Commerce Bank Budgeting Tool

Architecture/Design Document

**Table of Contents**

[**1**](#_gjdgxs) **INTRODUCTION 3**

[**2**](#_30j0zll) **DESIGN GOALS 4**

[**3**](#_1fob9te) **SYSTEM BEHAVIOR 4**

[**4**](#_3znysh7) **LOGICAL VIEW 5**

[**4.1**](#_2et92p0) **High-Level Design (Architecture) 5**

[**4.2**](#_3dy6vkm) **Mid-Level Design 6**

[**5**](#_4d34og8) **PROCESS VIEW 11**

**6** **DEVELOPMENT VIEW 12**

[**7**](#_2s8eyo1) **PHYSICAL VIEW 12**

[**8**](#_17dp8vu) **USE CASE VIEW 12**

Revision History

|  |  |  |
| --- | --- | --- |
| **Author** | **Date** | **Description** |
| Jonathan Whipple | 3/14/17 | Added high Level Architecture - Section 4.1 |
| Jeremiah Trahern | 3/14/17 | Small addition to Design Goals - Section 2 |
| Jeremy Troshynski | 3/14/17 | System Behavior - Section 3 |
| Emily Thomas | 3/15/17 | Threads in Process View - Section 5 |
| Alex Sweeney | 3/15/17 | Introduction edits, added class sequence diagram |
| Colin Zeiler | 3/15/17 | Use Case View - Section 8 |

# **Introduction**

**Architecture and Design**

The purpose of the architecture/design document is to explain the organization of the code. A well-written architecture document will make it easier for new programmers to become familiar with the code.

The architecture/design document should identify major system components and describe their static attributes and dynamic patterns of interaction.

Software architecture and designs are typically expressed with a mix of UML models (class and sequence diagrams being the two most common) and prose. Dataflow diagrams are also helpful for understanding the interaction between components and overall flow of data through the system.

**About this Template**

This template suggests one way of documenting a software system’s architecture/design. You aren’t required to include every section in this template nor all the content in the sections you do include. However, the document you do submit should pass the following checklist:

* Are design objectives clearly stated? For example, if performance is more important than reusability, this should be made clear at the start of the design specification.
* Does the architecture partition the implementation into clearly defined subsystems or modules with well-defined interfaces?
* Does the architecture express in a clear way the main patterns of communication between subsystems and modules?
* Does the architecture satisfy the requirements?
* Is the architecture traceable to requirements?
* Any models created should either be expressed with a well-known modeling language, or if a well-known modeling language isn't used, the syntax and semantics of the symbols that are used should be defined.

This document describes the architecture and design for the Budgeting Tool application being developed for Commerce Bank. The Budgeting Tool provides users with help budgeting their money. The Tool supports the users endeavors by creating a visual look into their money, emphasizing saving by inducing a badge system.

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

Here the architecture of the Budgeting Tool application is described from 4 different perspectives:

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.

# **Design Goals**

The design priorities for the Commerce Bank Budgeting Tool are:

* The design should minimize complexity and development effort.
* The design should make the application easily maintainable.
* The design should make the application easily adaptable into another environment
* The design should provide a variety of types of users with ease of use, and intuition when using the system.
* The design should allow for the base application to be easily expanded upon.

# **System Behavior**

The Commerce Bank Budgeting Application will allow users to interact with their bank accounts in a way that promotes better spending habits. The system will display to users a list of their previous transactions with an associated category tag. These transactions will also be grouped by category within graphs, to allow for users to better understand how their money is being spent. Once a user has set up their budgeting goals for each spending category, their spending throughout the month will be presented to them and they will be notified if they overspend in a given month. The system will allow users to earn badges by completing specific tasks. The system will allow users to create non-recurring savings goals to achieve.

