Software Requirements Specification
Plutos: Smart Budgeting Expense Tracker

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

April 4, 2025

Contents

List of Tables

Revision History

Table 1: 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
10/11/2024	0.07	Add reflection to SRS
03/08/2025	0.08	Link Requirements Elicitation Report
03/08/2025	0.09	Move constants to Section 10
04/02/2025	0.10	Link rationale and (un)likely changes with 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.

The requirements specified in this document are based on the findings from the requirements elicitation process with end users, as detailed in the Requirements Elicitation Report.

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

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

Receipt Image Tensor:
The input to the OCR model is a 3D tensor $I \in \mathbb{R}^{H \times W \times C}$ where H is height, W is width, and $C = 3$ for RGB channels.

Detected Item (in receipt):
Each detected item d_i from a receipt is represented as a tuple:

$$d_i = (n_i, c_i, p_i) \quad \text{where:}$$

- n_i is the item name (string)

- c_i is the category (e.g., groceries)

- p_i is the price $\in \mathbb{R}^{\geq 0}$

4.2.7 Data Types

Not applicable.

4.2.8   Instance Models

Let α be the input image. The OCR model extracts a set of detected characters $T = \{t_1, t_2, \ldots, t_n\}$ with corresponding confidence scores $P = \{p_1, p_2, \ldots, p_n\}$.
Each text element t_i is predicted using a convolutional neural network (CNN) with a softmax layer:

$$P(t_i \mid \alpha) = \text{softmax}(Wf(\alpha) + b)$$

where $f(\alpha)$ is the image feature vector, W and b are learnable weights and biases.
Budget Performance Calculation
$BudgetProgress = \frac{TotalSpent}{BudgetLimit} \times 100\%$
where $TotalSpent$ is the total amount spent in a category and $BudgetLimit$ is the maximum budget set by the user for the current month.

4.2.9   Input Data Constraints

Input data constraints are as follows:

- Receipt image: $I \in [0, 255]^{H \times W \times 3}$, where $H \le 2000$, $W \le 2000$

- Max image file size: $S \le MAX\_RECEIPT\_FILE\_SIZE$

4.2.10   Properties of a Correct Solution

Not applicable.

4.2.11   Undesired Event Handling

While the model is designed to be highly accurate, occasional errors may occur. To address this, users have the ability to review and correct any inaccuracies through an edit feature. This allows them to manually adjust expense items if they believe the generated information is incorrect. By providing this functionality, the application helps ensure that users maintain control over their financial records, improving overall accuracy while accomodating variations in our OCR performance.

