Software Requirements Specification
Plutos: Smart Budgeting Expense Tracker

Team #10, Plutos
Payton Chan
Eric Chen
Fondson Lu
Jason Tan
Angela Wang

October 11, 2024

Contents

Revision History

Date	Version	Notes
10/07/2024	0.01	Added in the following sections to the SRS: Reference Materials (1), NFRs (5.2), Likely Changes (6), Unlikely Changes (7), Development Plan (9), Auxiliary Constraints (10)
10/08/2024	0.02	Added in the following sections to the SRS: Purpose of Document (2.1), Organization of Document (2.4), User Characteristics (3.2)
10/09/2024	0.03	Added in the following sections to the SRS: Characteristics of Intended Reader (2.3), Functional Requirements (5.1)
10/10/2024	0.04	Added in the following sections to the SRS: System Constraints (3.3), Requirements Rationale (5.3)
10/11/2024	0.05	Enumerate requirements, clean up doc, review
10/11/2024	0.06	Add traceability matrix for requirements

1 Reference Material

This section records information for easy reference.

1.1 Table of Units

N/A; units are not used in this SRS.

1.2 Table of Symbols

N/A; symbols are not used in this SRS.

1.3 Abbreviations and Acronyms

symbol	description
AI	Artificial Intelligence
CI	Continuous Integration
FR	Functional Requirement
JWT	JSON Web Token
LC	Likely Change
ML	Machine Learning
NFR	Non-Functional Requirement
NLP	Natural Language Processing
OCR	Optical Character Recognition
SRS	Software Requirements Specification
ULC	Unlikely Change
UI	User Interface

1.4 Mathematical Notation

N/A; mathematical notation is not used in this SRS.

2 Introduction

2.1 Purpose of Document

The purpose of this SRS document is to describe the functional and non-functional requirements of the Plutos budgeting application. This document serves as a formal agreement between stakeholders, including developers, project managers, and end users, to ensure a shared understanding of the system's objectives, capabilities, and constraints.

The SRS defines the expectations for the project, detailing the system features, behaviour, and performance requirements. It serves as a reference throughout the development lifecycle, guiding the design, implementation, testing, and validation phases to ensure the final product meets the agreed-upon specifications. Additionally, this document will support future maintenance and scalability of the application by providing clear and detailed requirements that facilitate ongoing enhancements.

2.2 Scope of Requirements

The scope of this project encompasses the development of a budgeting application, Plutos, that automates expense tracking and categorization using AI. The system is designed to address key pain points for young adults struggling to manage their finances, specifically focusing on automating the process of tracking spending through receipt scanning, categorization of expenses, providing metrics to users regarding their spending habits, and providing feedback on how users can meet their budgeting goals.

However, some features that are outside the scope of the project (though may be implemented later) are:

- The system will not account for complex financial scenarios such as investments, stock portfolios, or retirement planning.
- Advanced financial forecasting or predictive analytics beyond simple budgeting trends will not be implemented.
- The AI model will not handle non-standard or highly complex receipts (e.g., multi-page invoices or handwritten receipts).
- The application will not offer integration with external financial accounts such as credit cards or bank accounts.
- The model will be trained to detect receipt data from Fortinos and will not initially support various receipt formats.

These exclusions allow the project to concentrate on core features, such as receipt scanning and expense categorization, while ensuring a manageable scope for the development process.

2.3 Characteristics of Intended Reader

The primary audience for this SRS document is the group of undergraduate software engineering students who will oversee and complete the project's design and implementation. This group of specialists, which includes system architects, software developers, and quality assurance testers, has extensive experience in the range of technologies needed for the project. Through their coursework and personal experiences, they are familiar with the process of developing mobile applications, and as they move through the project milestones, they will deepen their understanding of artificial intelligence (AI) and machine learning (ML). They will actively learn how to apply AI/ML technologies in real-world circumstances throughout the project, which will advance their development as software engineers. The collaborative effort will help them improve their real-world application development abilities as the project progresses through seven milestones, ensuring that they are prepared to produce a practical and user-friendly budgeting solution.

