**CS673 Software Engineering** 

**Team 1 - PennyWise**

**Project Proposal and Planning**

| Team Member | Role(s) | Signature | Date |
| --- | --- | --- | --- |
| Brain Fenstermacher | Team leader | *Brian Fenstermacher* |  |
| Jisoo Lee | Requirement leader | *Jisoo Lee* | 09/08/23 |
| Sherif Zeyada | Design and Implementation leader | *Sherif Zeyada* | 09/08/23 |
| Mali Rivera | QA leader | *Mali Rivera* | 09/08/23 |
| Clyde Yeung | Configuration leader | *Clyde Yeung* | 09/08/23 |
| Sean Rawson | Security leader | *Sean Rawson* | 09/08/23 |
| Chaitanya Saraogi | Requirement Leader 2 | *Chaitanya* | 9/10/23 |
|  |  |  |  |
|  |  |  |  |

**Revision history**

| **Version** | **Author** | **Date** | **Change** |
| --- | --- | --- | --- |
|  | **Chaitanya Saraogi** | **10/16/2023** | **Updated requirements and added cumulative work flow from Jira with Kan Report for completed Kans** |
|  | **Jisoo Lee** | **10/16/2023** | **Updated scores in Jira, labeled by iterations for user stories. Use case diagram. Requirement updates.** |
|  |  |  |  |
|  |  |  |  |

[Overview](#_heading=h.gjdgxs)

[Related Work](#_heading=h.30j0zll)

[Proposed High level Requirements](#_heading=h.1fob9te)

[Management Plan](#_heading=h.3znysh7)

[Objectives and Priorities](#_heading=h.2et92p0)

[Risk Management (need to be updated constantly)](#_heading=h.tyjcwt)

[Timeline (need to be updated at the end of each iteration)](#_heading=h.3dy6vkm)

[Configuration Management Plan](#_heading=h.1t3h5sf)

[Tools](#_heading=h.4d34og8)

[Deployment Plan if applicable](#_heading=h.2s8eyo1)

[Quality Assurance Plan](#_heading=h.17dp8vu)

[Metrics](#_heading=h.3rdcrjn)

[Code Review Process](#_heading=h.26in1rg)

[Testing](#_heading=h.lnxbz9)

[Defect Management](#_heading=h.35nkun2)

[References](#_heading=h.1ksv4uv)

[Glossary](#_heading=h.44sinio)

# Overview (Brian)

With constant rising inflation and costs of living, budgeting can be one of the more difficult areas for college students and young professionals to excel in. Traditionally, a lot of budgeting is done through spreadsheets and that can be a very tedious and time consuming task. More recently, there have been a number of budgeting applications that target this age range of individuals but they tend to either have a steep cost with lots of integrations or they’re free and extremely limited with a tall pay wall to access the other features that many users would like to have. Our project aims to bridge that gap by providing users with the features that they want, and no pay wall stopping them from having better finances. Our product will feature:

1. Add monthly income and expenses
   1. Initial input that prompts users for net monthly income
      1. \*This information will be stored in a database so users will not have to re-enter the information each time they sign in
   2. Next, the user is prompted for their monthly expenses
      1. \*This information will also be stored in the database\*
      2. This will be related to item #3 below where expense items can be arranged into categories
2. Login and access control (security for users)
   1. When the user first arrives to the application, they are prompted to either log in or sign up
      1. Log in requests the users username and password
      2. Sign up requests the desired username, the users first and last name and a password
         1. \*This information will also be stored in the database\*
         2. We’ll need to have a method that checks if the desired username is already associated with another account before the new account can be created
3. Budgeting Categories
   1. We’ll have some default/predefined categories such as concrete expenses, variable expenses, recreational expenses, and disposable expenses
   2. We can also give users the ability to add in new categories like savings or investments but this will be lower priority
4. Tracking Expenses
   1. More of a backend feature - as more expenses are added in, there will be a running total of how much of the net income is left over, how much each category has totaled and how much the running total of all expenses across all of the categories is
   2. UI - bar tracking
5. Additional feature -> tips from experts
   1. *Low Priority*
   2. We can have this somewhat built into the tracking expenses, maybe we have it built into the backend as well so whenever a certain expense exceeds the ‘expert recommendation’ you receive a little warning or reminder that you should look into reducing that area of expense

We intend to build this out using the Java tech stack.