4.3   Normal Operation

Figure 2 shows the use case diagram for the system, under normal operation.
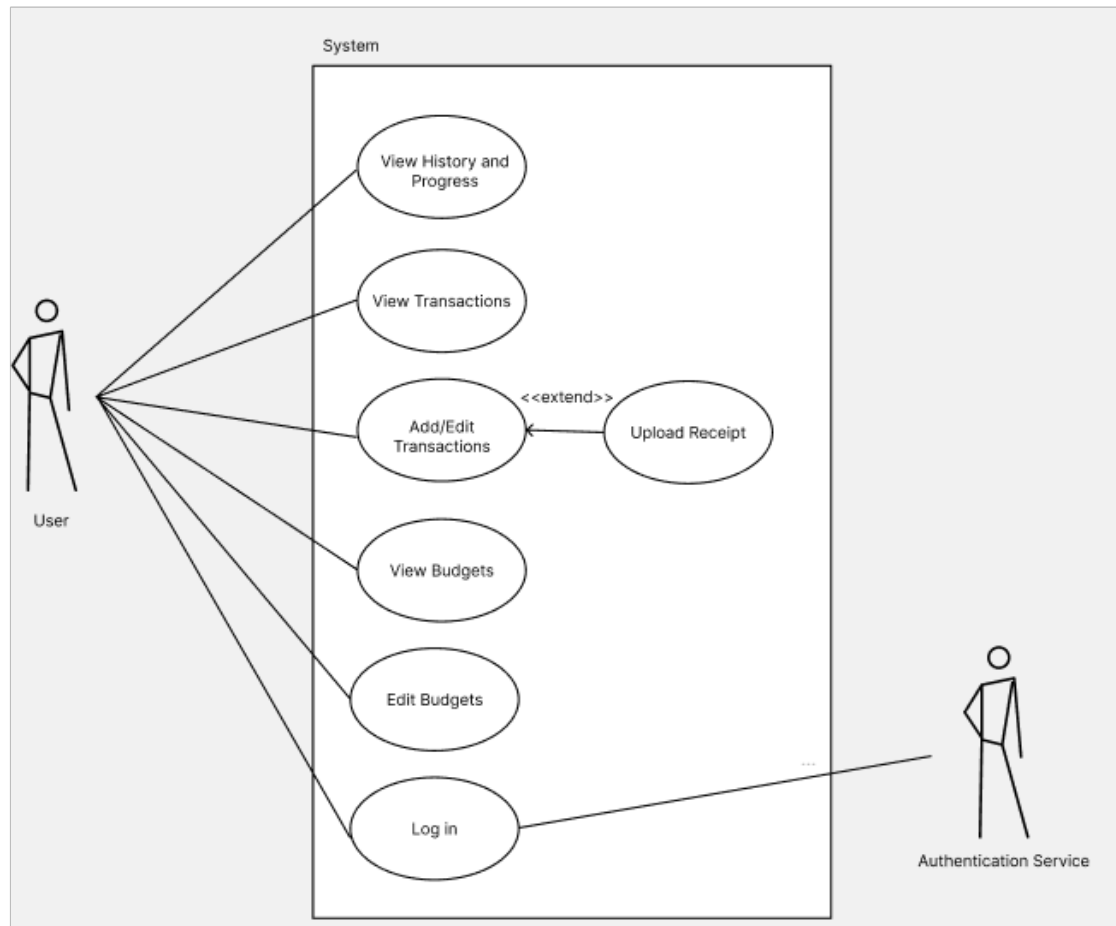
Figure 2: Use Case Diagram

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.

The format of each requirement is as follows: 'Requirement-ID (Priority Level) Requierment'. The priority levels are high (H), medium (M), and low (L), and are used to indicate the importance of the requirement to the system.

5.1   Functional Requirements

User Account Management Requirements

FR-UAM-1 (H) Users must be able to create an account using a name, email address, and password.

Fit criterion: A new user account must be successfully created within **5 seconds** of submitting valid registration details.

FR-UAM-2 (L) Users must be able to update their account information after account creation.

FR-UAM-3 (H) Users must be able to log in using their registered email and password.

Fit Criterion: The system must authenticate valid user credentials and grant access within **2 seconds** 95% of the time.

FR-UAM-4 (H) Users must log in before accessing the application.

FR-UAM-5 (L) Users must be able to log out of their account.

Fit Criterion: The system should successfully log out the user and redirect to Login page within **3 seconds** if a user presses "Log Out" under the profile page.

FR-UAM-6 (L) Users must be able to reset their password if forgotten.

Fit Criterion: A password reset email must be sent within **30 seconds** of a valid request submission.

Receipt Scanning Input Requirements

FR-IP-1 (H) Users must be able to take a picture using the application.

FR-IP-2 (M) Users must be able to upload an image from their device to the application.

FR-IP-3 (L) The system shall allow a limit of *MAX_RECEIPT_FILE_SIZE* per receipt.

Fit Criterion: The system must reject receipt images larger than *MAX_RECEIPT_FILE_SIZE*.

FR-IP-4 (H) The system must display a preview of the uploaded image, allowing users to confirm or retake the image if necessary.

Fit Criterion: The system must display the preview of the uploaded image within **3 seconds** of selection.

FR-IP-5 (H) The system shall upload the confirmed image to the server for processing.

Fit Criterion: The system must successfully upload a confirmed receipt image to the server within **5 seconds** over a standard 4G connection.

Manual Receipt Input Requirements

FR-MIS-1 (M) Users must be able to manually input receipt details. This includes items, their associated costs, and the date of purchase.

FR-MIS-2 (M) The system shall validate the input to ensure required fields are completed.

