Requirements Document

PiggyBank

Designer Organization

January 27th 2023

Table of Contents

Revision History

Name	Date	Reason for	Version

		Change	
Connor Newbery	January 30th	Added sections 2.3,2.4,2.5,2.6	1.0
Charlie Wager	February 1st	Added section 1.1, 1.2, 1.5	1.1
Whole Team	February 2nd	Added section 2.1, 3.1, 3.2, 3.3, 3.4, 3.5, 4.1	1.2
Connor Newbery	February 7th	Added Sections 4.2, 4.3, 4.4	1.3
Connor Newbery	February 9th	Added Sections 5.1,5.3,5.4	1.4
Charlie Wager	February 9th	Added section 2.2	1.5
Whole Team	February 10th	Added Section 5.3 adaptability and finalized doc	1.6

1 Introduction

1.1 Purpose

PiggyBank is an intuitive software for financial management and budgeting that addresses the needs of users with limited financial experience. Our solution helps users budget and manage their finances with ease by providing guidance and simplifying complex financial concepts. With PiggyBank, users no longer have to struggle with managing their finances due to a lack of knowledge and support.

1.2 Project Scope

The scope of this project includes helping to relieve some of the stress experienced by users with low financial literacy when budgeting and managing their finances. Creating an easy to use interface that provides tips and guidance to the user will help eliminate some of the stress and confusion associated with this process. Users banking and personal data will be kept secure and confidential.

1.3 Glossary of Terms

Term	**Definition**
Personal Account	A PiggyBank account created for an individual user of the application. One personal account may have multiple bank accounts associated with it.
Bank Account	A savings or chequing account associated with a particular user account.
Credit Account	An account for a credit card associated with a user’s personal account
Transaction	Any individual transaction in any one of a user’s bank or credit accounts. For example, a payroll payment or a credit card transaction.
Third party service / application	Will be required for payment gateways and loan management systems. Can be extended to any other service that PiggyBank chooses to use.

1.4 Overview

First, the Requirements Document provides an overall description of the PiggyBank software system containing a Product Perspective, Product Features, User Classes and Characteristics, Operating Environment, Design and Implementation Constraints, and Assumptions and Dependencies. Next Requirements Document covers the systems features and their functional requirements. These features include [list of features]. Requirements Document then describes the External System Requirements which define requirements for the User Interface, Hardware Interface, Software Interfaces, and Communication Interfaces. Finally, the Requirements Document defines Non-Functional Requirements such as Performance, Safety, and Security Requirements as well as Software Quality Attributes.

2 Overall Description

2.1 Product Perspective

PiggyBank is a software banking application for students. The app allows users to create a savings account with no fees, and to track their spending and earnings. The app also has a budgeting feature that allows users to set spending limits for themselves and to see where their money is going. PiggyBank is intended to help students manage their finances and to make saving money easier. It is free for all users.

PiggyBank is an application built on NextJS, MUI, and uses a MySQL database. It is a simple web application that allows users to save money by setting up a savings goal and then making regular deposits into their PiggyBank account. The app keeps track of the user's progress towards their savings goal and allows them to see how much they have saved over time. PiggyBank is a great way to save money and reach your financial goals!

The PiggyBank web application is designed to help users manage their finances and save money. The app has the following design and implementation constraints:

- The app must be able to track income and expenses
- The app must be able to categorize expenses
- The app must be able to set goals and track progress
- The app must be secure and private
- The app must be easy to use

Assuming that the user has an internet connection and a web browser, the PiggyBank web application will function as intended. The user will need to create an account and deposit money into that account in order to use the PiggyBank features. The user will be able to view their account balance and transactions, as well as create and manage savings goals. In order to access the PiggyBank web application, the user will need to enter their username and password.

PiggyBank aims to be a self-adaptive software that customizes what the user sees based on how they use the application and what their saving goals are. It will allow users to track their spending each month, and track their progress in saving up for a specific purchase.

2.2 Product Features

PiggyBank facilitates easier financial management and clearer budgeting for users with limited financial knowledge. The PiggyBank application will allow authenticated users to create bank accounts, manage these bank accounts, access budgeting tools, and access tips and tutorials about budgeting and financial management.

2.3 User Classes and Characteristics

This banking system is designed for students. They are our only user type, although others such as young people who are not going to school may also choose to use our banking system.

2.3.1 Students

As students will be the only targeted users of this platform, they will have access to all of the system features. These features will be designed for ease of usability by the average student with no technical knowledge.

2.4 Operating Environment

The platforms the system will have to be able to operate on includes:

- Desktops and laptops: running popular operating systems such as Windows, macOS, and Linux.
- Tablets: including popular models from Apple, Microsoft, and Android manufacturers.
- Smartphones: with support for both iOS and Android operating systems.
- Web browsers: through a responsive and optimized web-based interface.

2.5 Design and Implementation Constraints

Security

Ensuring that sensitive financial information is protected and complying with relevant data privacy regulations.

Integration With Existing Systems

Ensuring seamless integration with banks, financial institutions, and other relevant systems for transactions and data syncing.

Accessibility

Ensuring accessibility for users with different abilities and disabilities, such as vision or hearing impairments.

