SaveNScore

Architecture/Design Document

**Table of Contents**

[1 Introduction 3](#_Toc131876418)

[2 Design Goals 4](#_Toc131876419)

[3 System Behavior 5](#_Toc131876420)

[4 Logical View 5](#_Toc131876421)

[4.1 High-Level Design (Architecture) 6](#_Toc131876422)

[4.2 Mid-Level Design 7](#_Toc131876424)

[4.3 Detailed Class Design 8](#_Toc131876425)

[5 Process View 9](#_Toc131876426)

[6 Physical View 10](#_Toc131876428)

[7 Use Case View 10](#_Toc131876429)

Change History

**Version:** 1.0

**Modifier:** Benjamin Rahman

**Date:** Finalized 04/01/2019

**Description of Change:** Initial Document

**Version:** 2.0

**Modifier:** Benjamin Rahman

**Date:** Finalized 05/04/2019

**Description of Change:** Diagram Updates

# Introduction

**Architecture and Design**

This document describes the architecture/design of the SaveNScore web application being developed by UMKC-CS451-Group 5 for Commerce Bank.

SaveNScore is a web-based application for personal finance requiring no computer knowledge outside of the ability to access the internet and an understanding of UI conventions. The purpose of the SaveNScore application is to empower Users to take control of their finances by setting personal, and/or business expense goals in as detailed a manner as they prefer. In addition, the SaveNScore application also boasts a Goals feature allowing the User to attach goal(s) to any account; thus, providing an element of gamification to entertain all types of Users.

The purpose of this document is to describe the architecture and design of the SaveNScore 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 for 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 or she 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, modules should be designed around specific expertise. For example, all UI logic might be encapsulated in one module. Another might have all business logic.
* Maintenance Programmers – they want assurance that the system will be easy to evolve and maintain on into the future.

The architecture and design for a software system is complex and individual stakeholders often have specialized interests. There is no 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 SaveNScore 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. Physical View – outlines the responsibilities of each system involved in the architecture.
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.

# Design Goals

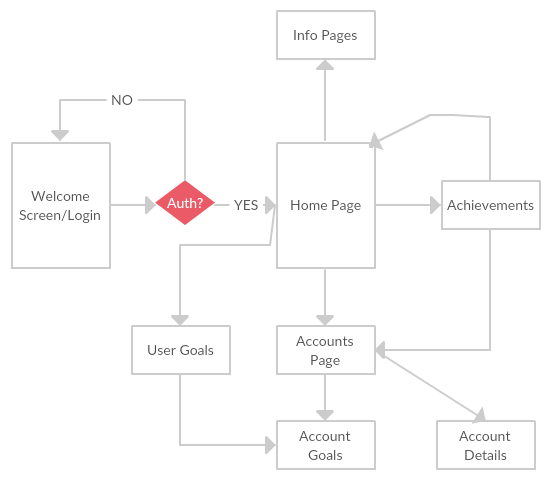
There is no absolute measure for distinguishing between good and bad design. The value of a design depends on stakeholder priorities. For example, depending on the circumstances, an efficient design might be better than a maintainable one, or vise versa. Therefore, before presenting a design it is good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

The design priorities for the SaveNScore application are:

* The design should minimize complexity and development effort.
* The design should package components into neatly organized, logical modules. However, it should be noted that these modules shouldn’t be created or organized with the intention of evolving the product in the future.
* The design should take into account the size of the team creating the product and offer flexibility on non-critical feature implementation.

# 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.

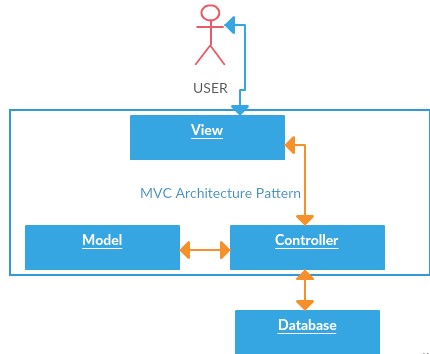


# Logical View

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.

