# ClariFi

Architecture/Design Document

## Table of Contents

Change History

_____

**Version:** 0.1
**Modifier:** Haley Nilsen, Abdul Mohammed
**Date:** 09/26/2025
**Description of Change:** Initial draft

_____

# 1  Introduction

This document describes the architecture and design for the ClariFi application being developed for Commerce Bank in partnership with the University of Missouri – Kansas City (UMKC). ClariFi is a web application that allows users to set financial goals and easily track them all in one spot. The application will have LLM integration that will assist users in answering any questions the user may have.  ClariFi offers a simple, yet bold user interface so users are easily able to use and navigate the website.

The purpose of this document is to describe the architecture and design of the ClariFi application in a way that addresses the interests and concerns of all major stakeholders. For this application the major stakeholders are:

- Users and the customer – they want assurances that the architecture will provide system functionality and exhibit desirable non-functional quality requirements such as usability, reliability, etc.
- Developers – they want an architecture that will minimize complexity and development effort.
- Project Manager – the project manager is responsible for assigning tasks and coordinating development work. He wants an architecture that divides the system into components of roughly equal size and complexity that can be developed simultaneously with minimal dependencies. For this to happen, the modules need well-defined interfaces. Also, because most individuals specialize in a particular skill or technology, or have interest in certain technologies, modules should be designed around specific expertise/interests. For example, all UI logic might be encapsulated in one module. Another might have all logic related to financial management tasks

The architecture and design for a software system are complex and individual stakeholders often have specialized interests. There is not one diagram or model that can easily express a system's architecture and design. For this reason, software architecture and design is often presented in terms of multiple views or perspectives [IEEE Std. 1471]. Here the architecture of the ClariFi application is described from 4 different perspectives [1995 Krutchen]:

1. Logical View – major components, their attributes and operations. This view also includes relationships between components and their interactions. When doing OO design, class diagrams and sequence diagrams are often used to express the logical view.
2. Process View – the threads of control and processes used to execute the operations identified in the logical view.
3. Development View – how system modules map to development organization.
4. Use Case View – the use case view is used to both motivate and validate design activity. At the start of design the requirements define the functional objectives for the design. Use cases are also used to validate suggested designs. It should be possible to walk through a use case scenario and follow the interaction between

high-level components. The components should have all the necessary behavior to conceptually execute a use case.

## 2 Design Goals

There is no absolute measure for distinguishing between good and bad design. The value of a design depends on stakeholder priorities. For example, depending on the circumstances, an efficient design might be better than a maintainable one, or vise versa. Therefore, before presenting a design it is good practice to state the design priorities. The design that is offered will be judged according to how well it satisfies the stated priorities.

The priorities for the design that follows are:

- **The design should allow for additional feature in the future (ex: different AI models or visualization tools) without much rework**

- **Maintain a separation of convers**

- **UI, backend logic, AI processing, data storage, and visualization should be well-defined so that changes made to one area do not affect other parts of the process**

- **Design should be easy to implement and maintain, this will ensure faster development**

- **Allow for reuse wherever possible**

## 3 System Behavior

The use case view is used to both drive the design phase and validate the output of the design phase. The architecture description presented here starts with a review of the expect system behavior in order to set the stage for the architecture description that follows. For a more detailed account of software requirements, see the requirements document

## 4 Logical View

The logical view describes the main functional components of the system. This includes modules, the static relationships between modules, and their dynamic patterns of interaction.

In this section the modules of the system are first expressed in terms of high level components (architecture) and progressively refined into more detailed components and eventually classes with specific attributes and operations.

### 4.1 High-Level Design (Architecture)

The high-level view or architecture consists of 5 major components:
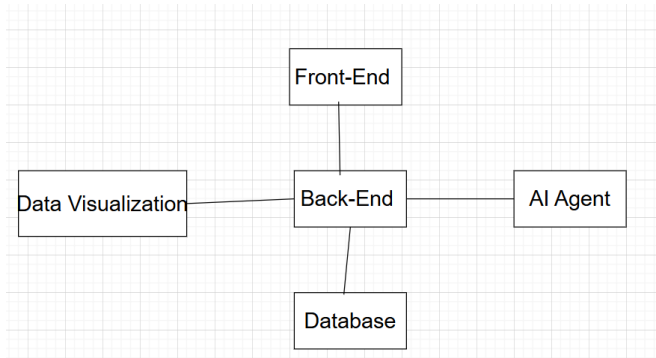
- The **frontend** will be responsible for the visual aspect of the app, providing all user interactions. It can collect user input such as text and display things like tables, charts or dashboards.
- The **backend** will be receive request from the frontend, validate and process the user input and data and handel the connections between AI agent and database. Additionally it will also log the inputs and outputs into the database while formatting the visualization results.
- The **database** will store the user profile which would include name, email, phone number, along with financial information for each user
- The **AI Agent** will take in user input on the chatbot page then work within the agent hierarchy to deliver the user's desired results
- The **data visualization** tool will work in collaboration with the AI agent to ensure that the user is given their requested visual

## *4.2 Mid-Level Design*

## *4.3 Detailed Class Design*

# 5 Process View

# 6 Development View

# 7  Physical View

[TBD]

# 8  Use Case View

(The following is more algorithm than use-case view. Fix later.)

**Basic Tour:**
1.