Scalability

The software should be designed to handle increased user numbers and transactions, while maintaining performance and stability.

2.6 Assumptions and dependencies

2.6.1 Assumptions

Development Environment

Assuming the availability of necessary hardware and software resources, including development tools and platforms, for the development and testing of the software.

User Acceptance

Assuming that the user requirements have been accurately captured and that the software will meet the needs and expectations of the target user group, students..

Budget and Time Constraints

Assuming the availability of sufficient budget and development time to implement all the required features and meet project deadlines.

Data Privacy and Security

Assuming the ability to comply with relevant data privacy and security regulations, such as GDPR or HIPAA, and to protect sensitive financial information.

Technical Limitations

Assuming the compatibility of the software with commonly used hardware platforms and operating systems, as well as potential future compatibility issues.

2.6.2 Dependencies

Third-party Components

Third-party components will be used throughout the system, and this project will depend on the availability and compatibility of these software components that will be integrated into the student banking software. These third-party components will be responsible for tasks such as payment gateways or loan management systems.

Integration with External Systems:

The application will depend on the ability to integrate the student banking software with existing bank and financial systems, and that the necessary data and transactions can be transferred and synced smoothly.

3 System Features

3.1 User Authentication and Account Creation

3.1.1 Description and Priority

This feature must provide secure and convenient access to the PiggyBank software for users, allowing them to easily create and manage their personal accounts. The user authentication and account creation feature is a crucial component of the PiggyBank software and has been designated as **high priority**. Ensuring the security and accessibility of user financial information is paramount to the success of the PiggyBank solution.

3.1.2 Functional Requirements

REQ-1: Account Creation: Users can create a new PiggyBank personal account by providing their personal information such as a valid name, email, password, and any other required information. Account creation should be possible entirely online.

REQ-2: User Login: Users can log in to their existing PiggyBank personal account using their email and password.

REQ-3: Password Security: PiggyBank implements secure password storage and encryption to ensure user data is protected.

REQ-4: Email Verification: After account creation, users will receive an email verification link to confirm their email address.

REQ-5: Forgotten Password Recovery: Users can reset their password if they have forgotten it by providing their registered email.

3.2 Bank Account Management

3.2.1 Description and Priority

Our online banking platform allows users to easily open a new bank account after completing the sign-up process. Users can choose from two account types: checking, savings and provide the necessary information to open the bank account. The system will then verify the information and create the new bank account, allowing users to start making transactions and managing their finances with ease. This feature has been designed as a **high priority** feature.

3.2.2 Functional Requirements

REQ-1: An authenticated user must be able to view all bank accounts linked with the user account.

REQ-2: An authenticated user must be able to send money between their accounts.

REQ-3: An authenticated user must be able to send and receive money from outside banking establishments.

REQ-4: An authenticated user must be able to create a new bank account.

REQ-5: An authenticated user must be able to disable any of their bank accounts.

3.3 Budgeting

3.3.1 Description and Priority

This feature allows users to create and manage a personalized budget plan. The budgeting system will allow users to set income and expenses, track their spending, and receive alerts when they are approaching or exceeding their budget limits. Budgeting is a critical aspect of financial management and is essential for users to stay on track with their financial goals. This feature is important for the success of the PiggyBank software and will be considered a **medium priority**.

3.3.2 Functional Requirements

REQ-1: Income tracking: Users will be able to enter and track their income from all sources, including salaries, investments, and other sources of revenue.

REQ-2: Expense tracking: Users will be able to enter and track their expenses, categorizing them by type (e.g. housing, transportation, entertainment, etc.).

REQ-3: Budget creation: Users will be able to create a monthly budget that will allocate funds to each expense category.

REQ-4: Budget monitoring: Users will be able to track their progress on their budget throughout the month in each category of spending. They will be able to view and make edits to the budget throughout the month.

3.4 Tips and Tutorials

3.4.1 Description and Priority

This feature will assist users with creating budgeting plans. This feature will provide multiple tutorials that explain various concepts around proper budgeting. It will also provide tips to users while they create their budget plans through a small window in the corner of the screen. This feature is considered a **medium priority** feature and is needed as many users are unsure of the best ways to budget.

3.4.2 Functional Requirements

REQ-1: Tutorials List: Users will be able view a list of all the tutorials available to them.

REQ-2: Tutorial Selection: Users will be able to select and complete the desired tutorial.

REQ-3: Tutorial Progress: Users should be able to save and see their progress on uncompleted tutorials.

REQ-4: Tips Display: Appropriate and relevant tips will be shown to the user when the system decides a tip is needed.

REQ-5: Tips Language: Tips should be concise, helpful, and very easy to understand.

4 External Interface Requirements

4.1 User Interfaces

The system will take user input from their mouse, as this is a web application. The login screen will prompt the user to log in with their username and password, or sign up for an account if none has been created. Once a user is signed in, they will see their home screen, which is the main way they will navigate to the other components. One component will be the accounts view. The user will be able to see an overview of all their bank accounts and credit cards from the home screen, and clicking on each one of these accounts will take you to a full page summary of the account that displays all of the transactions made on this account. Another component will be budgeting. If the user uses our budgeting features, they will have a summary of their progress of their monthly budgeting shown on the home screen. Clicking on this summary will navigate them to the budgeting section, where they can view and edit their monthly budget. They will also have a transfer section from their home screen. Clicking on this will allow users to choose between sending transfers between their accounts, and send Interac e-transfers.

