Team 4 (PennyPal)

1. **Tasneem Iqbal**
    - o **Student ID**: 033181614
    - o **CSULB Email**: tasneem.iqbal01@student.csulb.edu
2. **Allison Adoum**
    - o **Student ID**: 033465755
    - o **CSULB Email**: allison.adoum01@student.csulb.edu
3. **Kent Do**
    - o **Student ID**: 031089810
    - o **CSULB Email**: kent.do01@student.csulb.edu
4. **Jean Guedes Sobreira**
    - o **Student ID**: 030840561
    - o **CSULB Email**: jeanguedessobreira01@student.csulb.edu
5. **Frankie Guzman**
    - o **Student ID**: 032919664
    - o **CSULB Email**: frankie.guzman01@student.csulb.edu

Preface

Version	Date	Changes
1.0	3/18/2025	Initial Version
1.1	3/18/2025	Added Glossary
2.0		Fixed version date notation, …

Purpose

This document serves as a comprehensive guide for the development and understanding of the software project titled "PennyPal”.

Audience

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

Introduction

Project Overview

PennyPal is a web-based **Personal Budget Management System** that helps users track expenses, set budgets, and achieve financial goals. It offers real-time insights and alerts to ensure better control over spending.

Project Goals

- **Simplify Budget Management** – Provide an intuitive web interface for users to easily track income, expenses, and savings goals.
- **Enhance Financial Awareness** – Offer detailed insights and visual reports to help users analyze spending patterns and make informed decisions.
- **Promote Financial Discipline** – Enable budget alerts and notifications to prevent overspending and encourage consistent savings habits.

Glossary

- **API**: Application Programming Interface.
- **Budget:** A financial plan that helps users allocate their income towards expenses and savings.

User Requirements and Use Cases

User Stories

1. As a business owner, I want to securely log in so that my financial data remains protected.
2. As a freelance bartender, I want to add income and expenses so that I can keep track of my financial activities.
3. As a recently married man, I want to categorize my transactions so that I can better understand where my money is going.
4. As a travel influencer, I want to set a monthly budget for different categories so that I can control my spending.
5. As a treasure hunter, I want to receive notifications when I am close to exceeding my budget so that I can adjust my spending habits.
6. As a future homeowner, I want to set and track savings goals so that I can stay motivated and work toward achieving them.
7. As a statistics professor, I want to view visual reports and analytics so that I can analyze my spending trends over time.
8. As a tax client, I want to export my financial data so that I can share or back it up if needed.
9. As a film reviewer, I want to set recurring transactions so that I don't have to manually input fixed expenses.
10. As an elderly man, I want to reset my password if I forget it so that I can regain access to my account.

Use Case: Adding a New Product

Identifier	[E.g., "UC-1 Book Flight"]
Goal	[The initiating actor's goal]
Requirements	[Requirements that are addressed by this use case]
Initiating Actor	[Actor who initiates interaction with the system]
Participating Actor(s)	[Other actors who participate, if any]
Pre-conditions	[The state of the system before the interaction can start]
Post-conditions	[What must be true about the system after the goal is achieved or abandoned]
Included Use Case(s)	[Use cases that are included by this use case, if any]
Extension(s)	[Extensions of this use case, if any]

Table 1: Typical Course of Action

Seq#	Actor's Action	System's Response
1	Enter expense description	
2	Enter value	
3	Submit	
4		Checks submitted information
5		Determines if information is valid
6		Commits user information to user database
7		Displays confirmation message stating "Expense logged successfully"

Table 2: Alternate Course of Action

Seq#	Actor's Action	System's Response
1	Enter expense description	
2	(Did not enter value)	
3	Click Submit	
4		Checks submitted information
5		Determines if information is missing
6		Information missing, prompts to enter the required value
7	Enter missing value	
8		Repeat steps 4, 5, and display a success message

Table 3: Exceptional Course of Action

Seq#	Actor's Action	System's Response
1	Enter expense description	
2	Enter values	
3	Does not send (click submit) / Closes app	
4		Waits a time for submission
5		Times out! / Logged out!
6	Reopens app	
7		Display message: "Information was not submitted, please restart."

System Architecture

(Describe the high-level design of the software.)

Components

Deployment Diagram