Report

for

RECEIPTLY: RECEIPT MANAGEMENT SYSTEM

Prepared by :
1. Subikshaa Sakthivel (IMT2023020)
2. Vriddhi Agrawal (IMT2023611)
International Institute of Information Technology, Bangalore

Contents

1. Introduction

1.1. Purpose

This Software Requirements Specification (SRS) document provides a detailed description of the functional and non-functional requirements for the web application **Receiptly**. It defines the purpose, scope, features, interfaces, and performance expectations of the system.

Receiptly is designed to simplify expense tracking and financial planning by automating the extraction of itemized data from receipts and bills. Using Optical Character Recognition (OCR) and Natural Language Processing (NLP), the system digitizes receipt content and presents meaningful insights through dashboards and reports.

This document serves as a guide for developers, testers, project managers, and other stakeholders involved in designing, developing, testing, and maintaining the system.

1.2. Document Conventions

The following conventions are used throughout the document:

- **REQ-n:** Each functional requirement has a unique identifier (e.g., REQ-1, REQ-2).
- *Italics:* Used to emphasize system features or important concepts.
- **Priority levels:**
  - High (H): Must be implemented in the first release.
  - Medium (M): Desirable but can be implemented in later versions.
  - Low (L): Optional features for enhancement.

1.3. Intended Audience and Reading Suggestions

This document is intended for:

- **Developers:** For understanding and implementing the technical aspects of each requirement.
- **Testers:** For designing test cases to ensure system reliability.
- **Project Managers:** For tracking progress and ensuring alignment with goals.

- **End Users / Reviewers:** For understanding system capabilities and functionality.

1.4. Product Scope

The project's goal is to build a system that digitizes receipts, extracts structured data, and provides actionable insights into user spending.

Objectives:

- Automate data extraction from uploaded receipts.
- Organize extracted data for visualization and reporting.
- Provide chatbot-based budget assistance.
- Offer monthly spending summaries and forecasts.

Expected Benefits:

- Eliminate manual expense logging.
- Simplify budget management and spending analysis.
- Provide a centralized, easy-to-use dashboard for users.

2. Overall Description

2.1. Product Perspective

Receiptly is a standalone web application that uses a client-server architecture integrating multiple technologies. The system consists of the following modules:

- **Frontend (React.js):** Provides an interactive interface for uploading receipts and viewing dashboards.
- **Backend (Node.js + Express):** Handles API calls, user authentication, and communication between modules.
- **OCR/NLP Engine (Python):** Extracts text from images using Tesseract OCR and processes it with spaCy to classify data.
- **Database (MongoDB):** Stores receipt data, user information, and financial summaries.

System Context:

User → Frontend → Backend (Node.js) → Python OCR/NLP Engine → MongoDB → Dashboard

2.2. System Motivation

In an era of increasing digital transactions, many users still rely on paper receipts for expense tracking. Manual logging is tedious, time-consuming, and error-prone. Receiptly addresses this problem by automating the process and offering intelligent categorization and visualization.

2.3. Product Functions

At a high level, the Receiptly system provides users with a digital platform to record, organize, and analyze their expenses automatically. The major functions include:

- **User Authentication:** Register, log in, and log out securely to access personalized features.
- **Receipt Management:** Upload receipt images, extract information using OCR, and categorize expenses automatically using a machine learning model.

- **Budget Planning:** Define income, savings goals, and category-wise expenditure limits, and view spending distributions through charts.
- **Dashboard and Analytics:** Visualize spending trends, receive insights, and interact with an AI assistant for financial queries.
- **Calendar View:** Access receipts and expense details for specific dates to track daily transactions.
- **Data Storage and Management:** Securely store, update, and retrieve user receipts, budgets, and analytics data.

2.4. System Flow Description

The overall flow of the *Receiptly* system begins when the user launches the application. Upon opening the app, the user is first required to either register for a new account or log in using existing credentials. Successful authentication grants access to the main interface, where the user can navigate to the different functional modules of the system.

