Team 6 (PennyWise)

1. **Emily Hawkins**
   * **Student ID**: 028199975
   * **Email**: emily.hawkins01@student.csulb.edu
2. **Jonny Olswang**
   * **Student ID**: 030865326
   * **Email**: jonny.olswang01@student.csulb.edu
3. **Kenneth Nguyen**
   * **Student ID**: 027559894
   * **Email**: kenneth.nguyen02@student.csulb.edu
4. **Hong Liang Zhao**
   * **Student ID**: 029524142
   * **Email**: HongLiang.Zhao01@student.csulb.edu
5. **Steven Pham**
   * **Student ID**: 027769779
   * **Email**: Steven.Pham03@student.csulb.edu

## Preface

| Version | Date | Changes |
| --- | --- | --- |
| 1.0 | 05/09/2024 | Initial Version |

### Purpose

This document is a comprehensive guide for the development and understanding of the PennyWise personal budget management system software project.

### Audience

The intended audience of this document includes developers, testers, and anyone involved in the project lifecycle.

## Introduction

### Project Overview

PennyWise is a personal budget management system designed for individuals seeking to streamline their financial tracking and planning. It offers tools for budget creation, expense categorization, and budget tracking.

### Project Goals

* Improve the process of tracking expenses.
* Provide consistent updates on changes in spending.
* Provide customization via budget category creation and spending limits.

## Glossary

* **Budget**: A group of similar categories.
* **Category**: A label used to group expenditures and incomes.
* **Expense**: Any transaction tracked by a budget.
* **Recurring Expenses**: Expenses that occur regularly, such as monthly bills and subscriptions.
* **Reset Date**: The date the budget cycle ends and a new cycle begins.
* **Financial Timeline**: specify a time frame for a budget or financial plan.
* **Budget Adjustment**: Changes to the budget based on actual expenditures.

## User Requirements and Use Cases

### User Stories

1. As a potential future user, I want to be able to register an account so that I can manage my expenses.
2. As a registered user, I want to be able to add a new expense to a budget that I create so I can accurately track my spending.
3. As a registered user, I want to be able to count the expenditures that I log during a session so that I don’t have to recount to verify if everything has been inputted.
4. As a registered user, I want to be able to receive notifications when my current expenditure rates are different from the previous budget period(s) so I can understand my habits and plan accordingly.
5. As a registered user, I want to be able to set up a monthly budget for different categories so that I can manage my spending and allocate the proper funds to that category.
6. As a registered user, I want to be able to set a budget to be within a specific time period so that I can manage different financial timelines and to prioritize specific budgets and expenses.
7. As a registered user, I want to be able to set a date for when the budget resets so that I do not have to manually reset the budget and I know when I can start spending within that budget.
8. As a registered user, I need to be able to delete or edit budget categories so that I can reflect changes in my spending lifestyle.
9. As a privacy-focused registered user, I need to be able to frequently change my account password so that I can bolster the security of my account.

### Use Case: Adding a New Expense

| Identifier | UC-2 Add new Expense to Budget |
| --- | --- |
| Purpose | Update budget category with new expense and cost. |
| Requirements | User Story #2 |
| Development Risks | None |
| Pre-conditions | Registered user is logged in and on add expenses page. |
| Post-conditions | New expense is added to the budget category. |

***Table 1: Typical Course of Action***

| Seq# | Actor’s Action | System’s Response |
| --- | --- | --- |
| 1 | On Add Expenses page, registered user fills in expense information in the form fields. |  |
| 2 | On Add Expenses page, user clicks ‘Add’ on the form. |  |
| 3 |  | Runs checks on information in submitted form |
| 4 |  | Checks are successful |
| 5 |  | New expense is added to the database and budget calculations are updated. |