2.4 Organization of Document

The following sections of the SRS will further describe the system and its requirements. The sections will be as follows:

3. General System Description
4. Specific System Description
5. Requirements
6. Likely Changes
7. Unlikely Changes
8. Traceability Matrices and Graphs
9. Development Plan
10. Values of Auxiliary Constants

3 General System Description

This section provides general information about the system. It identifies the interfaces between the system and its environment, describes the user characteristics and lists the system constraints.

3.1 System Context

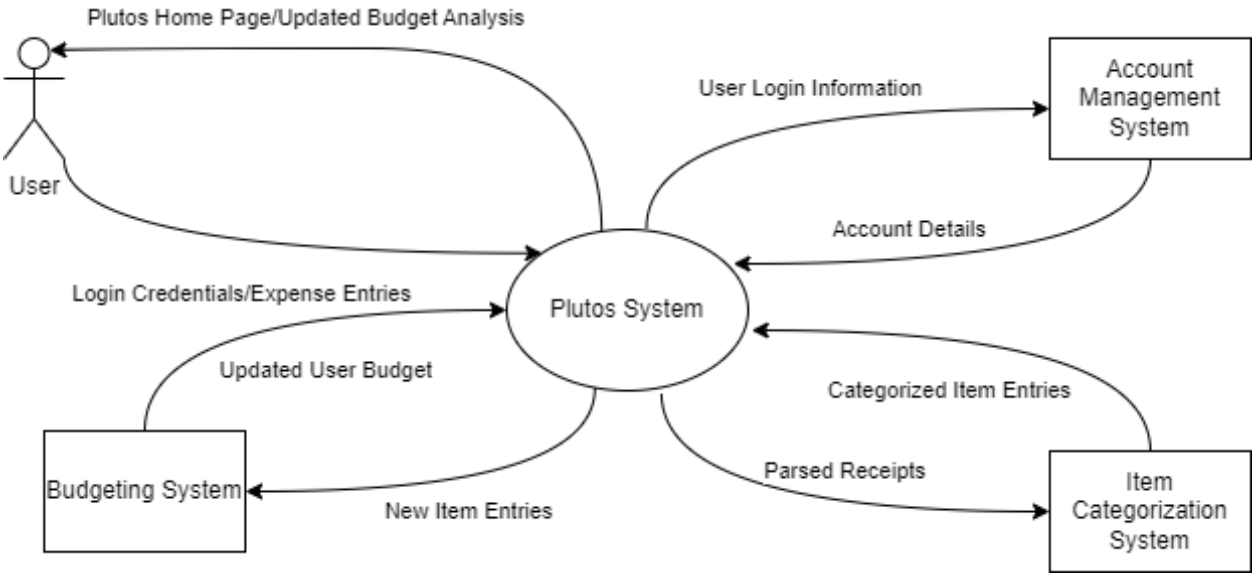

Figure 1: System Context

The system context for this project consists of a user(s), an expense tracking system, a budgeting analysis generator, and several databases. The user provides expense tracking inputs, such as physical/digital receipts or manual expense entries, and receives financial assessments. The expense tracking system processes user inputs, generates budgetary evaluations, and interacts with the financial analysis program and databases. The expense tracking system is also responsible for ensuring the validity of all inputs and provides a response if an invalid input is identified. The budgeting analysis generator provides expense categorization and evaluation functionalities, such as data analysis and visualization, and supports the expense tracking system in generating outputs. The database will be used to store user account information, user budget data, and supports the expense tracking system in storing and retrieving specific items and their respective categories. The system will typically be used for personal finance management, budgeting, and expense tracking, and may also be used for educational purposes. While the system is not safety-critical, it is important to ensure that the system is reliable and accurate in its calculations and analysis to provide users with trustworthy insights into their financial situation.

3.2 User Characteristics

The users of the *Plutos* budgeting application can be grouped into four main categories based on their financial experience and needs. These groups represent varying levels of financial literacy and desired interaction with budgeting tools.

1. **First and Second-Year Undergraduate Students [Primary]**
   This group consists of young adults who are new to managing their finances independently, such as those living away from home for the first time. The characteristics of these users are as follows:

   - **Financial Knowledge:** Limited experience with budgeting, managing expenses like rent and groceries. Usually have limited financial independence.
   - **Time Management:** Busy schedules juggling school, work, and social commitments. Likely to prefer simple, intuitive interfaces that reduce time spent on budgeting.
   - **Pain Points:** Lack of education on budgeting, potential to overspend due to unfamiliarity with managing day-to-day expenses.