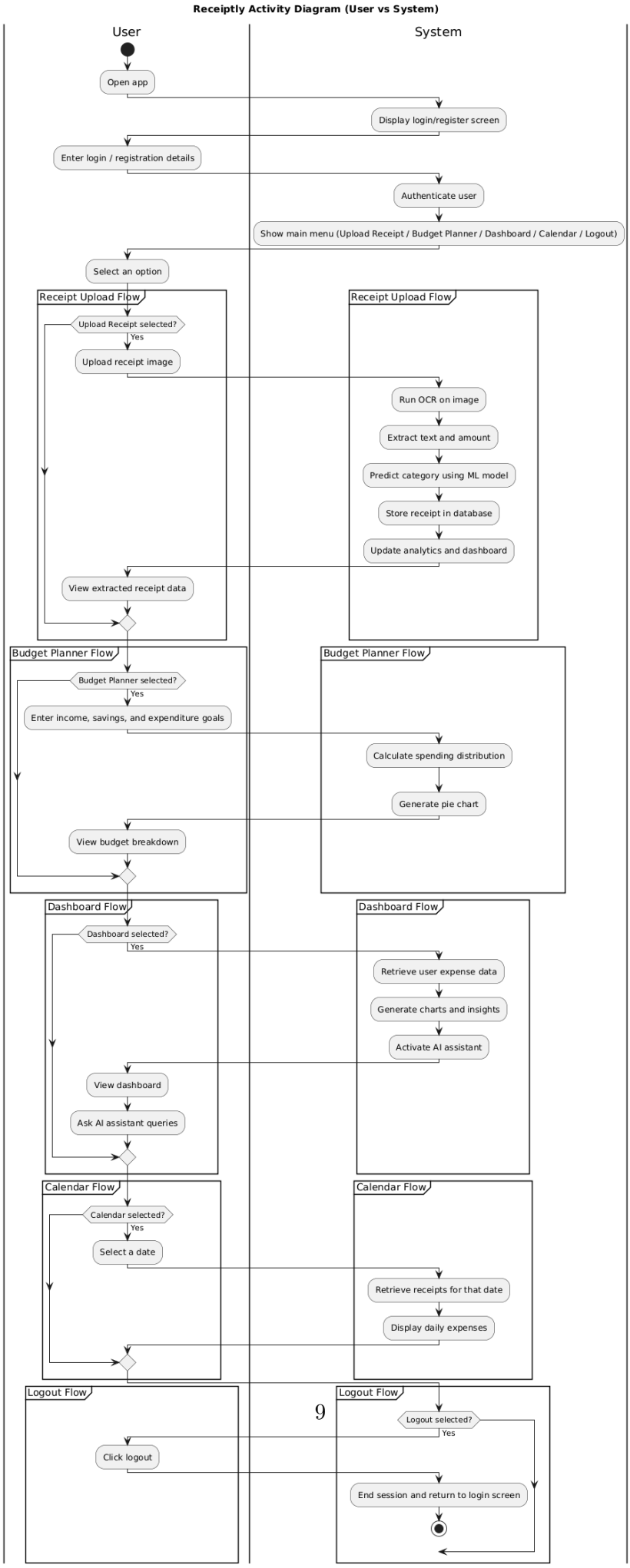
Once logged in, the user is presented with a main menu offering several options: *Upload Receipts*, *Budget Planner*, *Dashboard*, *Calendar*, and *Logout*.