      Fit Criterion: The system must prevent submission if any required fields (e.g., item name, cost, date) are missing and display a descriptive error message.

Database Management Requirements

FR-DM-1 (H) The system shall store user account information, including usernames, email addresses, and passwords, in a secure database.

FR-DM-2 (H) The system shall store receipt images and extracted data in a secure database.

Item Recognition and Categorization Requirements

FR-RS-1 (H) The system shall identify and display item names and costs from the uploaded receipt image.

      Fit Criterion: The system must correctly extract **at least 90%** of item names and prices from a clear receipt image under standard lighting conditions.

FR-RS-2 (H) The system shall sort the items into predefined categories, and display the category alongside each item.

      Fit Criterion: At least **80% of identified items** must be categorized correctly based on predefined classification rules.

FR-RS-3 (H) The user must be able to modify the name, price, and category of the identified items.

Financial Tracking Requirements

FR-FT-1 (M) The system shall generate an overview of the user's spending history, including total spending by item category and trends over time.

      Fit Criterion: The system must generate a spending summary within **5 seconds** for a user with up to **3 months** of transaction history.

FR-FT-2 (M) Users shall be able to set up budgets for each spending category.

      Fit Criterion: Fit Criterion: Users must be able to set or modify a budget within **5 clicks/taps** from the main financial tracking screen.

FR-FT-3 (M) Users shall be able to view their spending in relation to their set budgets.

FR-FT-4 (L) The system shall notify users when they reach *NOTIFICATION_BUDGET_THRESHOLD*% of their set budget limit.

FR-FT-5 (L) The system shall notify users when they achieve specified savings goals.

FR-FT-6 (L) The system shall store up to 3 years of user transaction history.

5.2   Nonfunctional Requirements

Accuracy Requirements

NFR-ACC-1 (H) The machine learning model must categorize expenses into predefined categories (e.g., groceries, utilities, entertainment) with at least *CATEGORIZATION_ACCURACY*% precision to ensure users' financial data is correctly organized.

NFR-ACC-2 (H) All monetary calculations (e.g., totals, budgets, currency conversions) must maintain a precision of up to *FINANCIAL_PRECISION* decimal places to ensure accurate financial reporting.

NFR-ACC-3 (H) The OCR model must correctly recognize text from images of receipts with a minimum accuracy rate of *OCR_ACCURACY*%, ensuring minimal manual corrections by users.

NFR-ACC-4 (H) The application must guarantee *CALCULATION_PRECISION*% precision in summing up expenses, incomes, and savings across different periods and categories.

NFR-ACC-5 (M) The application should ensure *SYNC_CONSISTENCY*% consistency of financial data upon syncing across multiple devices and sessions.

Performance Requirements

NFR-PERF-1 (M) The application must load user account information and financial metrics within *MAX_DATA_LOAD_TIME* seconds.

NFR-PERF-2 (M) The application must respond to user actions, such as adding a new entry or generating a report, by completing the requested operation and displaying the result within *MAX_INTERACTION_LOAD_TIME* seconds.

NFR-PERF-3 (L) The system must be able to handle a minimum of *MIN_CONCURRENT_USERS* concurrent users without significant performance degradation.

Usability Requirements

NFR-USAB-1 (M) 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 *MAX_COMMON_TASK_CLICKS* clicks.

NFR-USAB-2 (M) New users should be able to complete the account setup and understand core application features within *MAX_ACCOUNT_ONBOARD_TIME* minutes.

NFR-USAB-3 (L) The application shall provide users with an introductory tutorial and tooltips during their first use.

NFR-USAB-4 (M) The application shall display error messages within *MAX_ERROR_MESSAGE_DELAY* second of invalid input submission.

NFR-USAB-5 (M) All error messages will include the error type, the invalid input, and a suggested corrective action.

NFR-USAB-6 (L) Users should be able to recover from errors (e.g., wrong data entry) with no more than *MAX_ERROR_RECOVERY_STEPS* steps.

NFR-USAB-7 (M) Common user tasks, such as adding a new receipt or setting a budget limit, should be completable within *AVG_USER_TASK_TIME* seconds on average, assuming all required information is readily available.