## High-Level Design (Architecture)

The high-level view or architecture consists of 4 major components: 

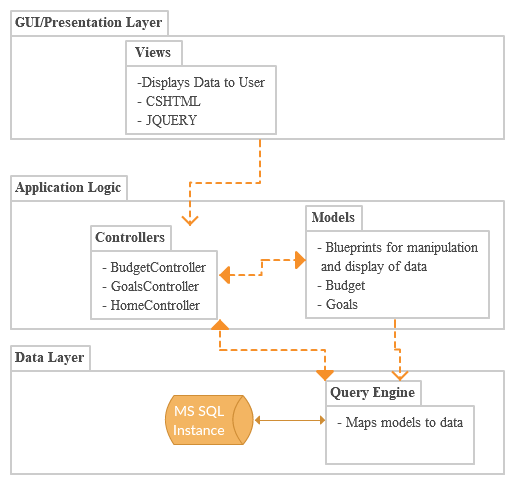
View - > Represents the Graphical User Interface the User interacts with through their choice of browser. The primary purpose of the View component is to display its corresponding Model’s data to the User. In addition, this system also allows the user to modify the data which will then be passed to the Controller and finally to the Database for storage.

Model -> Represents the structure of the data and business logic. Model Objects are then used by the Controller to retrieve and store data in the Database.

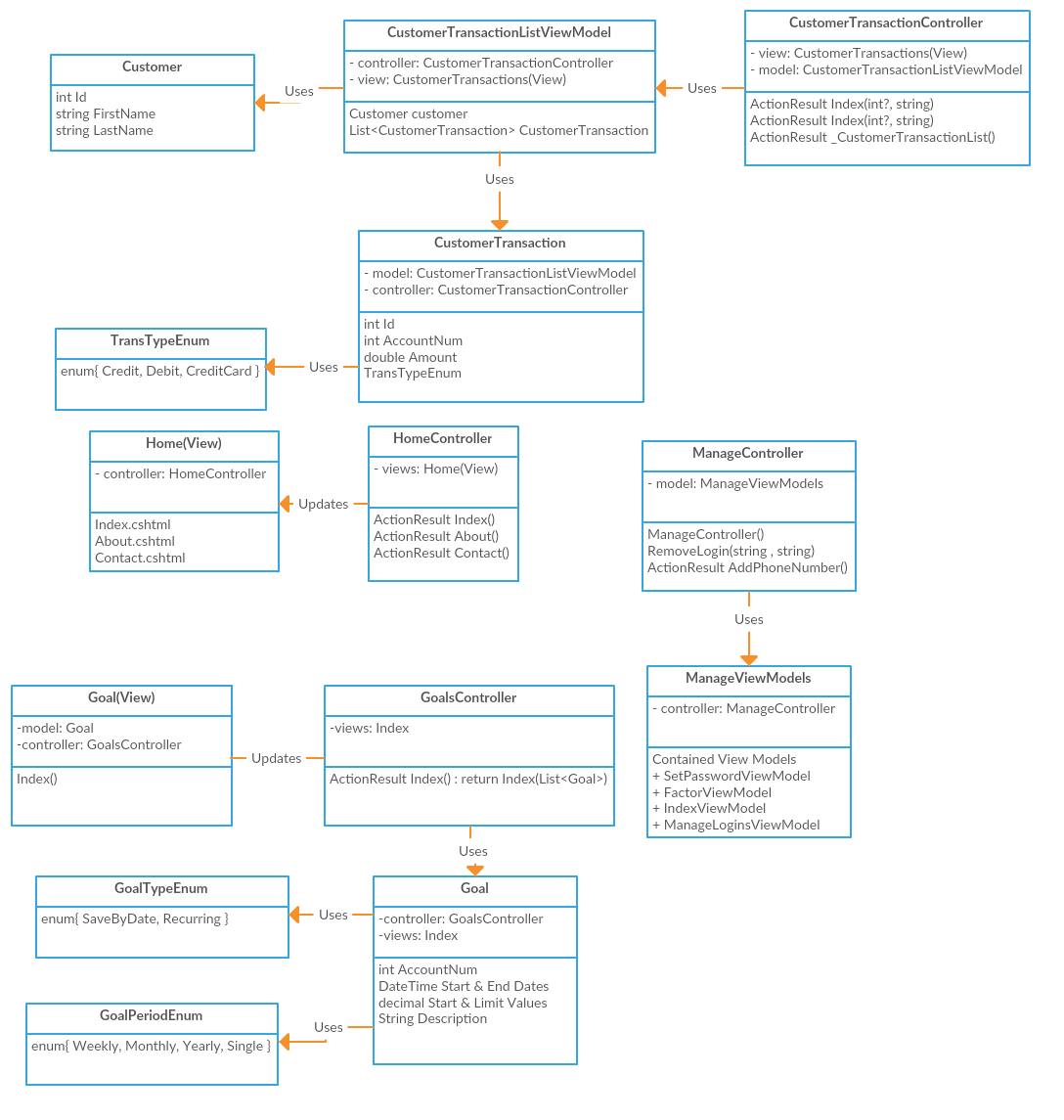
Controller -> Handles interactions between the User (via a View) and the Model data of whichever Model the User’s current View corresponds too. In addition, the Controllers handle all User requests and responses.