- **Receipt Upload** When the user selects the *Upload Receipts* option, they can upload an image of a physical receipt. The system performs Optical Character Recognition (OCR) to extract relevant information such as merchant name, date, and total amount. The extracted data is then processed by a machine learning model to automatically classify the expense into predefined categories. The processed receipt is stored in the database, and the updated expense analytics are displayed on the user's dashboard.
- **Budget Planner** In the *Budget Planner* section, the user can enter their income, desired savings, and category-wise spending limits. The system calculates the distribution of expenses and presents a pie chart showing the proportion of spending across categories. This visualization helps users understand how their planned expenditure aligns with their financial goals.
- **Dashboard** The *Dashboard* provides an overview of the user's financial activity through charts, graphs, and summaries representing spending trends over time. An integrated AI assistant allows users to ask natural language questions about their expenses, budgets, or spending patterns and receive intelligent insights and responses.
- **Calendar View** The *Calendar* view enables users to select a specific date and view the receipts and expenses recorded on that day. This feature supports tracking daily transactions and identifying spending patterns for particular days.

- **Logout** At any point, the user may choose to log out. The system terminates the session, ensures that user data remains secure, and redirects the user back to the login or registration screen.

2.5. Activity Diagram

The following activity diagram shows a sample flow of the above functionalities.


Receiptly Activity Diagram (User vs System)
User
System
Open app
Display login/register screen
Enter login / registration details
Authenticate user
Show main menu (Upload Receipt / Budget Planner / Dashboard / Calendar / Logout)
Select an option
Receipt Upload Flow
Upload Receipt selected?
Yes
Upload receipt image
Run OCR on image
Extract text and amount
Predict category using ML model
Store receipt in database
Update analytics and dashboard
View extracted receipt data
Budget Planner Flow
Budget Planner selected?
Yes
Enter income, savings, and expenditure goals
Calculate spending distribution
Generate pie chart
View budget breakdown
Dashboard Flow
Dashboard selected?
Yes
Retrieve user expense data
Generate charts and insights
Activate AI assistant
View dashboard
Ask AI assistant queries
Calendar Flow
Calendar selected?
Yes
Select a date
Retrieve receipts for that date
Display daily expenses
Logout Flow
Logout selected?
Yes
Click logout
End session and return to login screen

2.6. User Characteristics

User can upload receipts, view budgets, and analyze monthly expenses. Minimal technical skills required.

2.7. Operating Environment

- Frontend: React.js, compatible with Chrome, Firefox, and Edge.
- Backend: Node.js 18+, Express.js.
- OCR/NLP Engine: Python 3.11+, Tesseract OCR, spaCy.
- Database: MongoDB 6.0 or higher.

2.8. Design and Implementation Constraints

- Tesseract OCR must be installed locally and accessible.
- MongoDB must be running prior to backend initialization.
- Python and Node.js environments must be configured.
- REST API used for backend–Python communication.
- The accuracy of extraction depends on receipt image quality.

2.9. Assumptions and Dependencies

- Users will upload valid image files (JPEG, PNG, or PDF).
- Internet connectivity is required for chatbot functionality.
- External APIs (Google Generative AI) will remain stable.

3. External Interface Requirements

3.1. User Interfaces

The *Receiptly* system provides a clean, intuitive, and responsive graphical user interface (GUI) designed for ease of use across both web and mobile platforms. The interface enables users to register, log in, upload receipts, view analytics, plan budgets, and interact with the AI assistant.

Standard navigation buttons such as *Dashboard*, *Upload*, *Budget Planner*, *Calendar*, and *Logout* appear on every page. Error and success messages follow a uniform color scheme and notification style.

3.2. Hardware Interfaces

Receiptly is a web-based application and requires no specialized hardware interfaces. It operates on any standard computing device capable of running a modern web browser, such as:

- Desktop or laptop computers (Windows, macOS, Linux)
- Smartphones and tablets (Android, iOS)

The system interacts only with device-level components for:

- **File Uploads:** Access to the device's file system for selecting receipt images.
- **Display:** Rendering visual charts and pages through the browser's graphical interface.

No additional hardware drivers or peripherals are required.

3.3. Software Interfaces

The system is designed with a modular architecture and interacts with several software components:

- **Frontend:** Built using React.js, communicates with the backend via RESTful APIs.
- **Backend:** Implemented in Node.js and Express.js, providing endpoints for authentication, OCR processing, and data analytics.

