Assignment 1

CSE214 - Software Engineering Sessional
Md. Ashrafur Rahman Khan
July 2025

Due: 13 December 2025

1   Introduction

Personal budget tracking is an essential skill in modern life, and well-designed software makes this task easier to maintain and extend. This assignment provides hands-on experience with refactoring real, working code to improve its quality without changing its behavior. You will identify recurring problems in the codebase and apply appropriate solutions to eliminate duplication and improve maintainability.

2   Project Overview

2.1   What is BudgetBuddy?

BudgetBuddy is a command-line application for tracking personal expenses. It provides the following features:

- Load CSV expense files

- List and filter expenses

- Generate monthly and category summaries

- Export reports in plain text and HTML formats

The application is a real, runnable program with approximately 600 lines of code. It has no external dependencies and uses only pure Java SE.

2.2   Running the Application

First clone the repository from GitHub:

```
git clone https://github.com/risenfromashes/budgetbuddy
cd budget-buddy
```

Compile the application:

```
javac -d out src/**/*.java
```

Run the application:

```
java -cp out Main data/expenses.csv
```

Sample commands:

- `load data/expenses.csv` - Load expense data

- `list` - Display all expenses

- `summary month 2025-02` - Show monthly summary

- `export txt out/report.txt` - Export text report

- `export html out/report.html` - Export HTML report

2.3 Sample Data

The `data/expenses.csv` file contains 71 realistic expense entries with the format:

```
date;category;amount;notes
2025-01-03;Food;520;Lunch at cafe
2025-01-05;Transport;150;Uber
```

3 Code Structure

3.1 Directory Layout

```
src/
  Main.java
  model/
    Expense.java
  io/
    CsvLoader.java
    TxtReportWriter.java
    HtmlReportWriter.java
  service/
    ExpenseRepository.java
    Summarizer.java
  cli/
    Cli.java
    CommandHandler.java
  util/
    DateUtils.java
    TextUtils.java
data/
  expenses.csv
out/
  (compiled classes)
```

3.2 Key Classes

- **Expense**: Immutable data model (date, category, amount, notes)

- **CsvLoader**: Parses CSV files into Expense objects

- **ExpenseRepository**: In-memory storage for expenses

- **Summarizer**: Computes monthly and category totals

- **TxtReportWriter**: Generates plain-text reports with ASCII bars

- **HtmlReportWriter**: Generates HTML reports with tables

- **CommandHandler**: Processes CLI commands and coordinates services

- **Cli**: Main command loop and user interaction

- **DateUtils**: Date parsing and formatting helpers

- **TextUtils**: Text formatting and ASCII visualization

4 Design Issues

Recently the code has been reviewed by your friend who studied software engineering. They identified two issues which you need to address in this assignment.

4.1 Issue 1: Redundant Object Creation

Some classes like the `ExpenseRepository` and `CsvLoader` are instantiated multiple times across different parts of the codebase. But they do the same thing. Is it efficient? Does it maintain a consistent state?

4.2 Issue 2: Duplicated Visualization Logic

The writer classes (`TxtReportWriter` and `HtmlReportWriter`) both implement similar logic to format and write the expense reports. Seperating formatting and writing logic might be better, while making sure that appropriate formatting formatting used for each case.

4.3 Objective

Your task is to refactor the BudgetBuddy codebase to eliminate the design issues described above while preserving all existing functionality. The refactored code must produce identical outputs to the original. You can use `TestHarness.java` to verify.

5 Submission

Submit a ZIP archive containing the entire refactored `src/` directory. Ensure that your code compiles and runs correctly.

6 Grading Rubric

Criterion	Points	Details
Correctness & Functionality	20	Code compiles, all commands work, outputs match baseline, no regressions
Design Issue 1 Resolution	40	Addressing the issue with the correct design pattern
Design Issue 2 Resolution	40	Addressing the issue with the correct design pattern
TOTAL	**100**	