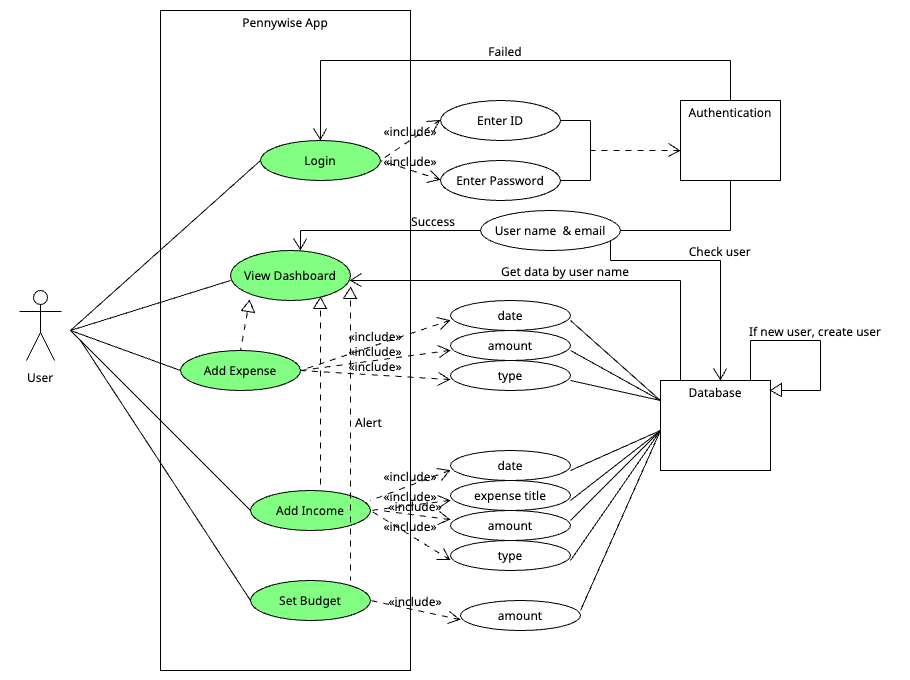
# Related Work (Brian)

(Please describe any similar software systems that you have found through the online research, and the differences between your software and those software systems.)

* Everydollar
  + The features here are very similar to our project. From the reviews, it sounds like there is a big difference between the free version and the paid version and I think that our product would bridge that gap a good bit. The free version offers a customizable budget and savings funds and the premium version offers bank connectivity, budget insights, paycheck planning, smart tracking suggestions, financial roadmap, goal setting, group coaching, csv download. Our application would offer a few of the premium version features to make it more competitive plus some original items.
  + From a design standpoint, everything is easy to use and the interface is very intuitive. I think this could be a great product for us to base some design elements on.
* Goodbudget
  + Very similar to the above but lets you categorize your ‘envelopes’ for spending. Everything must be manually entered (ie. your monthly income and what you spend) but I think given our timeframe for this project that will be a good approach for us as well.
  + From a design perspective here, I think this is a bit more confusing. The user gets dropped right into filling out information but it’s vague if you’re creating categories or inputting expenses. I could foresee myself being able to understand this but it definitely isn’t as clear as EveryDollar at first glance.
* PocketGuard
  + Easy to navigate dashboard, show spendable money after bills, goals and necessities are accounted for. Pie chart shows each category. Easily see how your student loans fit in with the rest of your income and spending.
  + Additionally feature that finds recurring monthly expenses and notifies you of them (this would require bank integration but is definitely a good way to save money if these monthly expenses are unknown to the user)
* Mint
  + One of the best rated budgeting apps. Securely connects to your bank account and automatically inputs purchases. Notification features (upcoming bills, low funds, avoid overspending etc).
  + This app seems very integration heavy, all of your accounts are added to your mint profile and can be tracked here which provides the user with real time updates on how payments are coming along.

# Proposed High level Requirements

Use Case Diagram:



* 1. Functional Requirements (Jisoo/Sherif)  
     Essential Features:

1. Monthly Income Tracking

1- As a user, I want to input my net monthly income, so that I can

understand and plan my budget. (Brian)

2- As a user, I want to ensure that my income is saved to my account/profile so that I do not have to re-input that information each time I use the program. (Brian)

Hours: 30 - 35 hours per person

1. Monthly Expense Tracking

3- As a user, I want to record monthly expenses, so that I can control

Expenditure. (Chaitanya)

4- As a user, I want to add monthly expenses and store them into

predefined categories into a database, so that I can manage

spedings effectively. (Chaitanya)

Hours: 30 - 35 hours per person

1. User Registration and Authentication

5- As a new user, I want to register with a unique username and

password so that I can securely access my financial information. (Jisoo)

6- As a user, I want to log in with my username and password to

access my personal budgeting dashboard (Jisoo)

1. Custom Expense Chart

7-As a user, I want to generate a chart illustrating my spending

distribution across categories (Mali)

* + 1. Desirable Features:

1. Expense Categorization and Tracking:

1-As a user, I want to categorize expenses into predefined categories

like concrete expense, variable expense, recreational expense, and

disposable expense, so that I can categorize my expenses easily. (Mali)

2- As a user, I want the system to track expenses by each category,

so that I can monitor the details of my expenses. (Mali)

Hours: 15 - 20 hours per person

1. Expense Tracking Visualization:

3- As a user, I want to view graphical charts of my expenses to

understand the spending trends. (Sean)