NFR-USAB-8 (M) Information displayed to users shall be concise and relevant, with the main dashboard limited to no more than *MAX_DISPLAYED_KEY_FINANCIAL_POINTS* key financial data points and advanced options accessible within *MAX_ADVANCED_OPTIONS_CLICKS* clicks or less.

NFR-USAB-9 (M) User comprehension of the displayed information must exceed *MIN_USER_COMPREHENSION*%.

Security Requirements

NFR-SEC-1 (H) The system must implement secure password storage and authentication mechanisms.

NFR-SEC-2 (H) The system must ensure that all data transmitted between the client and server is encrypted.

Maintainability Requirements

NFR-MTB-1 (M) 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.

NFR-MTB-2 (M) Dependencies must be regularly updated (i.e., within major patches) to prevent security vulnerabilities and maintain compatibility with new features.

NFR-MTB-3 (H) At least *MIN_CODE_COVERAGE*% 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

NFR-PORT-1 (H) The software shall be compatible with Android and iOS mobile devices running the latest software.

NFR-PORT-2 (H) All data must be stored in platform-independent formats (e.g., JSON, CSV) for easy transfer and compatibility between devices, databases, and systems.

NFR-PORT-3 (M) 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.

NFR-PORT-4 (L) The system must be able to operate in a variety of network environments, including Wi-Fi, cellular networks, and offline mode.

Reusability Requirements

NFR-REUS-1 (M) 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.

NFR-REUS-2 (M) 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.

NFR-REUS-3 (M) 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.

NFR-REUS-4 (L) 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

NFR-UND-1 (H) 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.

NFR-UND-2 (H) 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.

NFR-UND-3 (M) 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.

NFR-UND-4 (M) 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.

NFR-UND-5 (M) 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.

NFR-UND-6 (M) 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.

NFR-UND-7 (M) 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.

NFR-UND-8 (M) 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.

NFR-UND-9 (M) 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

NFR-REG-1 (H) The system must comply with the Canada's Privacy Act to ensure the privacy and security of user data.

NFR-REG-2 (H) The system must comply with Canada's Financial Administration Act to ensure the secure handling of financial information.

Data Retention and Deletion Requirements

NFR-DAT-1 (H) The system must retain user financial data for a maximum of *MAX_DATA_RETENTION* years, in compliance with regulatory requirements, and ensure all expired data is securely deleted within *TIME_DELETE_USER_FINANCIAL_DATA* hours of reaching the retention period, as verified through automated audit logs.

NFR-DAT-2 (H) The system must implement an automated deletion process for inactive accounts, ensuring all associated data is securely deleted after *TIME_USER_INACTIVITY_DELETION* months of inactivity, with notifications sent to users *TIME_NOTIFY_USER_DELETION* days prior to deletion, verified through testing and monitoring.

NFR-DAT-3 (M) The system must allow users to initiate data deletion requests through the application, ensuring that all user-requested deletions are processed and completed within *TIME_DELETE_DATA_REQUEST* days, as confirmed by system logs.

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.

Related requirements: Receipt Scanning Input, Manual Receipt Input, Item Recognition and Categorization, and Financial Tracking requirements.

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.

Related requirements: Maintainability, Reusability, and Understandability requirements.

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.

Related requirements: Receipt Scanning Input, Manual Receipt Input, Security, and Usability requirements.

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.

    - Deployment

        * **Staged Rollouts**: As Apple's App Store has a stricter and lengthier approval process compared to Google Play, the team might consider launching first on one platform (e.g., Android, due to its faster release cycle) to gather user feedback before finalizing the iOS release. Alternatively, if a simultaneous launch is necessary, parallel development with Apple's review constraints in mind will be key.