2. **Upper-Year Undergraduate and Post-Graduate Students [Secondary]** This group includes more experienced students who have more financial independence as they have had experience managing their finances (in previous years living independently). They have a better understanding of budgeting and have developed more mature spending habits.

   - **Financial Knowledge:** Some experience balancing school and work, likely to have a clearer understanding of budgeting needs and differentiating between wants and needs.
   - **Time Management:** Likely to have more experience managing limited time and money, though they may still struggle with large financial decisions such as housing or loans.
   - **Pain Points:** May underestimate or overestimate budget needs, particularly with larger expenses.

3. **Early Career Professionals [Tertiary]** New graduates or individuals recently introduced to the workforce fall into this category. They likely have more stable incomes and financial independence.

   - **Financial Knowledge:** More knowledgeable about saving, budgeting, and managing recurring expenses but still learning to navigate significant financial decisions.
   - **Time Management:** May experience stress due to work-related issues and life changes such as moving to a new city and budgeting/tracking expenses may be another stress inducer on top of the new environment.

- **Pain Points:** Struggles with planning for long-term financial goals or managing joint accounts with a partner.

4. **Retirees** [**Tertiary**] Although retirees are not a primary focus, they may use the application to simplify financial tracking and planning for a fixed income.

    - **Financial Knowledge:** Likely to have extensive experience with financial management but may struggle to adapt to modern budgeting tools.
    - **Time Management:** Older users may face physical limitations (e.g., difficulty typing or navigating) and slower adaptation to new technologies.
    - **Pain Points:** Difficulty managing finances without a steady income and adapting to increased living expenses.

3.3 System Constraints

C1 The system must be developed as a mobile application using a cross-platform framework (e.g., React Native, Flutter) to ensure compatibility with both iOS and Android devices.

C2 The system must utilize a cloud-based backend technology (e.g., AWS, Firebase) to provide scalability, reliability, and compatibility across environments.

C3 The system must integrate with a third-party OCR library (e.g., Tesseract OCR, Google Vision API) to provide OCR functionality. This is a constraint because it is not necessary to reinvent the wheel and build an OCR system from scratch.

C4 The system must implement secure password storage and authentication mechanisms using a trusted third-party authentication service (e.g., Auth0, Firebase). This is to ensure secure user data handling and privacy.

C5 The project must implement a Continuous Integration (CI) pipeline using GitHub Actions that automatically runs tests, checks code quality, and verifies builds with every commit. This is to ensure that the code is always in a deployable state.

C6 The system shall use a dependency management tool (e.g., npm for JavaScript, pip for Python) to track external libraries. This is to ensure organized dependency management across different working environments.

4 Specific System Description

This section first presents the problem description, which gives a high-level view of the problem to be solved. Following this, a solution characteristics specification is typically included; however this aspect is not applicable for this SRS.

4.1 Problem Description

The problem description can be found in Section 1 of the Problem Statements and Goals document.

4.1.1 Terminology and Definitions

This subsection provides a list of terms that are used in the subsequent sections and their meaning, with the purpose of reducing ambiguity and making it easier to correctly understand the requirements:

- **User Database**: A secure storage location for user information, including usernames, passwords, and personal financial profiles.
- **Item Categorization**: The classification of expenses and income into distinct categories, such as groceries, rent, and salary, to clarify spending habits.
- **Expense Tracking**: The practice of recording and monitoring expenditures to understand spending patterns and make informed financial decisions.
- **Budget Data Analysis**: The examination of financial data related to budgets to identify trends, patterns, and areas for improvement.
- **Personalized Budget Plan**: A customized budget tailored to an individual's specific financial goals, income levels, and spending habits.
- **Account Budget Database**: A system for storing and managing users' budget data, including allocations, spending limits, and financial goals.
- **Digital Financial Tools**: Software applications designed to help users manage their finances, including budgeting, tracking expenses, and analyzing data.
- **Data Security**: Measures taken to protect personal financial information from unauthorized access or breaches.
- **User-Friendly Interface**: An application design that prioritizes ease of use and accessibility, making it simple for users to navigate financial tools.