4.2 Hardware Interfaces

PiggyBank is a web application designed to be accessed from a device with an internet connection and a web browser. The supported device types include desktops, laptops, smartphones and tablets. There will be no required connection or communication with any hardware outside of the device being used to access the web application. The physical interface involves the communication protocols used to transmit the data between the software and the hardware components. This could involve the use of secure protocols such as HTTPS to ensure the privacy and security of the user's data. The software also uses session management to track the user's progress and to ensure that their session data is saved even if they log out or switch devices.

4.3 Software Interfaces

PiggyBank is built on NextJS, MUI and uses a MySQL database as its primary storage for user information. The app is designed to run on a web server that is compatible with the operating system of the user's device. Incoming data into the system includes user transactions, savings goals, and budget information. This data is processed and stored in the MySQL database. The data is then displayed back to the user. Outgoing data from the system includes updated account balances and transaction history, as well as the user's progress towards their savings goals. This information is displayed to the user. The communication between the software components is done through APIs, which allow the components to exchange data and messages. In terms of services, the app requires a web server to run, as well as a database to store user information. The app communicates with the database to store and retrieve user information and with the web server to serve the web pages and display the user interface.

4.4 Communications Interfaces

PiggyBank requires certain communication functions in order to operate effectively and provide a seamless experience to the user. These include:

- Web browser communication: PiggyBank is a web-based application that requires an internet connection and a compatible web browser to access and use the app. The user must have a web browser that supports HTML, CSS, and JavaScript to use the app effectively.
- Network server communication protocols: The app requires secure communication protocols, such as HTTPS, to transmit data between the software and hardware components, ensuring the privacy and security of the user's information.
- Email communication: PiggyBank may require email communication for password reset requests or account verification processes. The app may use Simple Mail Transfer Protocol (SMTP) to send emails to the user.
- Electronic forms: The app uses electronic forms for user input, such as creating a new account or adding transactions. These forms are processed and stored in the MySQL database.

5 Other Non Functional Requirements

5.1 Performance Requirements

Load time

The application should load in under 3 seconds on a reliable internet connection. This is important because users expect fast and responsive software, and slow load times can lead to frustration and decreased usage.

Budgeting feature performance

The budgeting feature should process financial data and update the user's budget in real-time, with a maximum delay of 1 second. This is important because users need to have an up-to-date view of their budget to effectively manage their finances.

Concurrent user performance

The application should be able to handle at least 1000 concurrent users without any noticeable lag or slowdowns. This is important because PiggyBank is intended for a large user base, and performance degradation with a high number of users would negatively impact the user experience.

Mobile device performance

The application should load in under 5 seconds on a 3G network and in under 3 seconds on a 4G network on mobile devices. This is important because users are increasingly using mobile devices to manage their finances and expect a seamless experience.

5.2 Security and Safety Requirements

Data encryption

All financial and personal information stored by the application should be encrypted using industry-standard encryption algorithms to prevent unauthorized access and ensure privacy.

User authentication

The application should have a secure user authentication mechanism, such as password or biometric authentication, to ensure that only authorized users can access their accounts.

Data access control

The application should have proper access control mechanisms in place to ensure that user data can only be accessed by authorized personnel, such as customer support representatives.

Regular security updates

The application should receive regular security updates and patches to fix any potential vulnerabilities and ensure the ongoing security of user data.

Compliance with regulations

The application should comply with relevant security and privacy regulations, such as the EU's General Data Protection Regulation (GDPR) and the Payment Card Industry Data Security Standard (PCI DSS).

Security certifications

The application should meet relevant security certifications, such as ISO 27001, to demonstrate its commitment to security and privacy.

5.3 Software Quality Attributes

Adaptability

For the application, self-adaptation will be integrated into the front end UI and API. The adaptability of the front end can include; the adjustment of different sections being displayed at the top of the main page. If a user transfers money to a particular recipient, the software will suggest that recipient first when they initiate a transfer in the future. Similarly for the API, depending on the users spending habits and income. The software can assist in creating a savings plan or depositing into a “Piggy Bank” with extra funds from rounded up transactions. By offering automatic adaptation, a university student banking application provides students with a streamlined and effortless way to manage their finances, allowing them to focus on their academic and personal pursuits.

Robustness

The application should be able to handle unexpected user inputs and errors without crashing or losing data. This will ensure that users can trust the application to store their financial information securely and accurately.

Maintainability

The application should be easily maintainable, with clear and organized code that can be easily modified or updated as needed. This will make it easier for the developers to fix bugs and add new features in the future.

Flexibility

The application should be flexible enough to accommodate the changing financial needs of its users, with the ability to add new features and integrations as needed.

Correctness

The application should accurately reflect users' spending, earnings, and savings, with no discrepancies or errors.