SYSTEM DESIGN SPECIFICATION (SDS) FOR INCOME & EXPENSE MANAGEMENT

Student Name: Timothy Kaburu

Registration Number: 24/01455

Unit Code: DIT 503

Unit Title: PROJECT FOR INCOME AND EXPENSE MANAGEMENT

Supervisor: Clive Onsumu

Date Submitted: 7/17/2025

# 1. INTRODUCTION

# 1.1 Purpose

* This System Design Specification (SDS) document details the design and architecture for the Income & Expense Management for the client, ASAD TECH.
* The document translates user and business requirements as stated in the Software Requirements Specification (SRS) into a structured design blueprint for developers, testers, and stakeholders.
* The SDS aims to provide a comprehensive understanding of how the system will be built, including its components, interfaces, data models, and security mechanisms, ensuring that all requirements are met and the application is fit for purpose.

# 1.2 Scope

* This SDS defines the design for an offline, browser-based system to allow individual users (The admin, staff, and other ASAD TECH stakeholders) to manage and track their income and expenses in Kenyan Shillings (Ksh).
* The system will enable secure login, entry management (add, edit, delete), real-time summaries, and easy-to-use interfaces.
* The application will not include server-side storage, multi-user support, or integration with external financial services at this stage.
* All operations are client-side, and data exists only for the session's duration for privacy.

# 1.3 Overview

* The Income & Expense Management Application provides a secure, user-friendly platform for tracking personal financial transactions.
* Its users are primarily the admin and staff at ASAD TECH, but it is also suitable for any individual seeking a simple, private financial management tool.
* The main components include a login screen, a dashboard with financial summaries, entry management forms, transaction history display, and session management features.
* Stakeholders are end users, ASAD TECH’s IT and management teams, project supervisors, and instructors.

# 1.4 Definitions, Acronyms, and Abbreviations

* SDS: System Design Specification
* SRS: Software Requirements Specification
* UI: User Interface
* CRUD: Create, Read, Update, Delete
* ASAD TECH: The client and sponsoring organization
* Ksh: Kenyan Shillings
* Session Data: Temporary data stored in the browser for a single session
* Frontend: The client-side part of the application
* Entry: An individual income or expense record

# 2. SYSTEM OVERVIEW

# 2.1 System Context

* The application is a standalone, browser-based solution. It does not interact with any backend server or external system.
* The system is accessed by users via a web browser, and its only dependencies are HTML, CSS, and JavaScript support.
* All data created during a session resides in memory; it is not persisted beyond logout or page refresh.
* The application is designed for personal use, respecting privacy and requiring no installation or special configuration.

System Context Diagram (Narrative)

* User: Interacts with the application through a web browser.
* Web Application: Renders UI, handles logic, and manages session data in memory.
* External Systems: None in current version. All operations are self-contained.

# 2.2 Assumptions and Dependencies

* Users have access to a modern web browser (e.g., Chrome, Edge, Firefox, Safari).
* JavaScript is enabled in the browser.
* The application is used for individual, non-commercial purposes only.
* No connection to external APIs, databases, or authentication services is necessary.
* The only required infrastructure is a static file server (optional; can run locally).

# 3. SYSTEM ARCHITECTURE

# 5.3.1 High-Level Architecture Diagram

+---------------------+

| Web Browser |

+---------------------+

|

v

+-------------------------------+

| Presentation Layer (HTML/CSS) |

+-------------------------------+

|

v

+-------------------------------+

| Logic Layer (JavaScript) |

+-------------------------------+

|

v

+-------------------------------+

| Data Layer (In-Memory Array) |

+-------------------------------+

Architectural Narrative

* Users interact with the presentation layer rendered in their browsers.
* The logic layer, implemented in JavaScript, manages user actions, validation, and application logic. The data layer is a JavaScript array stored in browser memory for the duration of the session. This design supports responsiveness, privacy, and simplicity.

5.3.2 Modules and Components

1. Authentication Module

Handles login and logout.

Validates credentials (user / 1234).

Controls access to the main dashboard.

2. Dashboard Module

Displays real-time summaries (income, expense, balance).

Provides navigation to entry management and logout.

3. Entry Management Module

Allows adding, editing, and deleting income and expense entries.

Validates input and updates session data.

4. History Module

Shows a tabular view of all transactions.

Supports filtering and color coding by entry type.

5. UI Components

Responsive layouts, forms, buttons, summary cards, and tooltips.

6. Session Management

Ensures all data is cleared on logout or refresh.

Does not persist any information beyond the session.