4- As a user, I want the chart to be interactive, so that I can click on

each category. (Sean)

Hours: 15 - 20 hours per person

1. Expense Alerts and Notifications

5- As a user, I want to receive notifications when predefined spending

limits are getting close to being reached. (Sherif)

6- As a user, I want these alerts to be based on both user-defined

Limits and expert recommendations (Sherif)

Hours: 15 - 20 hours per person

* + 1. Optional Features:

1. Budget Setting

1- As a user, I want to set spending limits for predefined custom

categories, so that I can manage my budget effectively. (Brian)

2- As a user, I want to receive alerts when I am near to or exceed my

budget, so that I can avoid unnecessary spendings. (Brian)

Hours: 20 - 25 hours per person

1. Financial Tips

3- As a user, I want to get notified of financial tips and suggestions

when my spending exceeds the limit so that I could make wise

decisions. (Chaitanya)

Hours: 20 - 25 hours per person

1. Financial Impact Assessment

4- As a user, I want to input estimated expenses for significant

purchases to understand their impact on my monthly budget. (Jisoo)

Hours: 10 - 15 hours per person

1. Expense Review and Recommendations

5- As a user, I want to review my recreational expenses quarterly and

adjust or drop categories based on my usage. (Clyde)

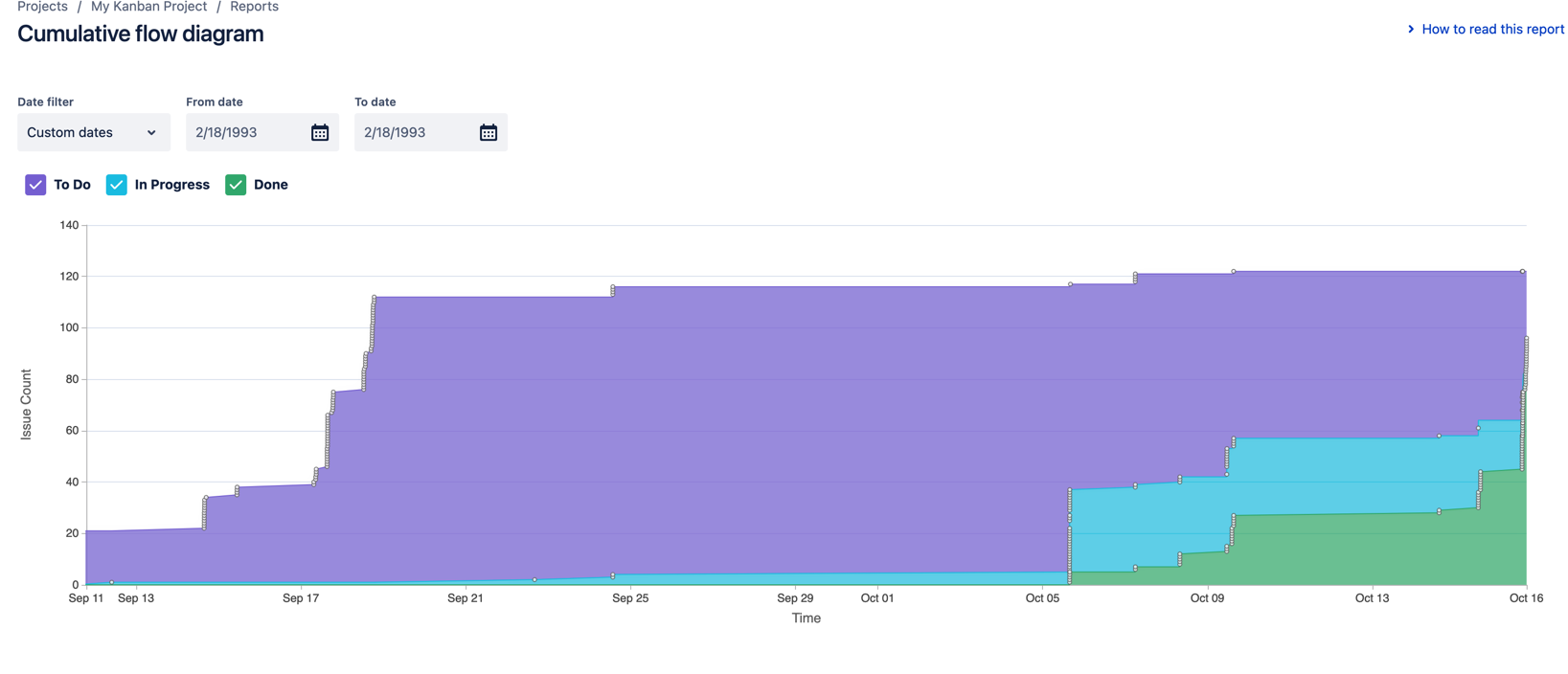
6- As a user, I want to receive recommendations on optimizing

expenses based on expert guidelines. (Clyde)

