

Software Design Document

SweetLedger



SWEETLEDGER

Version 1.0

SweetLedger Team:

Devin Perry, Matthew Crowley, Kendal Elison, Kahmin Keller, Connor Oberlin

Client: Dr. Ermias Mamo

Kennesaw State University

TABLE OF CONTENTS

1. Introduction	
1.1. Purpose	3
1.2. Scope	3
1.3. Overview	3
1.4. Reference Material	3
1.5. Definitions and Acronyms	4
2. System Overview	4
3. System Architecture	
3.1. Architectural Design	4
3.2. Decomposition Description	6
3.3. Design Rationale	6
4. Data Design	
4.1. Data Description	7
4.2. Data Dictionary	7
5. Component Design	8
6. Human Interface Design	
6.1. Overview of User Interface	9
6.2. Screen Images	9
6.3. Screen Objects and Actions	11
7. Requirements Matrix	11

1. INTRODUCTION

1.1 Purpose

This software design document describes the architecture and system design for SweetLedger, a web-based accounting application. SweetLedger is designed to help businesses manage financial transactions, generate reports, and maintain accurate accounting records. This document is intended for project managers, the development team, and other stakeholders involved in the implementation of the system.

1.2 Scope

This document details the implementation of the SweetLedger Web Application. The application is a secure, role-based system hosted on AWS Elastic Beanstalk and uses MongoDB for data persistence. SweetLedger will consist of major functional including: Authentication/Security, Chart of Accounts Management, Transaction Journalizing, Transaction Posting/Review, Financial Reporting, and Ratio Analysis. The design will ensure the application is accessible from various devices and is fully functional across popular web browsers.

1.3 Overview

This document serves as the formal technical specification for the SweetLedger application. It is organized to lead the reader through the system's design. Section 2 (System Overview) outlines the major project requirements, user roles (Administrator, Manager, Accountant), and core accounting capabilities. Section 3 (System Architecture) defines the three-tier design. Section 4 (Data Design) specifies the document structure and constraints for accounting data within MongoDB. Section 5 (Component Design) details the functionality and interfaces of the C# backend and JavaScript frontend modules. Section 6 (Human Interface Design) covers the design and usability for the web interface across various devices. Finally, Section 7 (Requirements Traceability) confirms that all original project requirements are addressed by the design.

1.4 Reference Material

"Getting Started." *React.dev*, 17 Jul. 2019, react.dev/learn/getting-started.

MongoDB. "Getting Started." *MongoDB Documentation*, MongoDB, 1 Oct. 2025, www.mongodb.com/docs/get-started/. Accessed 1 Oct. 2025.

1.5 Definitions and Acronyms

Acronym	Meaning
COA	Chart of Accounts
API	Application Programming Interface

2. SYSTEM OVERVIEW

SweetLedger is designed as a user-friendly web application for managing accounting tasks. Users authenticate via secure login to access functionality such as creating a chart of accounts, journalizing transactions, attaching source documents, approving transactions, viewing financial reports, and performing ratio analysis. The frontend is built using React (JavaScript, HTML, CSS) for responsive, interactive UIs. The backend logic, workflows, and APIs are implemented in C#. MongoDB stores structured data (users, accounts, transactions, logs). The system is deployed on AWS Elastic Beanstalk. Built-in help and a table-of-contents interface assist users in navigation. Administrators, managers, and regular users see different views and permissions.

3. SYSTEM ARCHITECTURE

3.1 Architectural Design

To support the system's functionality, responsibilities were divided into high-level subsystems that interact seamlessly to provide a secure and user-friendly interface. The User class represents individuals who interact with the system and contains attributes such as personal information, login credentials, and account status. Each user is associated with a Role, which determines their permissions and accessible features (Administrator, Manager, or Regular User). The PasswordPolicy class enforces authentication rules, ensuring that all credentials meet security standards before being accepted into the system. The Document entity represents files uploaded or downloaded by users, storing metadata such as filename, type, size, owner, and access permissions.