***Table 2: Alternate Course of Action***

| Seq# | Actor’s Action | System’s Response |
| --- | --- | --- |
| 1 | On Add Expenses page, registered user fills in expense information in the form fields. |  |
| 2 | On Add Expenses page, user clicks ‘Add’ on the form. |  |
| 3 |  | Runs checks on information in submitted form |
| 4 |  | Check fails and system alerts the user via an alert box that one or more fields have been left blank. |
| 5 | User clicks the ‘OK’ option on the alert box to close the message. |  |
| 6 | User fills in expense information in the form fields. |  |
| 7 | User clicks ‘Add’. |  |
| 8 |  | Runs checks on information in submitted form |
| 9 |  | Checks are successful |
| 10 |  | New expense is added to the database and budget calculations are updated. |

***Table 3: Exceptional Course of Action***

| Seq# | Actor’s Action | System’s Response |
| --- | --- | --- |
| 1 | On Add Expenses page, user selects a category for the expense via a dropdown menu, and inputs a non-numeric value into the amount field. |  |
| 2 | On Add Expenses page, user clicks ‘Add’ on the form. |  |
| 3 |  | Runs checks on information in submitted form |
|  |  | Checks fail and system alerts the user via an alert box that the amount field must only contain numeric values. |
| 4 | User clicks the ‘OK’ option on the alert box to close the message. |  |
| 5 |  | System highlights string in amount field. |
| 6 | User clicks the application home button. |  |
| 7 |  | System displays the application home page. |

### Use Case: Receiving a Notification

| Identifier | UC-9: Receive Notification |
| --- | --- |
| Purpose | Alert users of approaching spending limits within the allotted time period. |
| Requirements | Users should be notified of their spending habits at certain thresholds. |
| Development Risks | None. |
| Pre-Conditions | User is logged in and is about to enter a new expense into an existing category with a spending limit. |
| Post-Conditions | New expense is added to the budget and a notification is displayed. |

***Table 1: Typical Course of Action***

| Seq# | Actor’s Action | System’s Response |
| --- | --- | --- |
| 1 | User fills in and submits information for a new expense. |  |
| 2 |  | System runs checks on inputs, and the checks pass. |
| 3 |  | System adds new expense to the database and updates budget calculations. |
| 4 |  | Budget calculates that the user has exceeded 50% of their allocated spending for that category. |
| 5 |  | A notification displays that 50% of the allocated funds has been spent, along with the next renewal date. |

***Table 2: Alternate Course of Action***

| Seq# | Actor’s Action | System’s Response |
| --- | --- | --- |
| 1 | User fills in and submits information for a new expense, but leaves the amount field blank. |  |
| 2 |  | System runs checks on inputs, and the checks fail. |
| 3 | User correctly fills in all fields and submits information. |  |
| 4 |  | System runs checks on inputs, and the checks pass. |
| 5 |  | System adds new expense to the database and updates budget calculations. |
| 6 |  | Budget calculates that the user has exceeded 50% of their allocated spending for that category. |
| 7 |  | A notification displays that 50% of the allocated funds has been spent, along with the next renewal date. |

***Table 3: Exceptional Course of Action***

| Seq# | Actor’s Action | System’s Response |
| --- | --- | --- |
| 1 | User mistakenly fills in expense comment field with the spent amount, and leaves the amount field at $0.00. |  |
| 2 | User submits the information for the expense. |  |
| 3 |  | System runs checks on inputs, and the checks pass. |
| 4 |  | System adds new expense to the database and updates budget calculations to show an added expense of $0.00. |
| 5 | User navigates to the application home page. |  |
| 6 |  | The system displays the home page. |

## System Architecture

### 

* **Frontend**: Users interact with the web application through a web browser.
* **Request**: User actions prompt the frontend to send requests to the backend to process or retrieve information.
* **Backend**: Requests are received by a web server, which has a database connection to store and retrieve persistent data for the application.
* **Response**: The web server sends responses back to fulfill sent requests.

### Components

1. **Frontend**: Web-based UI (built with React and Javascript)
2. **Backend**: RESTful API (built with Spring Boot and Java)
3. **Database**: MongoDB for data storage
4. **Authentication**: Spring Security