# **Logical View**

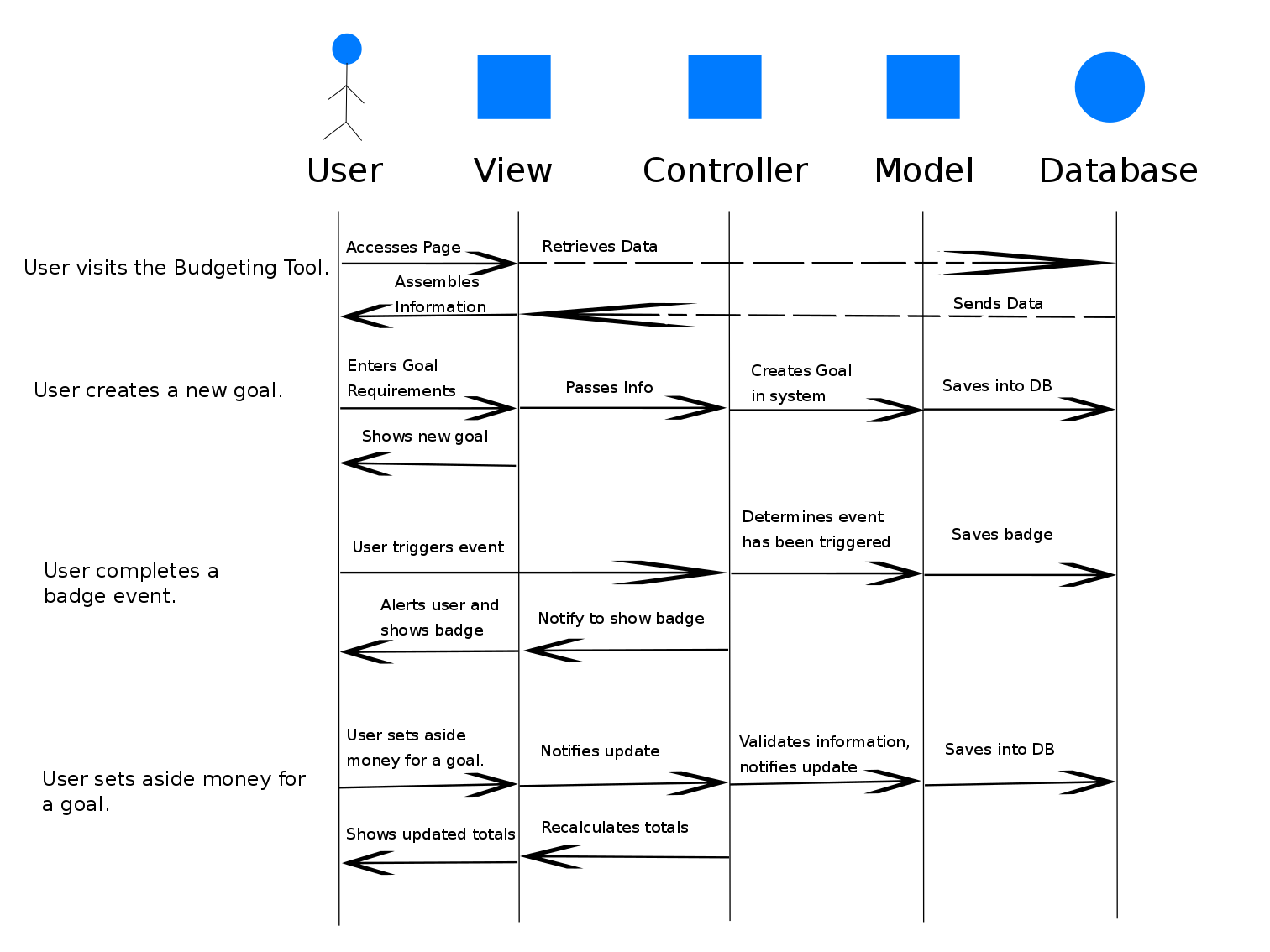
## ***High-Level Design (Architecture)***

The high-level view or architecture consists of 5 major components:



* The View consists of the web pages that users will see
* The Controller consists of the Logic of what to display or update on a page.
* The Model consists of the structure of how information is held or stored.
* The Database is the central location that stores off critical data.
* The User is the person or program using the application

## ***Mid-Level Design***



# **Process View**

**Dashboard thread**

This thread runs when the user opens the app. It is in charge of setting up the display. This includes the fetching and display of data in graphs (goals) and tables (transactions, categories). The thread also awaits actions by the user and responds as needed to those actions, such as adding a new category, adding a new goal, editing a goal, viewing information on transactions, badges/points earned for accomplished goals, progress toward goals, etc.

**Time thread**

This thread is continually running, whether the user has opened the app or closed it. Even when the app is not in use, there is an internal clock and management system that checks the countdown on goal periods and closes goals when their end dates are reached.

# **Physical View**

[TBD]

# **Use Case View**