Subsystem responsibilities are divided as follows:

- Login Module: Validates User credentials against stored data, applies PasswordPolicy rules, and grants role-based access.
- Create User Module: Allows administrators to register new users, enforce password policies, and assign appropriate roles.

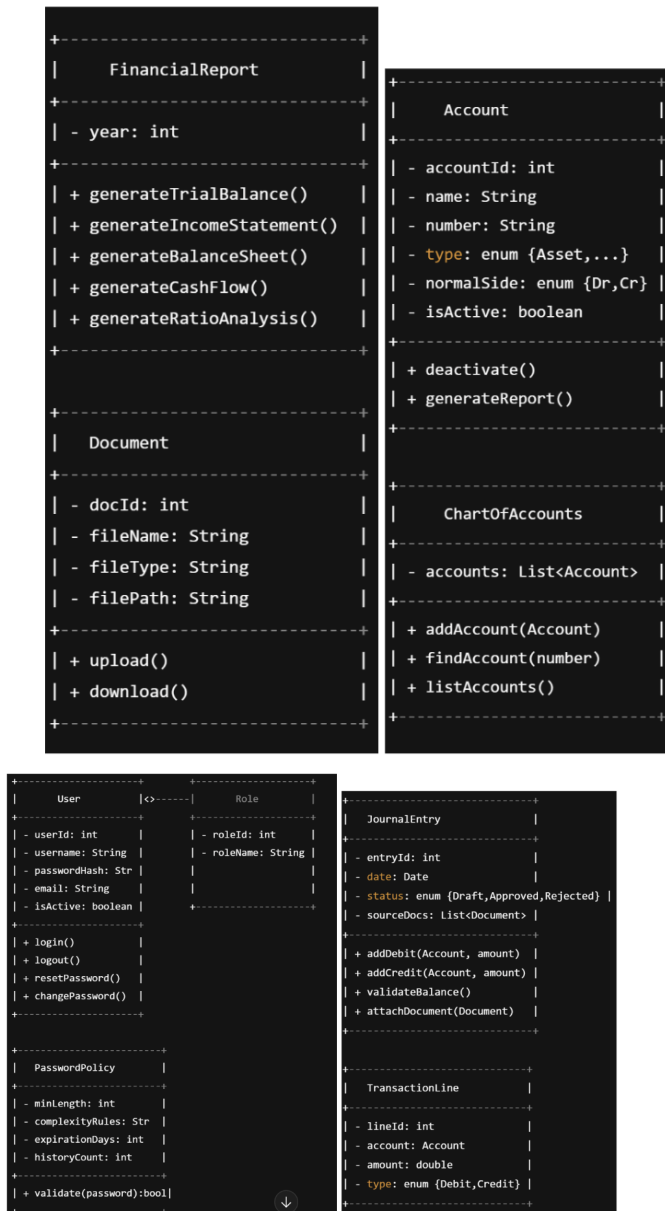
- Forgot Password Module: Uses PasswordPolicy to reset and validate new credentials securely.
- User Management Module: Enables administrators to view, suspend, activate, or update users, and generate reports on system users.
- Dashboard Module: Displays role-based navigation and personalized information (username, profile picture).
- Document Management Module: Provides users with the ability to upload and download documents. Documents are tied to User accounts, and access permissions are enforced based on Role.

These subsystems communicate across three layers:

1. User Interface Layer: Built with React, HTML, CSS, and JavaScript, presenting interactive login forms, dashboards, and document upload/download interfaces.
2. Application Layer: Implemented with C# APIs on AWS Elastic Beanstalk. This layer manages logic for user authentication, role enforcement, password validation, and document processing.
3. Data Layer: MongoDB stores User profiles, Roles, PasswordPolicy configurations, and Document metadata.