Database -> Currently, the database is a Local instance of an MS SQL DB that is automatically dropped and recreated each time a new system connects. The primary purpose of the database is to store data in tables corresponding to Models.

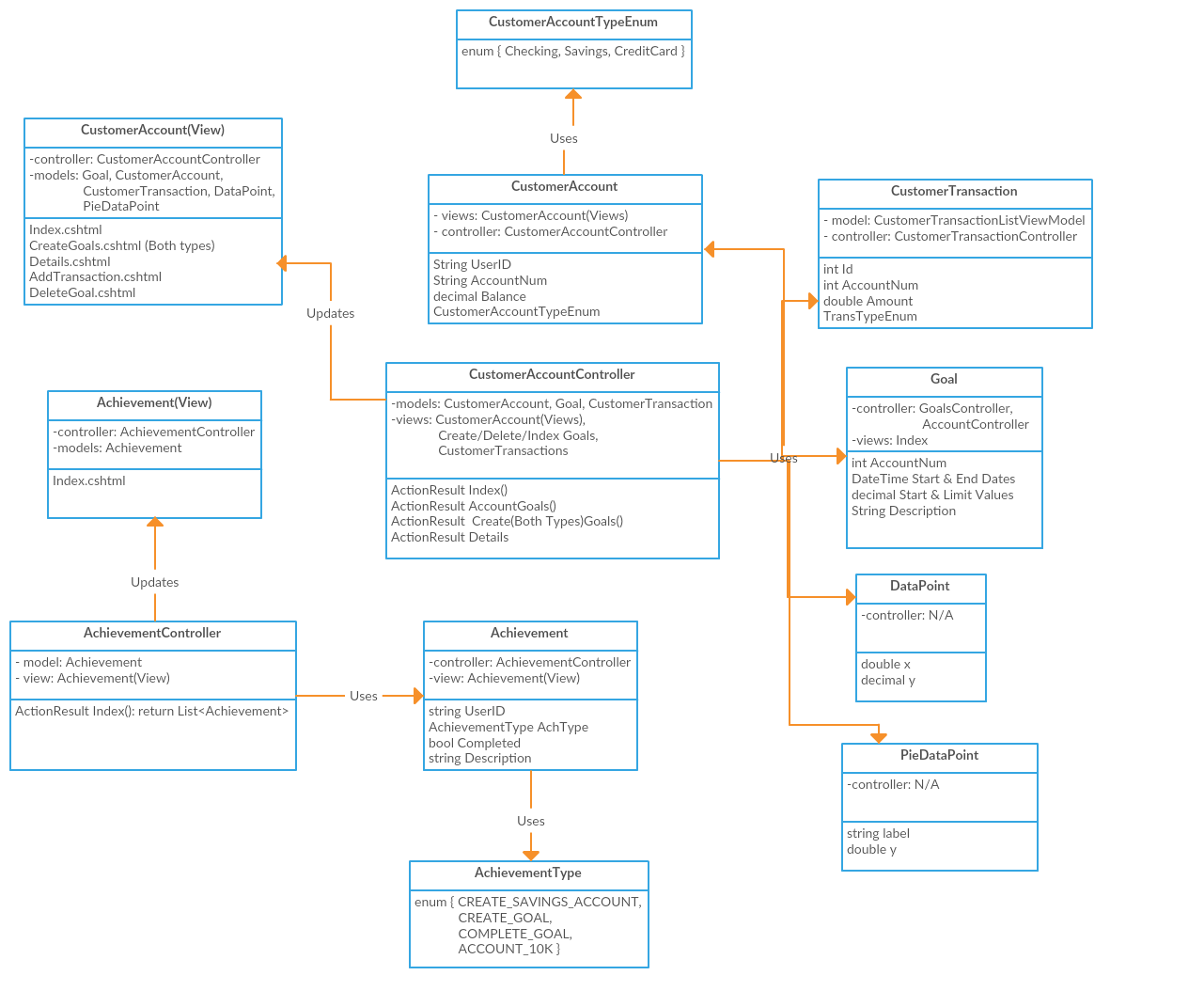
## Mid-Level Design



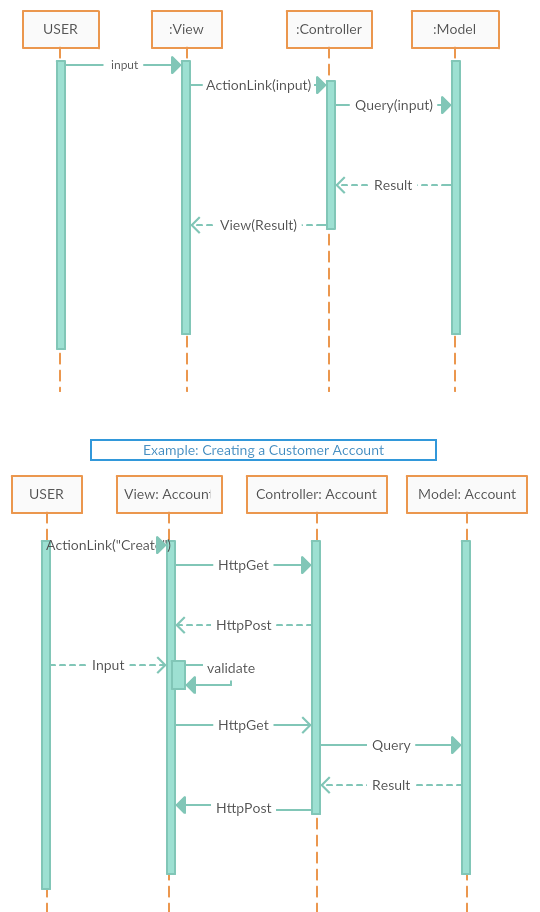
## Detailed Class Design



# 



# Process View



# Physical View

In its current state the SaveNScore application uses the Client-Server architecture. The client’s system uses HTTP protocols to access the web application from a browser. Once connected to the server the client’s system is only responsible for receiving and displaying data from an HTTP POST protocol. The server on the other hand is responsible for both receiving and sending data through HTTP protocols. In addition, the server is responsible for creating and maintaining a local MS SQL database instance to house client data.

# Use Case View

