|  |
| --- |
| Commerce Bank |
| IdeaBank |
| Architecture Document |

|  |
| --- |
| Jordan Larson, Anthony Laughlin, Matthew Parra, John Michaels, & Hayden McParlane  11-6-2014 |

# 1. Introduction

This architecture document is tailored to describe the architecture of the IdeaBank, a web application being developed for Commerce Bank in order to address increasing numbers of software engineering projects that Commerce Bank shares with local universities. There are three levels of end users who will access the IdeaBank – admin, ambassador, and contributor. These end users should be prioritized during development.

For this project, the major stakeholders are as follows:

* Commerce Bank employees – CB employees desire a ubiquitously usable web app environment where they can share ideas on future technological projects that will bring value to the company. They want an application that will be reliable, usable, and, ideally, portable to a mobile platform.
* Developers – developers desire an architecture that will decrease development effort by having architecture document items available to drive and enhance workflow.
* Project Manager – since the project manager is the overseer of the project, a lot of the planning rests on him. The architecture should, at a minimum, enable the manager to divide tasks proficiently and effortlessly. To make this possible, multiple perspectives are provided for the architecture that will separate the layers of front-end and back-end logic components.
* Maintenance Programmers – these programmers want to know that the system will be the maintainable, scalable, and reliable

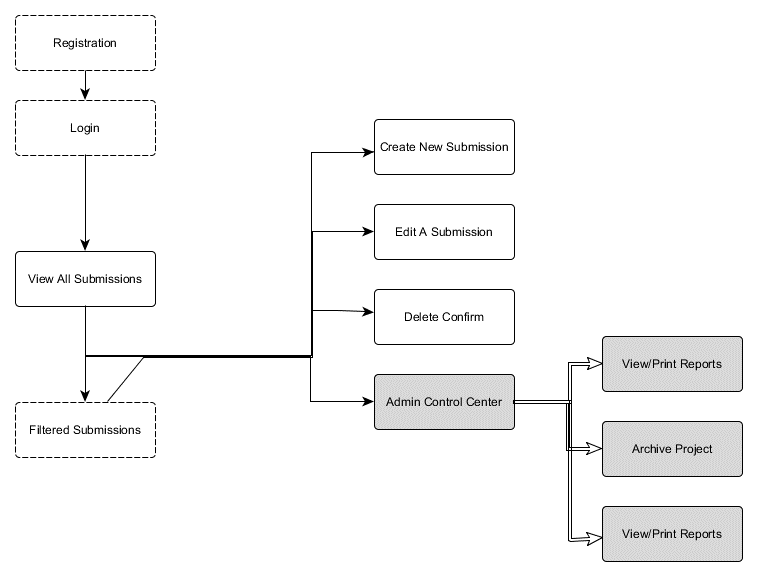
As mentioned in the project manager stakeholder desires listed above, this document will contain multiple architecture perspectives. They are broken up in to separate sections as follows:

1. Logical View
2. Process View
3. Development View
4. Use Case View

# 2. Design Goals

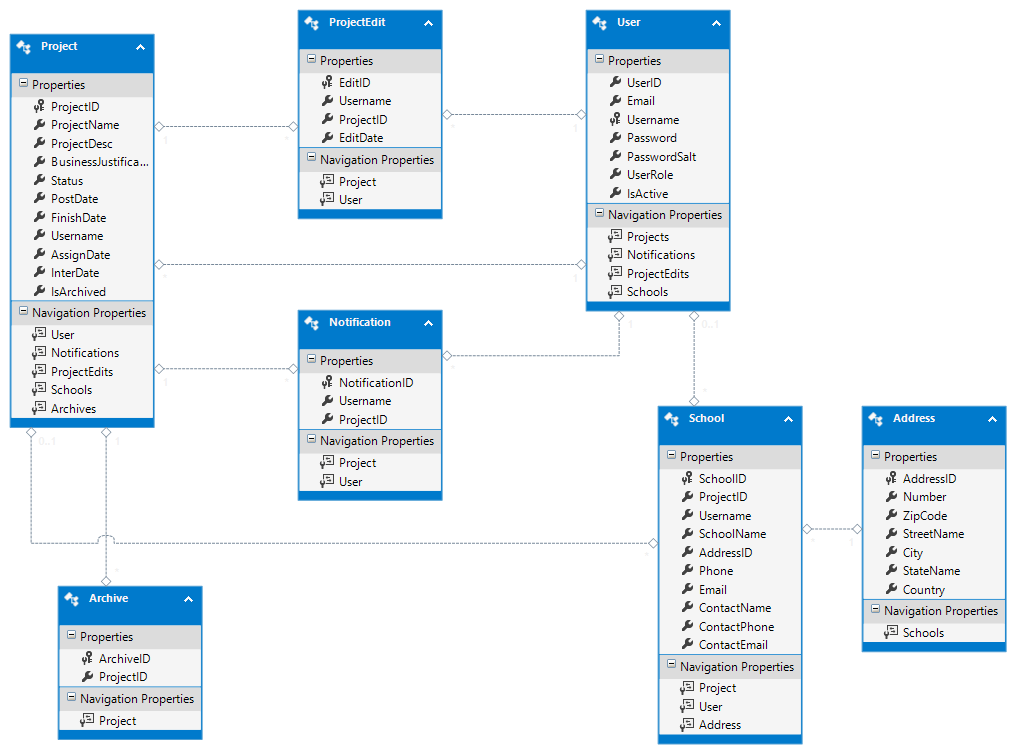
* The design should possess enough abstraction to simplify the complexity of the task at hand.
* The design should minimize the effort expended by developers.
* The design should possess enough separable components so that each development member can work autonomously on a component without being dependent on another component that is undeveloped or underdeveloped.
  + Furthermore, each member of the team should be able to focus on an individual aspect of development, such as UI design (css, cshtml, etc), database management, or model/view model components.
* Reusability should be of high importance to the design goals. The utility, in its final form, should be one that will contain reusable code for future maintenance programmers.