3.2 Decomposition Description

UML Class diagram showing the relationship between the different subsystems.



3.3 Design Rationale

The chosen three-tier architecture (User Interface, Application, and Data layers) was selected for its scalability, maintainability, and separation of concerns. Using React for the UI ensures responsiveness across devices, while C# APIs hosted on AWS Elastic Beanstalk provide reliable

backend logic and integration. MongoDB was chosen for its flexibility in handling dynamic user and document data.

4. DATA DESIGN

4.1 Data Description

The NewUser component handles new user registration. It stores user input in a React state object called formData, which contains fields for first name, last name, address, date of birth, email, username, and password.

Once the user is created, user account information, including usernames, roles, encrypted passwords, account status, and password expiration dates, is stored in the MongoDB database. This allows the system to maintain login credentials securely, enforce password policies, and track account status such as active, inactive, or suspended.

4.2 Data Dictionary

Field Name	Data Type	Description
id	varchar	Unique user ID
firstName	varchar	User first name
lastName	varchar	User last name
address	varchar	User's home address
dob	date	User's date of birth
email	varchar	User's email
username	varchar	Unique username created
passwordHash	varchar	User's password
createdAt	timestamp	Time at which the user is created
administratorID	varchar	Unique ID for administrators
approved	boolean	Indicates whether the account is approved or not

managerID	varchar	Unique ID for managers
-----------	---------	------------------------

5. COMPONENT DESIGN

User Class

Function	Pseudocode Summary
login()	If input username = username & input password = password then login is successful else not
logout()	loggedIn = false
resetPassword()	Password = newPassword
changePassword()	If currentPassword = password then input = password

PasswordPolicy Class

Function	Pseudocode Summary
validate(password)	If input=password then true else false

Document Class

Function	Pseudocode Summary
upload()	If file.isNotEmpty() then fileName = file.name....
download()	If filePath exists, then retrieve file from DB

6. HUMAN INTERFACE DESIGN

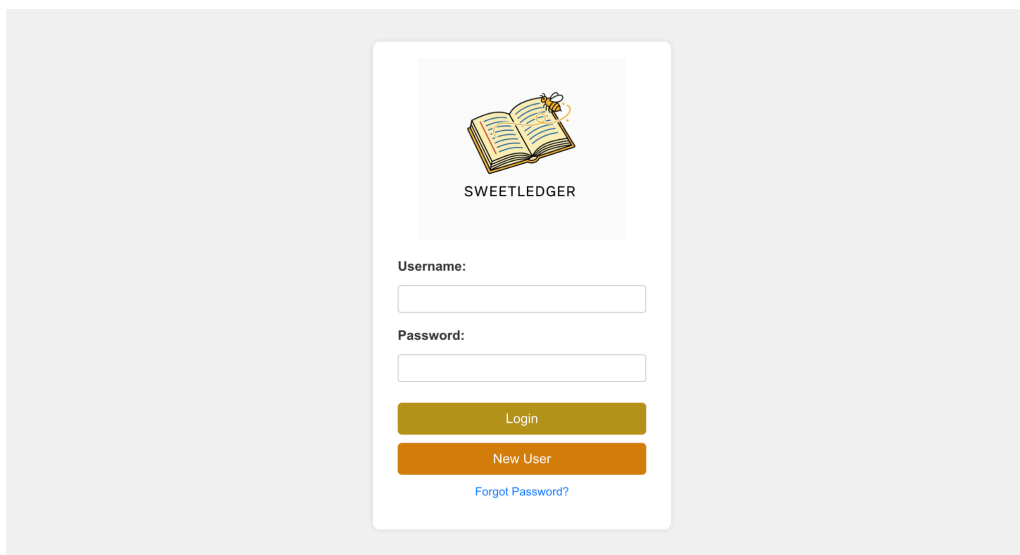
6.1 Overview of User Interface

Upon navigating to the login page, users are presented with clearly labeled text boxes for entering their username and password, along with buttons to submit their credentials, request a new account, or reset a forgotten password. Once successfully logged in, the user's name and profile picture are displayed prominently at the top right corner of the screen, providing immediate confirmation of their identity and access level.