- **Machine Learning Module:** A Python-based OCR and categorization pipeline using `pytesseract` and a pre-trained model for classifying expenses.
- **Database:** MongoDB is used for storing user profiles, receipts, budgets, and analytics data.
- **Middleware:** Includes authentication middleware (JWT-based) and error-handling modules.
- **External APIs:** Integration with OCR libraries and AI components (Gemini API) for intelligent query responses.

Data exchanged between the frontend and backend uses JSON format over HTTPS for secure communication.

3.4. Communications Interfaces

The *Receiptly* system relies on standard web communication protocols to ensure interoperability and data security.

- **Protocol:** All client-server communication occurs over HTTPS using RESTful APIs.
- **Data Format:** Uses JSON for data requests and responses.
- **Encryption:** Sensitive data such as passwords are encrypted before transmission and storage using bcrypt.
- **Network Requirements:** Requires stable internet connectivity for API communication, data retrieval, and model inference.

4. System Features

4.1. Receipt Upload and OCR Processing

Description: Users can upload receipt images. The backend sends them to the OCR engine for text extraction.
Priority: High **Functional Requirements:**

- **REQ-1:** Accept receipt images in JPG, PNG, or PDF format.
- **REQ-2:** Extract text using Tesseract OCR with confidence scoring.
- **REQ-3:** Display extracted items to the user for confirmation.

4.2. NLP-Based Data Classification

Description: The NLP module classifies extracted text into structured fields such as item name, quantity, and category.
Priority: High **Functional Requirements:**

- **REQ-4:** Identify item names, quantities, and prices using NLP.
- **REQ-5:** Categorize items (e.g., Groceries, Utilities, Dining).

4.3. Dashboard and Visualization

Description: Provides users with interactive charts summarizing spending trends.
Priority: High **Functional Requirements:**

- **REQ-6:** Display monthly spending in bar and pie charts.
- **REQ-7:** Show expenditure by category and date.

4.4. Chatbot and Budget Planner

Description: Provides conversational insights and budget tracking suggestions.
Priority: Medium **Functional Requirements:**

- **REQ-8:** Support natural language queries.
- **REQ-9:** Allow users to specify budget goals.

4.5. Class Diagram

The class diagram represents the object-oriented structure of the system, depicting classes such as User, Receipt, Item, and their relationships.