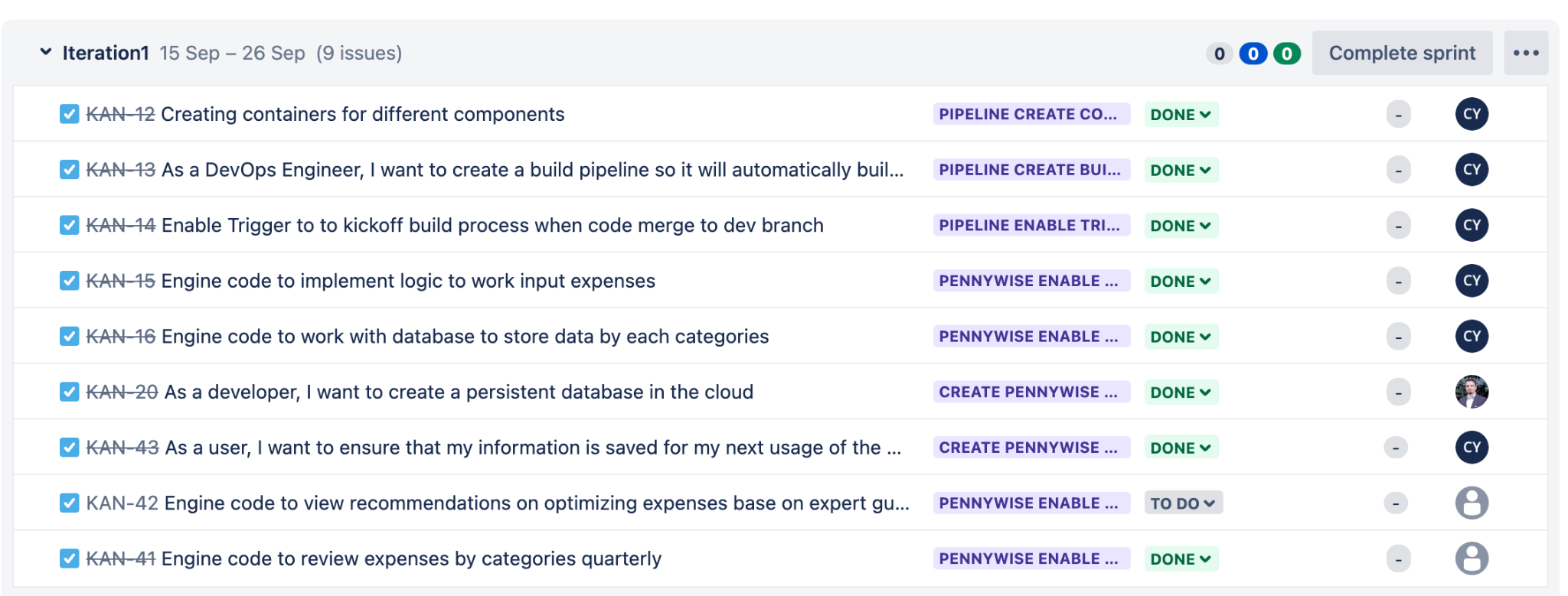
Hours: 15 - 20 hours per person

* 1. Nonfunctional Requirements (Sean)
     1. Security requirements
        1. Users should not be able to access other users’ data
        2. Users will authenticate using Google OAuth
        3. Role-based authorization for certain paths, i.e. role\_admin, role\_user
        4. Authentication and authorization will be handled by a separate service from front-end and back-end systems.
     2. Non-essential/nice-to-have
        1. Email confirmation for sign-up
        2. Once authenticated, a JSON web token can be used for subsequent requests. Token expiration of 1 hour (or less?)