5.3.3 Technology Stack

Languages: HTML5, CSS3, JavaScript (ES6+)

Frameworks/Libraries: None (vanilla JS; can be extended in future)

Design/Prototyping Tools: Figma, Draw.io, Eraser (for flowcharts)

Testing: Chrome DevTools, BrowserStack, manual user acceptance testing

Deployment: GitHub Pages, Netlify, or any static file server

5.4. Database Design

5.4.1 Data Model/Schema

Conceptual Schema

Field Type Description

id String Unique identifier (UUID)

desc String Description of income/expense

amount Number Amount in Ksh

type String 'income' or 'expense'

created\_at Date Timestamp of creation

modified\_at Date Timestamp of last modification

Example Data Structure (JavaScript)

[

{

id: "a1b2c3",

desc: "Book Purchase",

amount: 1200,

type: "expense",

created\_at: "2025-07-17T10:00:00Z",

modified\_at: "2025-07-17T10:00:00Z"

},

...

]

5.4.2 Data Flow Diagrams

Level 0 (Context Diagram)

[User]

|

v

[Income & Expense App]

Level 1 (Process Flow)

[User]

|

v

[Login Module] -> [Dashboard]

| | |

| v v

| [Entry Management] <-> [History]

| |

+--------------> [Logout]

Narration:

Users must first log in. Once authenticated, they can view the dashboard, add/manage entries, and view history. Logging out clears all data.

5.4.3 Data Storage

In-memory Storage: All data is stored in JavaScript arrays during the session.

No Persistence: Data is lost on logout or refresh, promoting privacy.

Retrieval: Data is accessed in real-time for dashboard and history updates.

Backup: No backup functionality is provided.

5.4.4 Data Security and Privacy

No data is transmitted or stored beyond the user’s browser session.

Data is purged on logout or refresh.

No personal information is stored except the session’s financial entries.

5.5. Program Design

5.5.1 Module Design

Authentication Module

Pseudocode:

function login(username, password):

if username == 'user' and password == '1234':

grantAccess()

else:

showError("Invalid credentials")

Flowchart:

[Start] -> [Login Form] -> [Validate Credentials] -> [Dashboard]

| ^

v |

[Show Error] <-------------

Entry Management Module

Pseudocode:

function addEntry(desc, amount, type):

if desc is not empty and amount > 0 and type in ['income', 'expense']:

add to entries array

updateDashboard()

updateHistory()

else:

showError("Invalid entry")

Flowchart:

[Add Entry] -> [Validate Input] -> [Update Array] -> [Update UI]

| |

v v

[Show Error] [Show Success]

Dashboard Module

Calculates total income, expense, and balance on each update.

Triggers re-render of summary cards.

History Module

Renders the transaction table.

Allows filtering and (future) editing/deleting entries.

Session Management

Invoked on logout or page refresh.

Clears all session data.

5.6. Interface Design

5.6.1 User Interface

Login Screen

Username and password input fields

Login button

Error message area

Demo credentials note

Dashboard

Application header with title and logout option

Three summary cards for income, expenses, and balance

Entry form for description, amount, and type

Transaction history table (sortable, color-coded)

Responsive design for desktop and mobile

Entry Form

Input fields: Description (text), Amount (number), Type (dropdown: income/expense)

Add Entry button

Transaction History

Table with columns: Description, Amount, Type

Color codes: Green for income, red for expense

Wireframes

(Attach separate Figma or Draw.io screens as needed. Example layouts: login screen, dashboard, add entry form, history table.)

5.6.2 External System Interfaces

None in this version

Future enhancements may include CSV export/import, cloud sync, or API integration.

5.7. Deployment Design

5.7.1 Infrastructure Diagram

[User Device]

|

[Web Browser]

|

[Static File Server (optional)]

Description

The application is distributed as static HTML/CSS/JS files.

Can be hosted on GitHub Pages, Netlify, or run directly from local storage.

5.7.2 Deployment Strategy

Initial Deployment: Upload all files to static hosting or distribute as a zipped folder.

Updates: Replace existing files with new versions.

Rollback: Restore previous file versions if necessary.

No server-side or database migration required.

5.8. Security Design

5.8.1 Authentication and Authorization

Simple login with demo credentials (user/1234).

No user registration or advanced roles.

All access to dashboard is gated by login.

5.8.2 Data Protection

Data only exists in the browser’s memory for the session.

No persistent storage or network transmission.

User is warned that data is lost on logout or refresh.

5.8.3 Audit and Logging

No persistent logging for privacy.

Client-side logs (console) are for developer debugging.

No compliance or audit trail required for this version.

References

IEEE Std 830-1998: Software Requirements Specifications

MDN Web Docs: HTML, CSS, JavaScript documentation

ASAD TECH internal documentation and requirements

Draw.io, Figma, Eraser (for diagrams and wireframes)

GitHub Pages/Netlify documentation for deployment

This System Design Specification was prepared for ASAD TECH and is intended to guide the development, testing, and future maintenance of the Income & Expense Management Web Application (Ksh Edition). The document will serve as a reference throughout the project lifecycle and should be updated as system needs evolve.