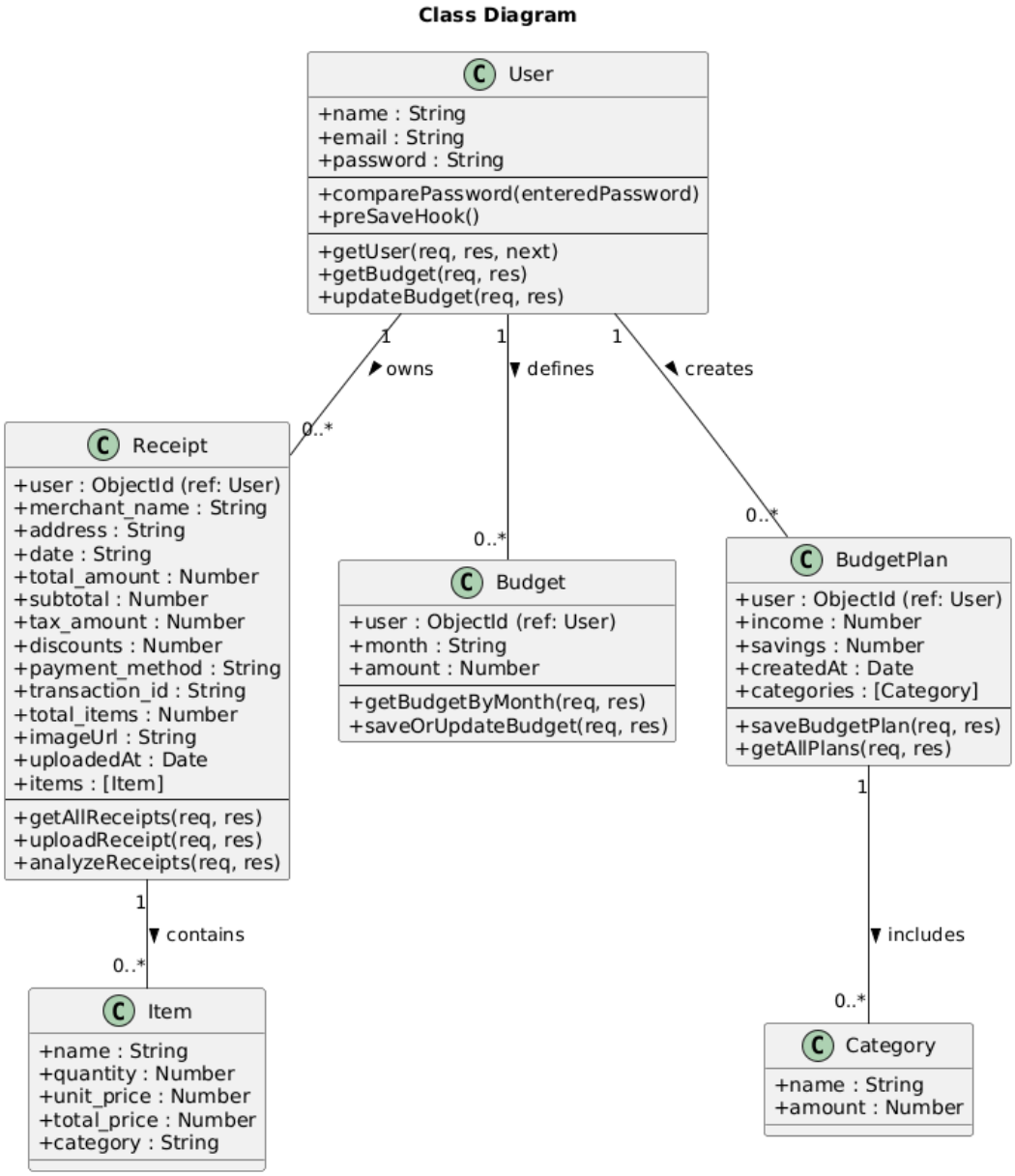


Figure 4.1.: Class Diagram for Receiptly

4.6. Sequence Diagram

The following sequence diagram represents the step-by-step interaction between the user and the system across the major functionalities of the Receiptly application.