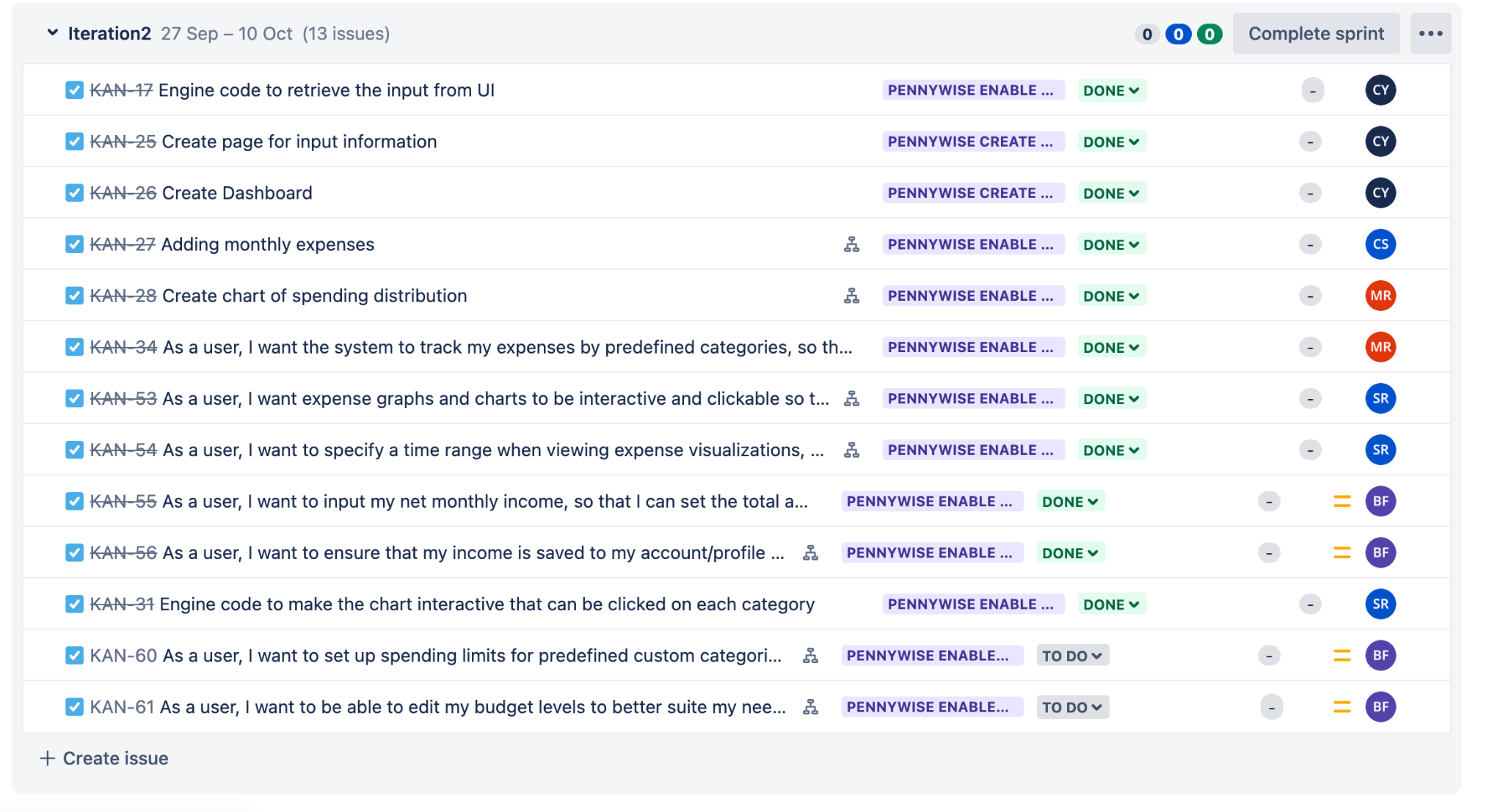
**Cumulative Flow Diagram from Jira**

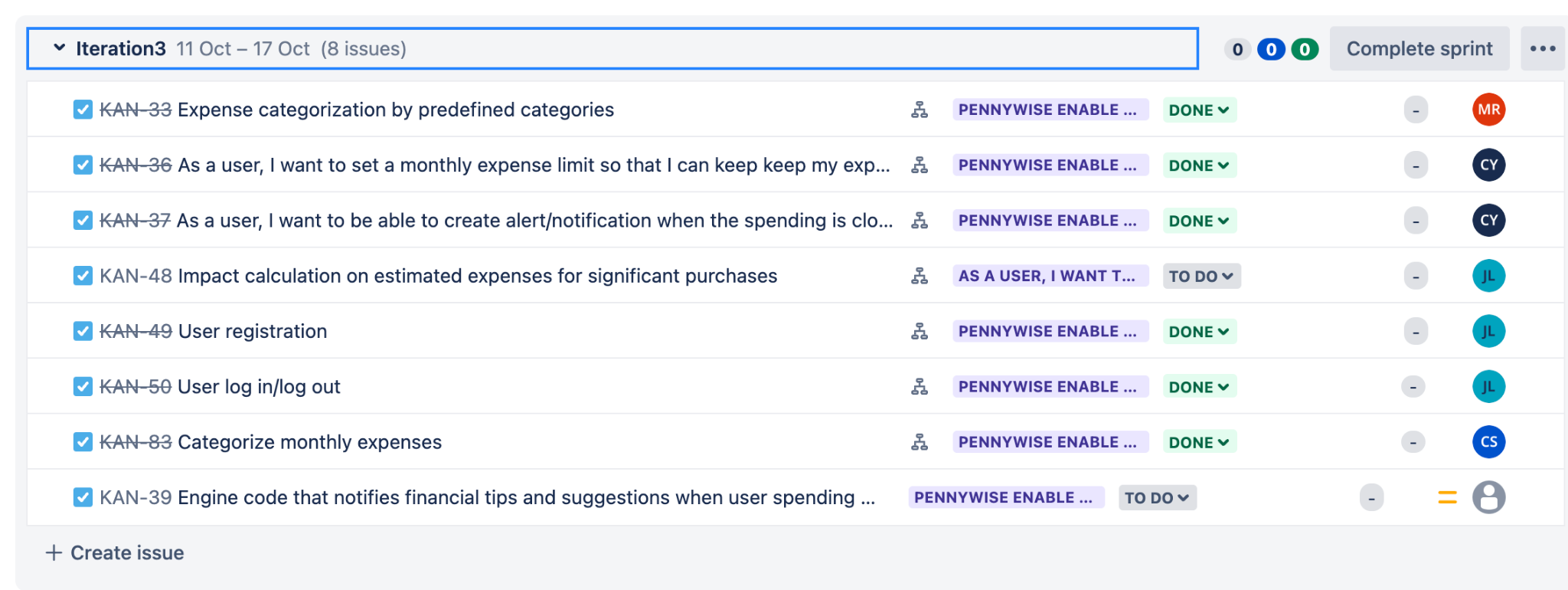
****

**Kan Report by Sprint (iterations)**

****

9 Stories were completed in iteration1 without any issues.

****

13 stories were completed in iteration2 and 2 stories were optional requirements that were moved to iteration3.****

8 stories were completed in iteration3. 3 stories are optional requirements that are flagged and not completed for this iteration.

**On-Screen Appearance Requirements:**

| **Identifier** | Category | **Requirement** |
| --- | --- | --- |
| REQ1 | Consistent Branding | The application should maintain a consistent and recognizable brand identity, including logos, color schemes, and typography, throughout the UI. |
| REQ2 | Intuitive Layout | The layout of the user interface should be intuitive and well-organized, with logical placement of elements such as buttons, forms, and menus. |
| REQ3 | Responsive Design | The application should be responsive to different screen sizes and orientations, ensuring a consistent and visually pleasing experience on desktop, tablet, and mobile devices. |
| REQ4 | Accessibility | The UI should adhere to accessibility standards (e.g., WCAG) to ensure that it is usable by individuals with disabilities. This includes providing text alternatives for images, keyboard navigation support, and sufficient color contrast. |
| REQ5 | Visual Hierarchy | Important information and actions should be visually emphasized using techniques such as font size, color, and spacing to guide the user's attention. |
| REQ6 | User-Friendly Icons | Icons used within the application should be clear, easily recognizable, and consistent with the overall design. |
| REQ7 | Error Handling | Clear and user-friendly error messages should be displayed in the event of user errors or system issues, helping users understand and resolve problems. |
| REQ8 | User Feedback | The application should provide feedback to users when actions are performed, such as successful form submissions or updates to data. |
| REQ9 | Interactive Elements | Buttons, links, and other interactive elements should have visual cues (e.g., hover effects, color changes) to indicate their interactivity. |