    - Cloud Cost Considerations:

        1. Firebase Authentication:
            * No cost for MAUs (Monthly active users) less than 50k
            * Follows Google pricing for MAU > 50k (e.g. 50,000-99,999 MAU charges at $0.0055 per MAU)
            * Phone Authentication and MFA Cost: $0.01 per SMS sent
            * Cost assuming 70K MAU and 10K users via SMS login
                · 70K MAUs using email login: (70K-50K) * $0.0055 = $110/month
                · 10K users logging in via SMS: 10K * $0.01 = $100/month
                · Total estimated cost: $210/month
        2. Supabase Database and API Costs

* **Free Tier**:
  · Unlimited API requests
  · 1 GB data transfer
  · 500 MB storage
  · Purpose: Initial development
* **Pro Tier** (Includes everything from free tier):
  · 8 GB database storage
  · 100 GB data transfer
  · Purpose: Post-deployment and maintenance (may change depending on MAUs and DB usage)
* **Variable cost**
  · $0.021 per GB of data transfer above included
  · $0.125 per GB for storage above included
* **Cost analysis for additional storage**
  · 20 GB Storage: (20 GB - 8GB) * $0.125 = $1.50/month
  · 50GB extra transfer: 50 GB * $0.021 = $1.05/month

- **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.

Related requirements: Receipt Scanning Input, Manual Receipt Input, Item Recognition and Categorization, Financial Tracking, Accuracy, Performance, and Usability requirements.

requirements.

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 80% 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 *MAX_RECEIPT_FILE_SIZE* 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.
Related requirements: This would add to the list of functional requirements.

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.
Related requirements: No directly related requirements.

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.
Related requirements: This would add to the list of functional requirements and affect Security (NFR-SEC) requirements.

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.
Related requirements: No directly related requirements.

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

LC6 **Data Privacy Compliance Updates:** Changes in legal requirements may require updates to the application's data handling and privacy policies to ensure compliance.
Related requirements: Regulatory (NFR-REG) and Data Retention and Deletion (NFR-DAT) requirements.

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.
Related requirements: Security (NFR-SEC) requirements.

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.
Related requirements: This would add to the list of functional requirements.

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).
Related requirements: Performance (NFR-PERF) requirements.

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.
Related requirements: No directly related requirements.

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.
Related requirements: Receipt Scanning Input (FR-IP) requirements.

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.
Related requirements: No directly related requirements.

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.
Related requirements: This would involve modifying or deleting existing functional requirements.

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.
Related requirements: No directly related requirements.

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.
Related requirements: User Account Management (FR-UAM), Financial Tracking (FR-FT), and Database Management (FR-DM) requirements.

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.
Related requirements: This would add to the list of functional requirements.

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.

The order of requirements to be phased in will follow Section 8. Project Decomposition and Scheduling in the Development Plan document linked above, with the High priority requirements taking precedence from each category. Only functional requirements are listed here as many non-functional requirements follow from functional requirements. These phases are as follows and are subject to change:

- **Phase 1**: Frontend Requirements (November 1 to December 1, 2024). These requirements will focus on the user interface and user experience of the application, which focuses on the High priority requirements from the User Account Management Requirements, Receipt Scanning Input Requirements, and Manual Receipt Input Requirements subcategories.

- **Phase 1.5**: Model Requirements (October 21 to March 21, 2025). These requirements will focus on the machine learning model that will be used to parse and categorize receipts. This phase will run concurrently with Phase 1 and Phase 2 and will focus on the High priority requirements from the Item Recognition and Categorization Requirements subcategory.

- **Phase 2**: Backend Requirements (January 1 to February 1, 2025). These requirements will focus on the server-side components of the application, which includes the Database Management Requirements and Financial Tracking Requirements subcategories.

- **Phase 3**: Medium Priority Requirements (February 1 to March 31). In this phase, the focus will be on implementing Medium priority requirements from all categories. Low priority requirements will be implemented if time permits.

10   Values of Auxiliary Constants

The following table defines the symbolic parameters that are used throughout the documents in this project.