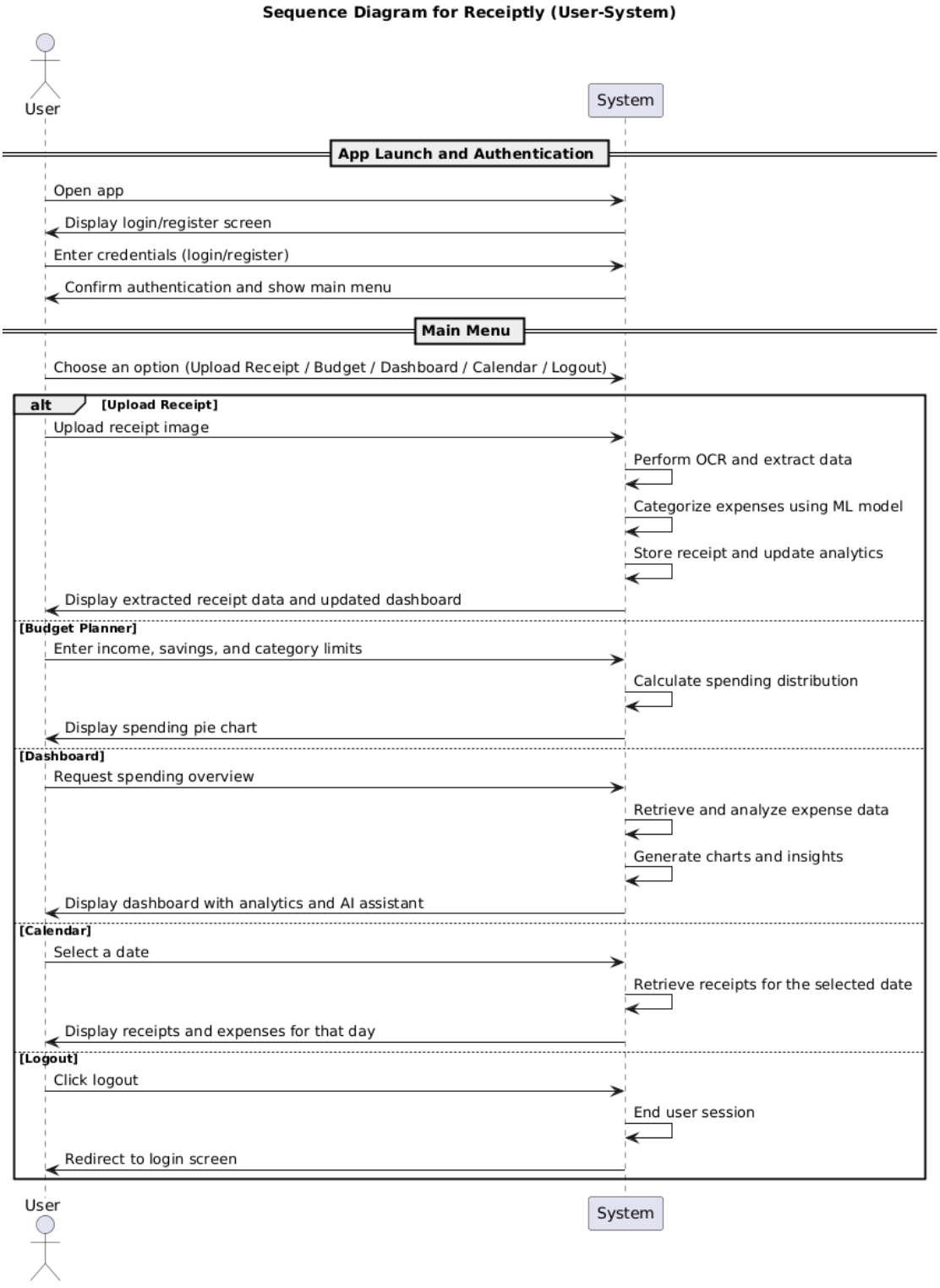


Figure 4.2.: Sequence Diagram for Receiptly

5. Nonfunctional Requirements

5.1. Performance Requirements

- OCR + NLP combined processing time less than 10 seconds.
- Dashboard queries load within 1 second.
- System must handle concurrent users.

5.2. Security Requirements

- User passwords are hashed and stored securely using **bcrypt**.
- Sensitive data (like DB credentials) stored in environment variables.
- All communication between frontend and backend via HTTPS using JWT tokens.

5.3. Software Quality Attributes

- **Usability:** Clean and intuitive interface.
- **Reliability:** Error handling for invalid uploads.
- **Maintainability:** Modular code and documentation.
- **Portability:** Cross-platform support for major OS and browsers.

5.4. Additional Features

- Authenticated users only can view personal expense data.
- Budgets can be reset on the first of every month.

6. Future Improvements

- Future integration with cloud OCR for better recognition.
- Optional dark mode for better accessibility.
- AI-based spending prediction and savings tips.

A. Appendices

A.1. Glossary

Term	Definition
OCR	Optical Character Recognition – Converts images to text.
NLP	Natural Language Processing – Extracts structured data from text.
MongoDB	NoSQL database for storing extracted data.
bcrypt	A hashing algorithm used for secure password storage.
spaCy	Python NLP framework for tokenization and classification.

A.2. Entity-Relationship Diagram

The following ER diagram represents the database design of the Receiptly system, highlighting key entities and their relationships.

