Commerce Bank Project

Architecture/Design Document

**Table of Contents**

**1 Introduction.. 3**

**2 Design Goals.. 3**

**3 System Behavior.. 4**

**4 Logical View... 5**

**4.1 High-Level Design (Architecture) 5**

**4.2 Mid-Level Design. 6**

**4.3 Detailed Class Design. 7**

**5 Process View... 8**

*6 Development View... 8*

*7 Physical View... 8*

*8 Use Case View... 8*

Change History

**Version:** 1.0

**Modifier:** Nick King

**Date:** 3/10/2017

**Description of Change:** Baseline

**Version:** 1.1

**Modifier:** Caleb Hillhouse

**Date:** 4/24/17

**Description of Change:** Changed system behavior graph

# **1** **Introduction**

**Architecture and Design**

The purpose of this architecture/design document is to explain the organization of the code to make it easier for new programmers to become familiar with the code.

This architecture/design document will 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.

*This document describes the architecture and design for the Commerce Bank Project application being developed for Commerce Bank online banking users. This software will provide gamification incentives to encourage users to set savings goals and budget their spending.*

The purpose of this document is to describe the architecture and design of the Commerce Bank Project 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 Commerce Bank Project application is described from 4 different perspectives [1995 Krutchen]:

1. Logical View – See [4.2] Mid-Level Design
2. Process View – See [5] Game Thread, GUI Thread
3. Development View – See [4.2] future edits. See how system modules map to development organization.
4. Use Case View – See [7] 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 design priorities for the Commerce Bank Project application are, in no desired order:

* The design should minimize complexity and development effort.
* The design will provide for maximum readability within the code, to increase the probability of our design being used in production.
* The design should allow for simplicity in the integration of new modules and features.
* The design will address the requirements listed in the requirements document.
* The design will prioritize simple modules to be used by gamification features.

# **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 expected system behavior to set the stage for the architecture description that follows.

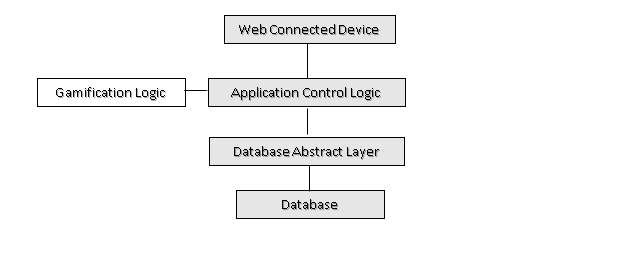
# **4** **Logical View**

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:



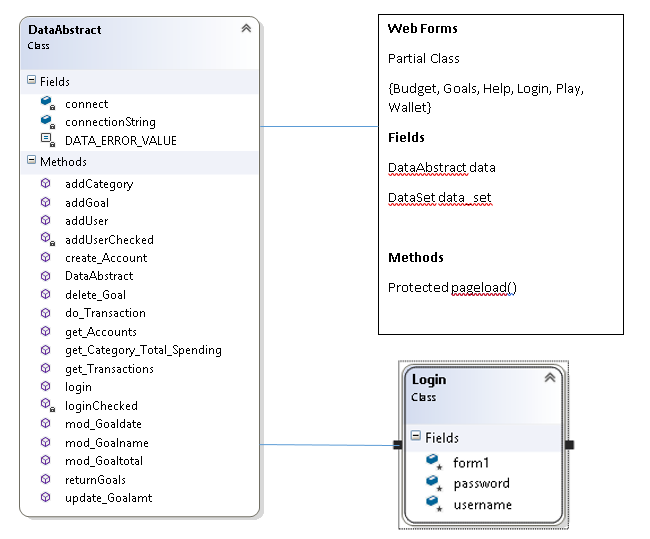
|  |  |
| --- | --- |
|  |  |
|  |  |

System Architecture

* The **Web Connected Device** provides the user with an interface to interact with the application
* The **Database** is a central repository for data on users, accounts, game accounts, and other necessary data for budgeting tools
* The **Database Abstraction Layer** is the collection of functions to query the database-- it responds to calls from the Application Control Logic layer.
* The **Gamification Logic**  is the driver for all game related information and is in charge of presenting game information to the user and reacting to user inputs.
* The **Application Control Logic** is the main driver of the application. It presents information to the user and reacts to user inputs.

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

Class Diagram -- will be edited as new classes are implemented



## **4.3** **Detailed Class Design**

--webforms include the abstract layer and create Data abstract objects that allow for the database to be queried through the webform. This data allows for the webform logic to manipulate the U.I. through scripts. The backbone logic is largely built, the front end logic is still being implemented.

The webforms use a DataAbstract object to call these functions.

*An example:*

public int mod\_Goaltotal(string goal\_name, double new\_total, int acctNumber)  
 {  
 string updateSQL = "UPDATE Goals SET TotalAmt = @TotalAmt WHERE Name = " + "'" + goal\_name + "'" + "AND AcctNumber = " + acctNumber;  
 SqlCommand Cmd = new SqlCommand(updateSQL, connect);  
 Cmd.Parameters.AddWithValue("@TotalAmt", new\_total);  
   
 connect.Open();  
 Cmd.ExecuteNonQuery();  
 connect.Close();  
 return 1;  
 } // end mod goal total

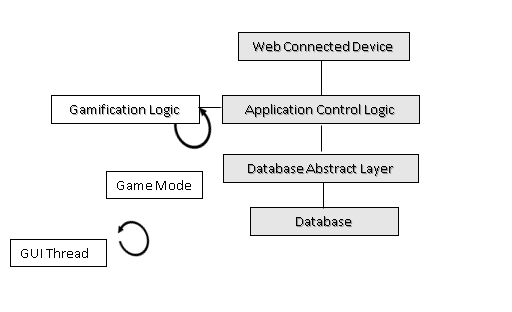
^a simple function query that modifies the data through the webform^

*More detailed interaction:*

public DataSet get\_Transactions(int acct\_num)  
 {  
 //selects accounts for a specific suer id  
 string selectSQL = "SELECT \* FROM Transactions WHERE AcctNumber = " + acct\_num;  
   
 SqlCommand cmd = new SqlCommand(selectSQL, connect);  
   
 DataSet temp = new DataSet();  
 SqlDataAdapter adapter = new SqlDataAdapter(cmd);  
   
 connect.Open();  
 adapter.Fill(temp);  
 connect.Close();  
 return temp;  
 } //end get\_Transactions

-More details/diagram work to come as features get implemented.

# **5** **Process View**



--GUI Thread

--Game Thread

# **6** **Development View**

--for future edit as front end service modules are developed

# **7** **Physical View**

--for future edit as front end communication and UI features are finalized

# **8 Use Case View**

--for future edit as we develop use case testing