4.1.2 Physical System Description

The physical system of Plutos, includes the following elements:

User Interface Elements

PS1 **Camera Interface**: A feature that enables users to capture images of receipts using their mobile device's camera.

PS2 **Receipt Preview**: A visual display showing the captured receipt for user verification before processing.

PS3 **Categorization Display**: An interface that shows identified items along with suggested categories for each item.

PS4 **Forms**: Input fields for entering financial data, such as income and expenses.

PS5 **Graphs and Charts**: Visual representations of spending patterns and budget performance, providing insights into financial health.

Data Processing Components

PS6 **Budgeting Algorithm**: An engine that analyzes user data to suggest personalized budgets based on historical spending.

PS7 **Notification System**: A feature that alerts users about important financial events, such as nearing budget limits.

PS8 **Optical Character Recognition (OCR)**: A technology that analyzes the scanned receipt image to extract text, identifying items, prices, and totals.

PS9 **Item Categorization Engine**: A module that automatically classifies extracted items into predefined categories (e.g., groceries, clothing, dining) based on user-defined rules or machine learning algorithms.

PS10 **Database**: A secure storage system for user data, including receipts, categorized items, and budget information.

Users interact with the application by entering financial data, setting budgets, and reviewing visualizations, using the User Interface Elements (PS1-PS5). The system responds by updating financial summaries, processing the data, and providing insights based on user behaviour (PS6-PS10).

4.1.3 Goal Statements

The goal statements can be found in Section 2 of the Problem Statement and Goals document.

4.2 Solution Characteristics Specification

This section is not applicable to the SRS; the solution characteristics are defined through the functional and nonfunctional requirements in Section 5 of this document.

4.2.1 Types

Not applicable.

4.2.2 Scope Decisions

Not applicable.

4.2.3 Modelling Decisions

Not applicable.

4.2.4 Assumptions

Not applicable.

4.2.5 General Definitions

Not applicable.

4.2.6 Data Definitions

Not applicable.

4.2.7 Data Types

Not applicable.

4.2.8 Instance Models

Not applicable.

4.2.9 Input Data Constraints

Not applicable.

4.2.10 Properties of a Correct Solution

Not applicable.

5 Requirements

This section provides the functional requirements, the business tasks that the software is expected to complete, and the nonfunctional requirements, the qualities that the software is expected to exhibit.

5.1 Functional Requirements

User Account Management

FR1 Users must be able to create an account using a name, email address, and password.

FR2 Users must be able to update their account information after account creation.

FR3 Users must be able to log in using their registered email and password.

FR4 Users must log in before accessing the application.

FR5 Users must be able to log out of their account.

FR6 Users must be able to reset their password if forgotten.

Receipt Scanning Input

FR7 Users must be able to take a picture using the application.

FR8 Users must be able to upload an image from their device to the application.

FR9 The system shall allow a limit of 5 MB per receipt.

FR10 The system must display a preview of the uploaded image, allowing users to confirm or retake the image if necessary.

FR11 The system shall upload the confirmed image to the server for processing.

Manual Receipt Input

FR12 Users must be able to manually input receipt details. This includes items, their associated costs, and the date of purchase.

FR13 The system shall validate the input to ensure required fields are completed.

Database Management

FR14 The system shall store user account information, including usernames, email addresses, and passwords, in a secure database.

FR15 The system shall store receipt images and extracted data in a secure database.

Item Recognition and Categorization

FR16 The system shall identify and display item names, quantities, and costs from the uploaded receipt image.

FR17 The system shall sort the entered products into predefined categories.

FR18 The user must be able to modify the name, quantity, price, and category of the identified items.

Financial Tracking

FR19 The system shall generate an overview of the user's spending history, including total spending by item category and trends over time.

FR20 Users shall be able to set up budgets for each spending category.

FR21 Users shall be able to view their spending in relation to their set budgets.

FR22 The system shall notify users when they reach 80% of their set budget limit.

FR23 The system shall notify users when they achieve specified savings goals.