Table 2: Auxiliary Constants

Symbol	Value	Unit
AVG_USER_TASK_TIME	10	seconds
CALCULATION_PRECISION	99	%
CATEGORIZATION_ACCURACY	80	%
FINANCIAL_PRECISION	2	decimal points
MAX_ACCOUNT_ONBOARD_TIME	5	minutes
MAX_ADVANCED_OPTIONS_CLICKS	3	clicks
MAX_COMMON_TASK_CLICKS	3	clicks
MAX_DATA_LOAD_TIME	5	seconds
MAX_DATA_RETENTION	7	years
MAX_DISPLAYED_KEY_FINANCIAL_POINTS	5	data points
MAX_ERROR_MESSAGE_DELAY	1	second
MAX_ERROR_RECOVERY_STEPS	2	clicks
MAX_INTERACTION_LOAD_TIME	2	seconds
MIN_CODE_COVERAGE	80	%
MIN_CONCURRENT_USERS	10	users
MIN_USER_COMPREHENSION	85	%
NOTIFICATION_BUDGET_THRESHOLD	80	%
OCR_ACCURACY	80	%
SYNC_CONSISTENCY	100	%
TIME_DELETE_DATA_REQUEST	7	days
TIME_DELETE_USER_FINANCIAL_DATA	24	hours
TIME_NOTIFY_USER_DELETION	30	days
TIME_USER_INACTIVITY_DELETION	12	months
MAX_RECEIPT_FILE_SIZE	5	MB

References

Appendix — Reflection

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

1. What went well while writing this deliverable?

   After finishing strong on our problem statement and development plan, our team started to discuss the SRS document and how we'd like to work through it. We decided that we should tackle the beginning couple of sections first because we found those sections to be the most straightforward. The result was we were able to finish the first sections relatively quickly and work on the meat of the document: the requirements.

   Prior to our meeting with Lucas, the team had a chat to discuss possible pain points for the SRS and any uncertainties we had concerning the remaining sections. We still wanted to focus on finishing our requirements as well before the meeting, so we also discussed possible requirements and created a rough draft that we felt confident about so we could ask for feedback on them. After the meeting, we were unblocked and each team member picked up a section to work on in the SRS document.

   Overall the team worked through the deliverable well and any disagreements/concerns were immediately brought up to the team and discussed. The team already had a general direction of the deliverable and documenting them went relatively smoothly.

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

   The main pain point that we came across was the template of the document, as we felt that many sections were not applicable to us and we sometimes had trouble discerning which sections were relevant to our project. The meeting with Lucas clarified that many of the sections probably won't be applicable to us and that helped us be more decisive when deciding which sections we should keep and also which sections we should delve deeper into.

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

   Many of our requirements were thought out before the team started writing this document, however after speaking with potential users and a TA from another course, we added financial tracking and usability into our set of requirements. A potential user brought up that they would like to see metrics relating to their spending and if they're on track with their budgeting goals. Also, a TA from our human-centred interface course mentioned that we should include usability requirements in terms of user interface and user experience.

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

Our human-centred interfaces course (4HC3) will help us with designing interfaces that consider user experience and meet our usability requirements. Also, all group members are currently taking (or have already taken) SFWRENG 4ML3: Introduction to Machine Learning, which will aid us in the development of the ML model.

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.

The team has some introductory knowledge on machine learning models but we will need to research how to train existing models and how to train them efficiently and effectively as well. Additionally, there are certain tools and languages that different members of our group will be using for the first time. It is imperative for each member to familiarize themselves with these resources before and during the implementation stages.

More generally, it would be beneficial for the team to understand how to integrate each member's personal abilities seamlessly to make a cohesive final output. This applies not only to the software aspect of the project, but also to the project development and decision making components. Not only will this benefit each individual's project experience, but it will also allow for more confidence in presenting our final design.

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?

One of the primary approaches to acquiring the technical knowledge mentioned in the previous question is to watch tutorials in order to learn more about machine learning models and other unfamiliar resources. All members of the team have never worked with a machine learning model, so going through existing models and their documentation, as well as some existing projects will strengthen our understanding of this area.

For the non-technical skills, many of these ideas will be put into practice throughout the creation of the project. Regular meetings will be scheduled between individuals of the team with the primary purpose of sharing updates about their work on the project and areas of struggle. The goal is that the team will be able to learn from each other through these meetings so that everybody is on the same page at the end of the project.

All team members will participate in the aforementioned meetings in accordance to their schedules. For learning the technical aspects, team members will decide which process they would like to go through, depending on their preferred method of learning. A few members of the team have stated that they are visual learners and find that looking over previous projects has proven to be extremely helpful for their own

understanding. The rest have chosen to explore the idea of machine learning further through elective courses and readings.