# 3a. System Behavior



Once a registered user enters the system and logs in, he is privy to all options besides the grayscale items. Those are reserved for admins. Administrator views are TBD at this point, but the above diagram depicts the system in its entirety, with its individual changes in behavior as user interactions occur.

# 3b. System Database Schema



# 4. Logical view.

## 4.1 High-Level Design (Architecture).

## The View present in the diagram above represents any web application user-interface. In this case, that translates to any web page with which a user can interact.

* + The **Controller** provides communication between each view and the model by use of the **Business Logic** interface.
  + The **Business Logic** component provides all of the web application controllers with access to the underlying model in a controlled and specific manner. This constrains the functionality of the web application so as to enforce the business specifications detailed during the requirements meeting.
  + The **Domain Model Access Interface** provides an extremely flexible interface into the entity framework allowing the **Business Logic** component to receive as much functionality as possible from the underlying database.
  + The **Task Handler** provides a mechanism by which the **Domain Model Access Interface** can manipulate collections of domain models and the **Query Handler** allows for the **Task Handler** to manipulate domain models individually.
  + The **Entity Framework** provides an extremely easy to use interface into the database providing querying capabilities along with data manipulation capabilities.
  + The **Database** provides a storage location for the application data. Examples of such data include projects, registered users, subscriptions, project edits, etc.

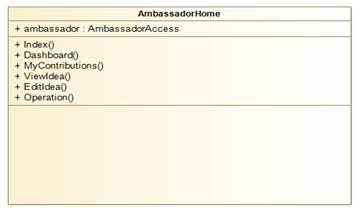
## 4.2 Detailed Class Design

# C:\Users\freedm\AppData\Local\Temp\msohtmlclip1\01\clip_image002.jpgFigure 1: The ModelAccess layer. Each higher level of security clearance adds additional functionality.



# Figure 2: AdminHome controller class.

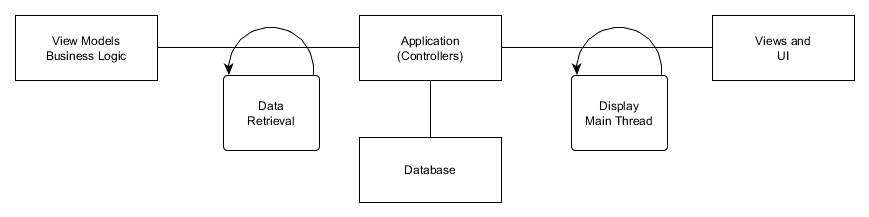


**Figure 3: AmbassadorHome controller class.** 

# Figure 4: The ContributorHome controller class.

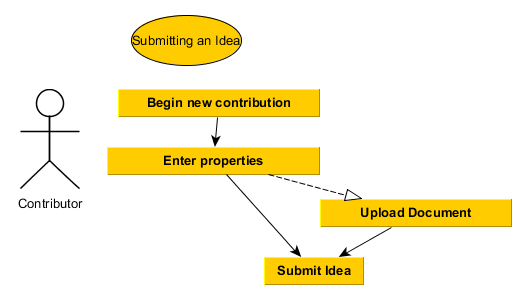
# C:\Users\freedm\AppData\Local\Temp\msohtmlclip1\01\clip_image008.jpg

# 5. Process View

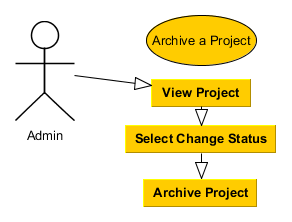


# 7. Use Case View

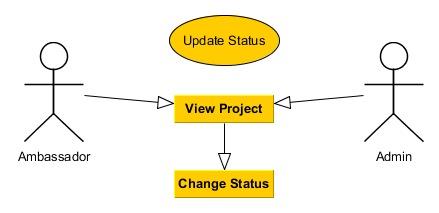
***7a. Submit an Idea***



***7b. Archive a Project***

******

***7c. Change Status***

******