| REQ10 | Loading Indicators | When the application is processing data or loading content, loading indicators (e.g., spinners or progress bars) should be displayed to inform users of the ongoing process. |
| REQ11 | Clear Typography | Use legible and consistent typography for text elements, with attention to line spacing and font choice for readability. |
| REQ12 | Image Quality | Ensure that images used in the UI are of high quality and appropriately compressed to minimize load times. |
| REQ13 | Testing Across Browsers | Verify that the UI renders correctly and consistently across various web browsers and platforms. |
| REQ14 | User Preferences | Provide options for users to customize certain aspects of the UI, such as font size or theme (light/dark mode), to enhance the user experience. |
| REQ15 | Guidelines Compliance | Adhere to design guidelines and best practices for the chosen platform |

**FURPS Table:**

| **Category** | **Description** | **Priority** |
| --- | --- | --- |
| **Functionality** | • Input monthly income and expenses.  • Expense categorization and tracking  • Budget setting and tracking.  • Expense review and recommendations.  • Custom expense chart.  • Financial impact assessment.  • Expense alerts and notifications. | High |
| **Usability** | • User-friendly interface.  • Intuitive navigation.  • Clear feedback and notifications.  • Cross-platform compatibility.  • Comprehensive documentation. | High |
| **Reliability** | • Data accuracy and consistency.  • Performance monitoring and optimization.  • Data backup and recovery. | High |
| **Performance** | • Efficient response to user actions.  • Scalability for future enhancements. | High |
| **Supportability** | • Database management for data storage.  • Compatibility with web browsers and mobile devices.  • Data privacy and security.  • Continuous performance monitoring. | High |

# Management Plan (Jisoo/Sean)

## Objectives and Priorities (Jisoo)

From the user survey, we could find meaningful outcomes like the usage of budgeting applications, frequency of use, pros and cons of existing applications, and desired features. Below table shows the priorities listed based on user survey feedback and their preferences. 45.5% of the users have not used budgeting applications before, 54.5% of the users have used budgeting applications like Mint, YNAB, and integration with other banking applications.