Administrators are presented with additional options, including the ability to create new users, update existing user information, activate or deactivate accounts, suspend users, view all users in the system, and send emails, all through interactive forms and buttons.

Regular users and managers only see the features permitted by their roles, ensuring simplicity and minimizing confusion. Throughout the interface, users receive clear feedback through inline messages for form validation, password complexity requirements, submission confirmations, or error notifications, enabling them to complete tasks efficiently while maintaining security and system integrity. The UI is designed to be responsive and accessible on both desktop and tablet devices, providing a consistent experience across platforms.

6.2 Screen Images



The image shows a login page for 'SWEETLEDGER'. At the top, there is a logo featuring an open book with a bee on it, and the text 'SWEETLEDGER' below it. Below the logo, there are two input fields: 'Username:' and 'Password:'. Under the 'Password:' field, there is a small text label 'Password:'. Below the input fields, there are two buttons: 'Login' (orange) and 'New User' (orange). At the bottom, there is a link 'Forgot Password?' in blue text.

Forgot Password

Username:

Email:

Next

Create New User

First Name:

Last Name:

Address:

Date of Birth:

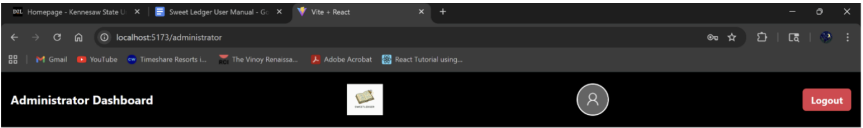
mm/dd/yyyy

Email Address:

Username:

Password:

Submit Request



User Management

No users found.
Failed to load user.

6.3 Screen Objects and Actions

On the Login Page, the main screen objects include text boxes for entering the username and password, a submit button for logging in, a “Forgot Password” button for initiating password recovery, a “Create New User” button for requesting access, and the company logo for consistent branding. Users can type their credentials, click buttons to navigate or submit data, and immediately see feedback messages for validation errors or successful actions.

The Create User Page allows new users to request access. It contains input fields for first name, last name, date of birth, address, and email, along with a submit button. Users fill in their personal information and submit a request, which triggers a notification for administrative approval. The Forgot Password Page contains fields for email, user ID, and answers to security questions, plus a new password field and submit button. Users can reset their passwords, while the system enforces complexity rules and provides real-time feedback through success or error messages.

For Administrators, the User Management Page displays a table of all users and includes buttons to update user information, activate or deactivate accounts, suspend users, and send emails. Administrators can select a user, perform the desired action, and receive immediate feedback through confirmation or error alerts.

7. REQUIREMENTS MATRIX

Requirment-ID	Requirement Description	Design Component	Test Case ID
1001	Admin, user, and manager can log in	LoginForm
1002	Admin can create & assign roles	CreateUserForm
1003	Admin can update user info	UserManagement
1004	Admin can activate/deactive users	UserManagement
1005	User can login in after credentials are created	LoginForm
1006	Display username &	Header

	picture after login		
1007	U/I for login page	Login Form
1008	New user request submission	CreateUserForm
1009	Forgot password	ForgotPassword
1010	Password minimum requirements are enforced	CreateNewUser
1011	Password reuse restricted	Password
1012	Password encrypted	LoginForm
1013	Max 3 failed login attempts	LoginForm
1014	Login info stored in DB	LoginForm
1015	Password expiration notification	LoginForm
1016	Admin reports to view all users	UserManagement
1017	Admin can suspend users	UserManagement
1018	Admin report for expired passwords	UserManagement
1019	Admin can send emails	UserManagement
1020	Username format	CreateUserForm