FR24 The system shall store up to 3 years of user transaction history.

5.2 Nonfunctional Requirements

Accuracy Requirements

NFR1 The machine learning model must achieve at least 80% accuracy in correctly identifying and extracting key information (i.e., item names, quantities, prices, dates) from receipts.

NFR2 The machine learning model must categorize expenses into predefined categories (e.g., groceries, utilities, entertainment) with at least 90% precision to ensure users' financial data is correctly organized.

NFR3 All monetary calculations (e.g., totals, budgets, currency conversions) must maintain a precision of up to two decimal places to ensure accurate financial reporting.

NFR4 The OCR model must correctly recognize text from images of receipts with a minimum accuracy rate of 90%, ensuring minimal manual corrections by users.

NFR5 The application must guarantee 99% accuracy in summing up expenses, incomes, and savings across different periods and categories, ensuring no rounding or summation errors.

NFR6 If the application syncs data across multiple devices, it should ensure 100% consistency of financial data across all instances.

Performance Requirements

NFR7 The application must load user data within five seconds.

NFR8 The application must be responsive to user actions (e.g., adding an entry, generating a report) within two seconds.

NFR9 The system must be able to handle a minimum of 10 concurrent users without significant performance degradation.

Usability Requirements

NFR10 The application must have a clean, intuitive, and easy-to-navigate interface, allowing users to perform common tasks (e.g., adding expenses, viewing budgets) within two to three clicks.

NFR11 New users should be able to complete the account setup and understand core application features within five minutes.

NFR12 The application shall provide users with an introductory tutorial and tooltips during their first use.

NFR13 The application should provide clear, informative error messages upon invalid inputs.

NFR14 The application should guide users through corrective actions when incorrect input is detected.

NFR15 Users should be able to recover from errors (e.g., wrong data entry) with no more than two steps.

NFR16 Common user tasks, such as adding a new receipt or setting a budget limit, should be completable within 10 seconds on average, assuming all required information is readily available.

NFR17 Information displayed to users must be concise and relevant, minimizing the cognitive effort needed to understand their financial data. Only essential details should be shown on the main dashboard, with advanced options accessible through menus.

Security Requirements

NFR18 The system must implement secure password storage and authentication mechanisms.

NFR19 The system must ensure that all data transmitted between the client and server is encrypted.

Maintainability Requirements

NFR20 New releases must maintain backward compatibility with previous versions, ensuring that users can seamlessly update the application without experiencing disruptions in their data or workflows.

NFR21 Dependencies must be regularly updated (i.e., within major patches) to prevent security vulnerabilities and maintain compatibility with new features.

NFR22 At least 80% of the codebase must be covered by automated unit tests to ensure maintainability and minimize the risk of introducing bugs when making changes.

Portability Requirements

NFR23 The software shall be compatible with Android and iOS mobile devices running the latest software.

NFR24 All data must be stored in platform-independent formats (e.g., JSON, CSV) for easy transfer and compatibility between devices, databases, and systems.

NFR25 Any external APIs or third-party services integrated into the application must support cross-platform usage, ensuring the application can access necessary services from any supported platform.

NFR26 The system must be able to operate in a variety of network environments, including Wi-Fi, cellular networks, and offline mode.

Reusability Requirements

NFR27 The application must be developed using reusable components (e.g., UI components, API services) where applicable, to prevent code duplication and allow for better code maintainability.

NFR28 The application must adhere to the principle of separation of concerns where possible, ensuring that business logic, data access, and presentation layers are kept separate and allowing individual layers to be reused independently.

NFR29 Common functionalities (e.g., receipt parsing, authentication, data validation) must be abstracted into reusable libraries or modules that can be shared across multiple applications or systems.

NFR30 Design elements (e.g., icons, typography, color schemes) shall be created as reusable assets, ensuring they can be reused consistently across multiple projects or platforms.

Understandability Requirements

NFR31 All source code must be thoroughly documented using inline comments and external documentation (e.g., README files, docstrings) to ensure that developers can easily understand the purpose and behavior of code components.

NFR32 Variables, functions, classes, and modules must follow consistent and meaningful naming conventions (e.g., camelCase, snake_case) that clearly describe their functionality and usage, making the code easier to understand.

