test Procedures: Tester must add spending entries that will exceed or meet the budget limit of a category. If the notification is received, the tester must press the notification and ensure that it redirects the user to the activity log. Tester should go to the notifications page to ensure that the budget alert displays there. If the

alert is displayed, the tester must press the alert to ensure that it redirects the user to the activity log. Tester should also ensure that the "Clear All" button on the notifications page works correctly.

Input Specification: Any float for the spending entries.

Output Specifications: Notification pop-up on the device's system. Notification/alert item in the notifications page. Push notification or alert item should redirect to the activity log. "Clear All" button removes all notifications/alerts on the notification page.

Pass/Fail Criteria:

Pass

- Notifications pop-up on the device's system.
- Notifications display on the notification page.
- Notifications redirect user to activity log on press.
- "Clear All" button removes all notifications on the notification page.

Fail

- No notifications are sent to the user's device.
- Notifications do not display on the notification page.
- Notifications do not redirect to the activity log.
- "Clear All" button does not remove all notifications on the notification page.

3 Test Results

#0 - Account Creation and Login

Date(s) of Execution: September 23, 2024 - September 27, 2024

Staff conducting tests: Diego Bravo, Alejandro Bravo, Jose Bolanos, Nick Filipov

Expected Results:

- User sees budgeting page after logging in with new account.
- Account has been added to the database.

Actual Results:

- User sees budgeting page after logging in with new account.
- Account has been added to the database.

Test Status: Pass

#1 - User Info Screen

Date(s) of Execution: September 30, 2024 - October 4, 2024

Staff conducting tests: Diego Bravo, Alejandro Bravo, Jose Bolanos, Nick Filipov

Expected Results:

- User can see all user information on the user info page.
- Text fields can be edited and saved successfully.

Actual Results:

- User can see all user information on the user info page.
- Text fields can be edited and saved successfully.

Test Status: Pass

#2 - Budgeting Test

Date(s) of Execution: October 7, 2024 - November 29, 2024

Staff conducting tests: Diego Bravo, Alejandro Bravo, Jose Bolanos, Nick Filipov

Expected Results:

- User sees pie chart visualization of spending data.
- Pie chart visualization is dynamic with added entries and category additions/removals.
- Activity log displays each spending entry.
- Spending entries and category changes are saved successfully.

Actual Results:

- User sees pie chart visualization of spending data.
- Pie chart visualization is dynamic with added entries and category additions/removals.
- Activity log displays each spending entry.
- Spending entries and category changes are saved successfully.

Test Status: Pass

#3 - Notifications

Date(s) of Execution: October 21, 2024 - November 29, 2024

Staff conducting tests: Diego Bravo, Alejandro Bravo, Jose Bolanos, Nick Filipov

Expected Results:

- Notifications pop-up on the device's system.
- Notifications display on the notification page.
- Notifications redirect user to activity log on press.
- "Clear All" button removes all notifications on the notification page.

Actual Results:

- Notifications pop-up on the device's system.
- Notifications display on the notification page.
- Notifications redirect user to activity log on press.
- "Clear All" button removes all notifications on the notification page.

Test Status: Pass

4 Regression Testing

N/A

IV Inspection

1 Items to be Inspected

Every piece of code that was pushed was inspected by each group member to ensure functionality worked for all our devices. Additionally, we wanted to make sure the code worked with multiple accounts that have different attributes/spending habits in order to ensure code was optimal.

2 Inspection Procedures

As a procedure each student would pull the code from the repository inorder to ensure it has newly pushed code and would run on their device using their account. They would test by going on the updated pages to make sure it works on their device and to test any new functionality like adding categories or changing user data. We all made sure any changes made locally were made through the database inorder to ensure that data updated correctly when applicable. There wasn't really a checklist due to limitations, we just made sure it looked appealing and most importantly functionality.

3 Inspection Results

Each inspection was done by each member of the group when code was pushed into the repository inorder to ensure the application worked and didn't cause any extra bugs. The time and date would vary depending on when the student would pull the code inorder to test and to be able to work on the project themselves. The result was giving feedback/communicating with the author of the code inorder to inform of any bugs/errors within the code. Additionally, we would adjust minor issues when applicable. Reinspection occurred when code was adjusted/fixed and was inspected again by all students.