Objectives and priorities

1. User centric design

Our primary objective is to create a user friendly budgeting application by

simplifying the budgeting tasks.

1. Expense tracking and budgeting categorization

Users will be able to input and categorize monthly expenses efficiently.

1. Expense analysis

This application will offer visual representation of the user's financial data through pie charts and bar graphs.

1. Data storage efficiency

We will store the user's monthly income and expenses data securely in a database.

1. User friendly registration and login

We prioritize developing a secure registration and login process. Users can easily create accounts and access the application.

1. Expert tips integration

Although a lower priority, we will try to integrate expert financial tips into our application. This feature might include reminders and notifications when certain expenses exceed the goal threshold.

Table.1 User preferences from the user survey result

| Priority | Objective | User Survey Response rate (%) |
| --- | --- | --- |
| 1 | Ease of Use | 90.9% |
| 2 | Cost of App | 81.8% |
| 3 | Goal Tracking | 72.7% |
| 4 | Trend Analysis | 72.7% |
| 5 | Integration with Other Apps | 72.7% |
| 6 | Personalized Financial Advices | 45.5% |
| 7 | Security | 45.5% |
| 9 | Gamification | 27.3% |

## Risk Management (need to be updated constantly) (Sean)

The current risks that we face as a group in iteration zero are mainly planning and requirements related, although they are less likely as the iteration comes to a close. Many of the essential requirements have been decided via group brainstorming and the user survey developed by Jisoo.

**Risk Management Sheet Link:**

[CS673\_SPPP\_RiskManagement\_team1.xlsx](https://docs.google.com/spreadsheets/d/1jIIdIjbgg8x2sBlz-vVtuNHZJ4onsWNG/edit?usp=drive_link&ouid=114587402770082422631&rtpof=true&sd=true)

## Timeline (this section should be filled in iteration 0 and updated at the end of each later iteration)

| Iteration | Functional Requirements(Essential/Disable/Option) | Tasks (Cross requirements tasks) | Estimated/real person hours |
| --- | --- | --- | --- |
| 1 | Design to provide real-time expense tracking, which will continuously update the user’s remaining budget on income, expenses, and amounts.  Design using Google Oauth2 and JWT to provide user authentication and use Spring security.  Designed UI for dashboard with navbars and made demo video for application UI. | Design and implementation Backend code implementation - create domain, controller, mapper, repository, service, and validation class for User, Expense, and Income class. | 30 - 35 hour |
| 2 | Essential Requirement: - User settings   * Expense (API) * Login (Google Oauth) * Dashboard (API)   - Integration with backend  - Set logic for controllers  - Notification set up | Monthly income set up and tracking  Monthly expense tracking User Registration and Authentication with Google Oauth Dashboard | 35 - 40 hour |
| 3 | Frontend/Stylistic & Additional Features - notification and expense tracking - Google sign-in | Google Oauth sign-in Expense Categorization and Tracking Expense Tracking Visualization Expense Alerts and Notifications  Integration with backend endpoints  Deployment finalization | 45 - 50 hour |

# Configuration Management Plan (Clyde)

## Tools

(In this project, we will use Git and Github as the version control tools. Please also specify any other tools to be used, e.g. IDE tools, CI/CD tools, container tools, SAST or DAST tools, and any other DevOps tools)

* Version Control: GitHub
* CI/CD: GitHub Actions (Automated pipeline)
* Project Management: Jira
* Containerization: Docker
* Deployment: Ansible+Docker Compose or Kubernetes
* Branching Strategy: GitFlow with Pull Requests
* Code Quality: SonarQube for DAST
* Container Orchestration: Kubernetes
* Deployment platform : Heroku
  1. Code Commit Guideline and Git Branching Strategy  
     (Please briefly describe criteria for the code commitment and the branching strategy used, e.g. what are the branches to be used, how the pull request will be used etc. Here is an article to give you some basic knowledge about different git branching strategies: <https://www.flagship.io/git-branching-strategies/>

CodeCommitGuideLine.docx

## Deployment Plan if applicable

DeploymentPlan.docx

# Quality Assurance Plan (Mali)

## Metrics

(Describe the metrics to be used in the project to measure the quality of your software. Each metric should be measurable and quantifiable. Examples of metrics include product complexity (LOC, # of files, # of classes, # methods, cyclomatic complexity, etc.) , defect rate (# of defect per KLOC), # of test cases, test case pass rate, cost (# of person hours used), # of user stories completed, etc. **The result of these metrics should be reported in the progress report/ iteration summary sheet.**)

Software Quality Assurance Metrics will be used to continuously monitor and improve the quality of PennyWise software. Test Coverage will be used to locate any gaps in testing that may need to be addressed. Defect Density will be used to measure the ratio of code defects and identify any problem areas. Execution Rate will be used to assess the effectiveness of the software. Test Case Effectiveness will be used to measure the effectiveness of the written test cases and their ability to detect defects. Defect Escape Rate will be used to assess the quality of the QA process. Test Automation Coverage will be used to assess how well we are leveraging automated tests.