NFR33 The application's user interface (UI) must be designed with simplicity and clarity in mind, using clear labels, icons, and tooltips to guide users through tasks such as adding expenses or reviewing their budgets.

NFR34 The application's codebase must be organized logically, with related files and components grouped together in well-defined directories, so developers can easily navigate and locate specific functionality.

NFR35 The code must follow industry-standard formatting guidelines (e.g., proper indentation, line length limits) to ensure that it remains readable and easy to follow, both for developers and code reviewers.

NFR36 The application must provide clear, descriptive error messages (both for users and in logs) that explain the cause of an issue and provide actionable steps to resolve it, improving user and developer understanding.

NFR37 Any APIs developed for the application must include detailed and easy-to-understand documentation, describing available endpoints, request/response formats, and example usage scenarios, ensuring that developers can integrate with them easily.

NFR38 All major changes to the codebase should be accompanied by clear commit messages and detailed changelogs, explaining the purpose of changes and their impact, helping future developers understand the evolution of the project.

NFR39 The language and terminology used throughout the application's UI must be aligned with the users' mental models and expectations, using non-technical and familiar terms to enhance understanding.

Regulatory Requirements

NFR40 The system must comply with the Canada's Privacy Act to ensure the privacy and security of user data.

NFR41 The system must comply with Canada's Financial Administration Act to ensure the secure handling of financial information.

5.3 Rationale

5.3.1 Scope Decisions

The scope decisions were made with the intention of ensuring that the application delivers value to its target audience while remaining manageable within development constraints. The application focuses on personal budgeting features, including receipt scanning, expense categorization, budget tracking, and financial analysis.

Rationale:

- **User-Centered Design**: The scope focuses on features that users expect from a smart budgeting app, aligning with current market demands for personal finance tools that offer automation (e.g., receipt scanning) and financial insights.
- **Technical Feasibility**: Developing an ML model for parsing and categorizing receipts makes the project a feasible challenge for the team, without making it overly complicated.
- **Time and Resource Constraints**: By narrowing the scope to core personal finance features, the team ensures that the project can be completed within the specified timeline and budget.

5.3.2 Modeling Decisions

The application's architecture follows a modular, service-oriented design, with key components (i.e., receipt scanning, expense categorization, and budgeting reports) being developed as independent, reusable modules.

Rationale:

- **Scalability**: A modular approach allows the application to scale more easily by isolating functionalities into distinct services (e.g., a dedicated service for expense categorization). This reduces interdependencies and facilitates future growth.
- **Maintainability**: Modular components are easier to update and test, as changes to one service do not necessarily impact others. This also allows the application to accommodate feature updates and bug fixes more efficiently.
- **Reusability**: By designing reusable components (e.g., UI elements, API services), the application can easily extend its capabilities or integrate with other systems in the future.
- **Use of JSON**: JSON is lightweight, human-readable, and widely used for web and mobile applications. Its simplicity makes it ideal for transmitting structured data between the client and server, ensuring smooth communication with minimal overhead.

5.3.3 Assumptions

Several assumptions were made during the development and design process. These assumptions include the expectation that users will regularly scan receipts, that mobile users are the primary audience, and that financial data privacy is of utmost importance.

Rationale:

- **Receipt Scanning**: The assumption that users will consistently scan receipts underpins the application's core functionality. This assumption is based on trends observed in personal finance apps where automation is a key value proposition for users.
- **Mobile-First Audience**: It is assumed that the majority of users will access the application via mobile devices. This assumption is driven by the current dominance of mobile platforms for personal finance management, offering convenience and accessibility.
- **Privacy and Security**: The assumption that users expect high standards of data privacy and security informs the application's adherence to data protection regulations. Users handling sensitive financial data expect robust protection mechanisms.
- **Stable Internet Connection**: It is assumed that users will have access to a stable internet connection while using the app, especially for features that require cloud processing (e.g., receipt scanning, data syncing, and report generation).
- **Regular Application Usage**: The assumption is that users will interact with the application on a regular basis (e.g., weekly or monthly), allowing the application to collect sufficient user data and provide up-to-date financial insights and budgeting trends.
- **User Financial Literacy**: The application assumes a basic level of financial literacy among its users, meaning that they will be able to understand concepts like budgets, expenses, savings, and categories without extensive in-application education.