V Recommendations and Conclusions

All code pushed has been covered and passed the inspection because its functionally and feasibility has been maximized as much as within the restrictions we had and time given for this project. In the case that it didn't pass the inspections then further testing would have to be done by each member to figure out what bugs the code has and if it leads to other bugs as well. Furthermore, adjustments would be made when the idea of the project shifts. So, necessary changes will be implemented accordingly.

VI Project Issues

1 Open Issues

Some open ended issues that happened during the money-man coding project was that we weren't able to add some of the features that money-man had due to the time we had to do this project, security risks, and permissions we would need for this. So, being able to add credit cards/banks we had to remove this idea and implement something else which just manually enters the amount a user will spend and add additional features that were not inside the money-man inorder to make the app as close as possible. Other issues occurred when trying to save user information we tried to save locally but that could cause safety concerns and it wouldn't work realtime. So, we used a realtime database in order to fix this problem.

2 Waiting Room

Assuming we are given more time and resources to actually make the application then we would like to extend the accessibility to this application as it currently is coded in Kotlin using android studio it will have to be coded in swift in order to be able to support this application. Additionally, we would also like to add more features like being able to connect to bank accounts/credit cards/or some form of way to track expenses made so users can be aware of what purchases they're making. Other features can include multiple types of graphs, being able to restrict the amount used daily/monthly/yearly, and giving

suggestions on how to save money/budget expenses. Furthermore, we would also improve the UI in order to make the experience as best as possible.

3 Ideas for Solutions

Some solutions can be getting more time to do this project because we had limited time to get the project the best we could. Another solution can be being able to link payment methods by using either a third party to help us connect to banks or collaborating with the banks themselves in order to use them. Additionally, we would like to be able to maximize the amount of usage so we will be able to release the app in desktop form for any computer users and to apple users which will lead to this app having to support swift. Furthermore, lawyers would have to be involved in order to make some of these features happen.

4 Project Retrospective

What worked well:

- Every group member worked on the project extensively.
- Active communication was applied whether through Jira/virtually or physically in-person.
- Everyone was on the same task when working on certain pages of the application.
- All deadlines were met regarding coding demos or scenarios.
- Adapted to the new IDE(android studio) and Kotlin.

What didn't work well:

- Figuring out how to downsize the application down because there were many ways to do it.
- Meeting at specific times due to schedule/personal conflicts.
- Figuring out what language/IDE to use took too much time.
- Figuring out how to replace core features.

VII Glossary

Accountant - A professional or system that analyzes financial accounts and transactions.

Budget - An estimate of spending funds available for a user within a defined period.

Spending Limit - A restriction placed on spending within specific categories to minimize costs.

Activity Log - A feature that displays a detailed view of how a user is spending their money over time.

Categories - Custom labels created by users to organize and track spending within specific types of expenses.

Notifications - Alerts sent to users about payment deadlines or when spending limits are approached.

Pie Chart - A circular chart divided into slices to illustrate proportions; in this app, it represents spending categories.

Database - A system for storing user data, including spending categories, limits, and activity logs, in real-time.

Android - A mobile operating system where the app runs.

Intercase Dependencies - The relationship or reliance between different test cases, where the outcome of one test may impact another.

Toast Messages - Brief, non-intrusive messages that provide feedback to the user, typically displayed at the bottom of the screen in Android applications.

Navigation Drawer - A UI component in Android applications, usually accessed via a menu button, that slides out from the side of the screen to display navigation options.

Regression Testing - A type of testing performed to confirm that recent code changes have not negatively impacted the existing functionality of an application.

Jira - A project management tool widely used for issue tracking and collaboration in software development.

IDE (Integrated Development Environment) - A software application that provides tools for software development, such as code editing, debugging, and compiling features.

Third-Party Integration - The process of incorporating external services or tools (e.g., APIs) into an application to extend its functionality, such as linking to payment systems or banks.

Budget Alert - A notification sent to a user when their spending exceeds or meets a predefined limit in a particular budget category.

VIII References / Bibliography

[1] Money-Man Group 32 Shambhavi D., Mashel A., and Deep D. Spring 2022.

[2] Robertson and Robertson, Mastering the Requirements Process.

[3] A. Silberschatz, P. B. Galvin and G. Gagne, Operating System Concepts, Ninth ed., Wiley, 2013.

[4] J. Bell, "Underwater Archaeological Survey Report Template: A Sample Document for Generating Consistent Professional Reports," Underwater Archaeological Society of Chicago, Chicago, 2012.

[5] M. Fowler, UML Distilled, Third Edition, Boston: Pearson Education, 2004.

IX Index