The metrics described below will be used as tools for our team to monitor problem areas and progress throughout the course of development.

| Metric Name | Description |
| --- | --- |
| Test Coverage | The percentage of code or functionalities covered by test cases. |
| Defect Density (KLOC) | The number of defects per thousand lines of code. |
| Execution Rate | The number of test cases executed in a given time frame. |
| Test Case Effectiveness | The number of defects measured by test cases. |
| Test Case Pass Rate | The percentage of test cases passed by a testing cycle. |
| Defect Escape Rate | The number of defects found post-production compared to the number of defects identified by QA |
| Test Automation Coverage | The percentage of tests that have been automated |

* 1. Coding Standard

(Describe any coding standard to be used)

A Coding Standard will be used as a guideline for best practices when coding. The rules of the coding standard will follow Industry-Specific Standards for software development. The intent of the Coding Standard is to improve readability, ensure compatibility, and improve code quality.

A Coding Standard Document is shared [here.](https://docs.google.com/document/d/1lmJguwDIESaRVq2h26xpJSyTfEi8DN7x9BKGyDgtOpQ/edit#heading=h.gjdgxs)

## Code Review Process

(Everyone should review all documents to be submitted. Here you will mainly describe how the code review will be done. Who will review the code, e.g. design or implementation leader will review all code or team members review each other’s code. Do you use pull requests for the code review? Is there a checklist to help review? What feedback should the reviewer provide?)

All team members should review the weekly project iteration code before it is submitted to blackboard. Ideally, team members will link up with another team member each week and walk through each other’s code. If team members are unable to meet for a live code review session, they can review each other’s code in their own time and use the Code Review Summary Worksheet shared [here](https://docs.google.com/document/d/1aoF2bAZW3emQUE39U59GyqVlGon45UBCh5B9XmB8v1g/edit#heading=h.gjdgxs) for guidance.

Developer summary:

1. Briefly describe the intent of the code
2. Briefly explain any problems that you encountered or concerns you may have with the code
3. List any questions you may have for the reviewer

Questions for the Reviewer to consider:

1. Is the code working as expected?
2. Is the code executing in a timely manner?
3. Is the code accomplishing the goal described above?
4. Do you agree with the potential issues described above, if applicable?
5. Do you see any other potential issues?
6. Could the code be more efficient?
7. Could the code be simplified?
8. Are there any functions that could be broken up into simpler functions?
9. Does the code follow the coding standard?
10. Is the code easy to follow and well-documented
11. Does the program align with the goals of the overall project?
12. How can we take the program further?
13. Are design patterns being used and/or used well?

## Testing

(Both manual testing and automated testing should be considered. Both unit testing and integration testing should be considered. Briefly describe the testing tools/framework to be used, the personnel involved (e.g. the QA leader will focus on the integration testing and each developer will unit test their own code), when and what types of testing will be performed, the testing objectives, etc)

Integration and unit tests will be used for this project. The plan is for each developer to unit test their code throughout development and for the QA leader to focus on integration testing. Unit tests will be written with JUnit5 and Mockito and integration tests will be implemented with JUnit5, Mockito, and MockMVC. The QA leader will take a lead on reviewing the QA metrics and provide updates on the testing progress (gaps in testing, etc.).

Testing tools/frameworks:

* JUnit5
* Mockito, MockMVC

## Defect Management

(Describe the tool to be used to manage the defect (e.g github issues). The types of defects to look at. The actions or personnel for defect management. )

The team will use Jira to track defects and manage progress. The Jira Software makes it easy to track issues and bugs throughout the course of development. Since we are already using Jira to track our project requirements, it will be helpful to track defects as well here. The QA leader will monitor resolutions to defects and communicate with the Requirements and Team Leaders when necessary.

The team will also use discord to share bugs in their code. In discord, developers can post screenshots and/or videos of defects to share with the team. This will allow for quick problem solving among developers.

# References

(For more details, please refer to the encounter example in the book or the software version of the documents posted on blackboard. )

Mint (https://mint.intuit.com/)

Jira (https://confluence.atlassian.com/jira)

Maven (<https://maven.apache.org/>)

React (<https://react.dev/>)

EveryDollar (<https://www.ramseysolutions.com/ramseyplus/everydollar?gclid=CjwKCAjwu4WoBhBkEiwAojNdXqhwOBTzam_dRlAEdkc2-YWG48yndJpB3V9CxdnqSDI6OhZxrlBsGBoCZkcQAvD_BwE>)

GoodBudget(<https://goodbudget.com/>)

PocketGuard(<https://pocketguard.com/pricing/>)

Heroku (https://www.heroku.com/)

# Glossary

(Any acronym used in the document should be explained here)