5.3.4 Typical Values

For certain features, typical values have been used to inform design and development, such as default budget categories, receipt parsing accuracy thresholds, and system performance benchmarks.

Rationale:

- **Default Categories**: Common budgeting categories (e.g., groceries, utilities, transportation) are provided as default to simplify user onboarding and offer a familiar framework. These categories were selected based on industry standards and user behavior in similar apps. This is also to simplify the ML model training process.

- **Parsing Accuracy**: An accuracy threshold of 90% for receipt parsing was established as the minimum acceptable standard, based on expected ML capabilities. This value ensures that the system can reliably extract key information from receipts without significant manual intervention.
- **Performance Benchmarks**: Typical values for system performance, such as load times of under two seconds for key operations (e.g., viewing reports), are set based on user experience research, ensuring that the application feels responsive and efficient.
- **Maximum Receipt Upload Size**: A limit of 5 MB per receipt ensures a balance between allowing high-resolution images for accurate OCR processing while keeping server storage and bandwidth requirements manageable.
- **User Transaction History Retention**: Storing up to 3 years of transaction history provides users with enough data to analyze long-term financial trends while limiting the storage requirements for each user. Older transactions could be archived or made available on demand.
- **Frequency of Budget Update Notifications**: Weekly budget update notifications help users stay on top of their spending without overwhelming them with too many alerts. Users are reminded to review their financial status regularly while avoiding notification fatigue.

6 Likely Changes

LC1 **Addition of New Features:** Based on user feedback, it is likely that new features (e.g., budgeting goal tracking, automatic expense categorization, or financial analytics) will be added to enhance user experience.

LC2 **User Interface Redesign:** As usability testing is conducted, it's likely that the UI will undergo several iterations to improve the overall user experience and incorporate user feedback.

LC3 **Integration with New APIs:** The application may need to integrate with new financial data APIs (e.g., bank transaction retrieval services) to enhance functionality, requiring changes to the codebase.

LC4 **Change in Tech Stack:** If the team encounters difficulties with the current technology stack, such as performance or compatibility issues, a transition to new technologies (e.g., switching from React to Vue.js) is likely.

LC5 **Performance Optimizations:** As the application scales, there may be a need for performance enhancements, such as optimizing database queries or implementing caching strategies.

LC6 **Data Privacy Compliance Updates:** Changes in legal requirements may require updates to the application's data handling and privacy policies to ensure compliance.

LC7 **Updates to User Authentication Method:** There may be a transition to a more secure authentication method (e.g., implementing two-factor authentication) based on user feedback and security best practices.

LC8 **Feedback Mechanism for Users:** Adding a feedback mechanism within the application (e.g., surveys or feedback forms) to gather user insights and suggestions for improvements is likely, ensuring continuous enhancement of the application based on user needs.

LC9 **Change in Database Choice:** During the development phase, the initial database choice may need to change due to performance issues or scalability requirements (e.g., moving from SQLite to PostgreSQL or MongoDB).

7 Unlikely Changes

ULC1 **Complete Rewrite of the Application:** A complete rewrite of the application's codebase is unlikely unless there are significant fundamental flaws that cannot be addressed through refactoring.

ULC2 **Switching Platforms:** Transitioning from a mobile-first approach to a purely desktop application is unlikely if the initial target audience is primarily mobile users.

ULC3 **Adopting a New Programming Language:** Changing the programming language for the entire project (e.g., from JavaScript to Ruby) midway through development is unlikely due to the complexity and resource investment required.

ULC4 **Elimination of Existing Features:** Removing core features that are central to the application's purpose (e.g., expense tracking) is unlikely, as these are fundamental to user expectations and functionality.

ULC5 **Change in Target Audience:** A shift in the target audience (e.g., from personal budgeting to corporate finance management) is unlikely as it would require a fundamental reevaluation of the application's design, features, and use cases.

ULC6 **Discontinuation of Data Storage:** Completely removing any form of data storage (e.g., opting for a stateless application) is unlikely, as the core functionality relies on data retention for budgeting purposes.