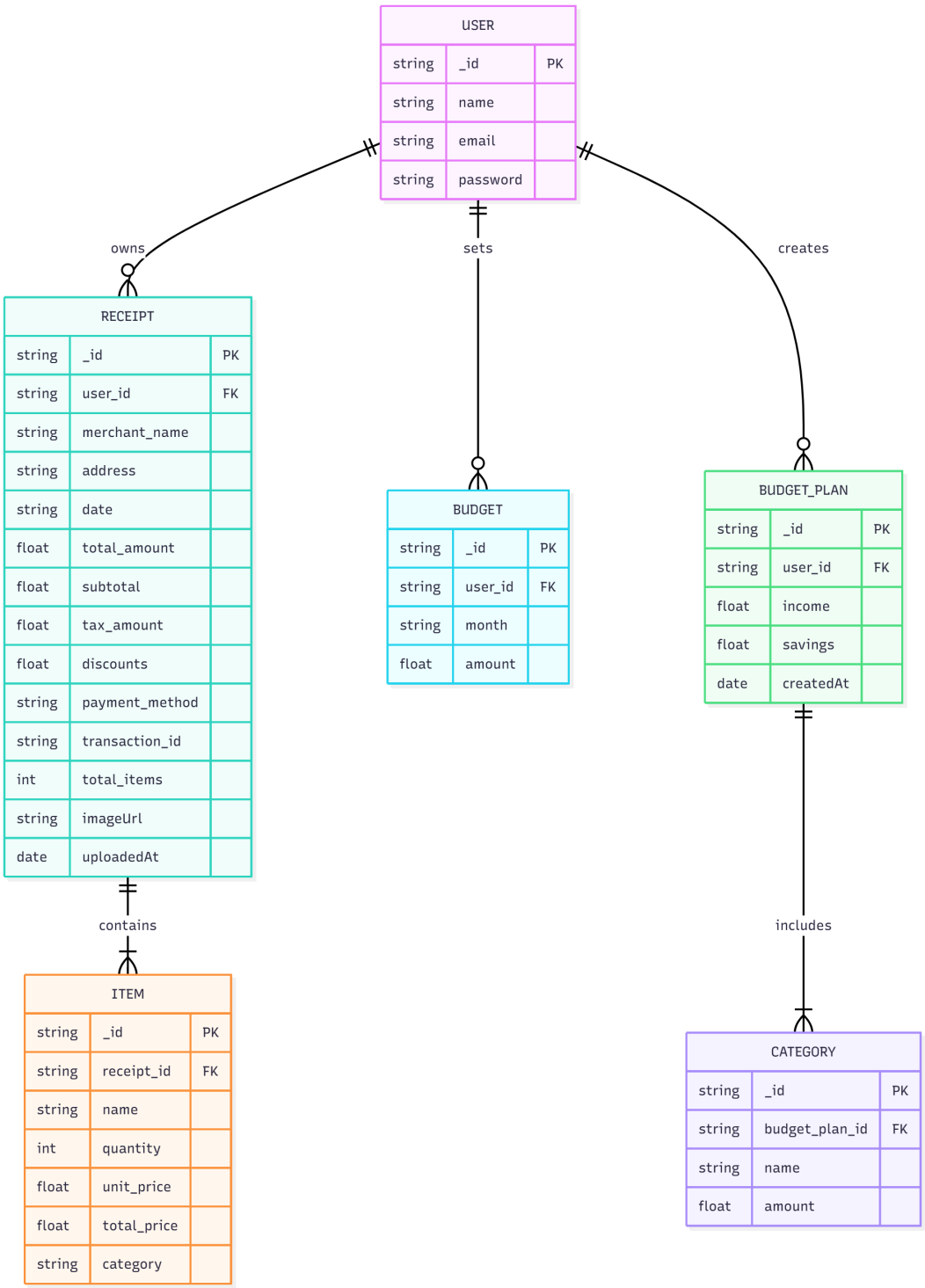


Figure A.1.: ER Diagram for Receiptly