ULC7 **Pivot to a Social Networking Feature Set:** A major pivot to add social networking features (e.g., allowing users to connect with friends or share budgets) is unlikely, as it diverges from the core functionality of personal budgeting and may complicate the application's primary purpose.

8 Traceability Matrices and Graphs

The purpose of the traceability matrices is to provide easy references on what has to be additionally modified if a certain component is changed. Every time a component is changed, the items in the column of that component that are marked with an "X" may have to be modified as well.
The traceability matrix can be found in the Traceability Matrix Excel

9 Development Plan

The development plan can be found in the Development Plan document.

Note that information in this document, especially regarding dates and deadlines, are subject to change and will be updated accordingly.

10 Values of Auxiliary Constants

There are no symbolic parameters used in this report.

References

Daniel M. Hoffman and Paul A. Strooper. *Software Design, Automated Testing, and Maintenance: A Practical Approach*. International Thomson Computer Press, New York, NY, USA, 1995. URL http://citeseer.ist.psu.edu/428727.html.

W. Spencer Smith. Systematic development of requirements documentation for general purpose scientific computing software. In *Proceedings of the 14th IEEE International Requirements Engineering Conference, RE 2006*, pages 209–218, Minneapolis / St. Paul, Minnesota, 2006. URL http://www.ifi.unizh.ch/req/events/RE06/.

W. Spencer Smith and Nirmitha Koothoor. A document-driven method for certifying scientific computing software for use in nuclear safety analysis. *Nuclear Engineering and Technology*, 48(2):404–418, April 2016. ISSN 1738-5733. doi: http://dx.doi.org/10.1016/j.net.2015.11.008. URL http://www.sciencedirect.com/science/article/pii/S1738573315002582.

W. Spencer Smith and Lei Lai. A new requirements template for scientific computing. In J. Ralyté, P. Ågerfalk, and N. Kraiem, editors, *Proceedings of the First International Workshop on Situational Requirements Engineering Processes – Methods, Techniques and Tools to Support Situation-Specific Requirements Engineering Processes, SREP'05*, pages 107–121, Paris, France, 2005. In conjunction with 13th IEEE International Requirements Engineering Conference.

W. Spencer Smith, Lei Lai, and Ridha Khedri. Requirements analysis for engineering computation: A systematic approach for improving software reliability. *Reliable Computing, Special Issue on Reliable Engineering Computation*, 13(1):83–107, February 2007.

W. Spencer Smith, John McCutchan, and Jacques Carette. Commonality analysis for a family of material models. Technical Report CAS-17-01-SS, McMaster University, Department of Computing and Software, 2017.

Appendix — Reflection

[Not required for CAS 741 —SS]

The information in this section will be used to evaluate the team members on the graduate attribute of Lifelong Learning.

The purpose of reflection questions is to give you a chance to assess your own learning and that of your group as a whole, and to find ways to improve in the future. Reflection is an important part of the learning process. Reflection is also an essential component of a successful software development process.

Reflections are most interesting and useful when they're honest, even if the stories they tell are imperfect. You will be marked based on your depth of thought and analysis, and not based on the content of the reflections themselves. Thus, for full marks we encourage you to answer openly and honestly and to avoid simply writing "what you think the evaluator wants to hear."

Please answer the following questions. Some questions can be answered on the team level, but where appropriate, each team member should write their own response:

1. What went well while writing this deliverable?

2. What pain points did you experience during this deliverable, and how did you resolve them?

3. How many of your requirements were inspired by speaking to your client(s) or their proxies (e.g. your peers, stakeholders, potential users)?

4. Which of the courses you have taken, or are currently taking, will help your team to be successful with your capstone project.

5. What knowledge and skills will the team collectively need to acquire to successfully complete this capstone project? Examples of possible knowledge to acquire include domain specific knowledge from the domain of your application, or software engineering knowledge, mechatronics knowledge or computer science knowledge. Skills may be related to technology, or writing, or presentation, or team management, etc. You should look to identify at least one item for each team member.

6. For each of the knowledge areas and skills identified in the previous question, what are at least two approaches to acquiring the knowledge or mastering the skill? Of the identified approaches, which will each team member pursue, and why did they make this choice?