**Vote Application based on iOS Detail Design Document**

Current Version: v0.2

Author: Jianghua Kuai, Shuai Zhao

Oct 26, 2013

Maintenance History

|  |  |  |  |
| --- | --- | --- | --- |
| **Author** | **Date** | **Content** | **Version** |
| **Jianghua Kuai**  **Jianghua Kuai** | Oct 26, 2013  Oct 27, 2013 | Build the Document  Add architecture design, data design and upgrade the structure of document. | V0.1  V0.2 |

Tabel of Content

Introduction 4

Purpose 4

Scope 4

Reference Material 4

System Overview 4

System Architecture 5

Architecture Design 5

Decomposition Description 6

Data Design 6

Human Interface Design 7

Overview of User Interface 7

Screen Images 8

Screen Object and Actions 8

Appendices 8

# Introduction

## Purpose

This software design document describes the architecture and system design of Vote. The main audiences of this document are developers. And this document is also a summary of all other detail design documents.

## Scope

This document will mainly contain the database design, class design for server and client, interface for the communication of client and server, and human interface design.

Some contents may lead to other document because it will be easier to maintain.

## Reference Material

1. Class Diagram for client and server
2. C/S communication Interface Design Document
3. Vote Database Design
4. Human Interface Design (stored on fluidui.com)

# System Overview

This project’s client application will implement on iOS platform which will use Objective-C and cocoa Framework, at the same time, this project will integrate Google Analytics into it. For communication with server, since we use the Plankton Server as our framework for server, so we’ll use Plankton API for iOS.

This project’s server application will implement by using C++ on CentOS, and the database is MySQL.

# System Architecture

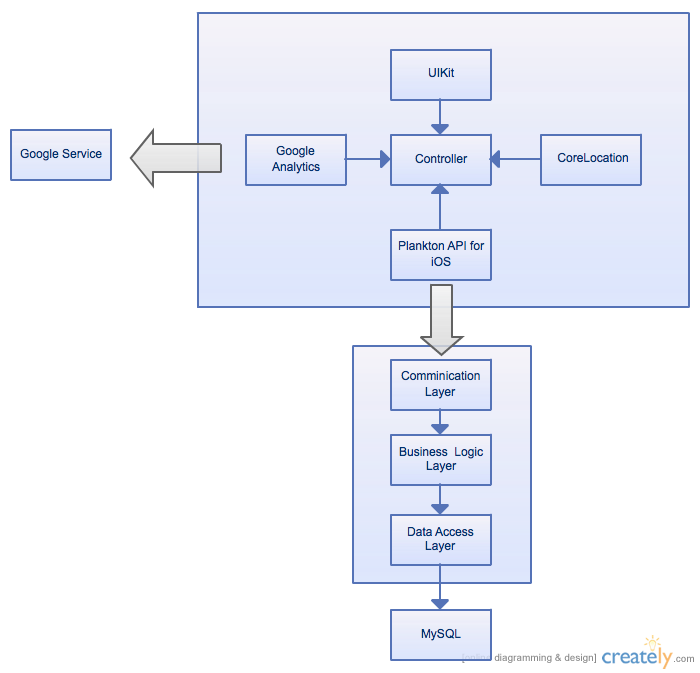
## Architecture Design

This project is implemented by using C/S architecture.

For the server part, Plankton Server is a standard three-tier server framework. We’ll focus on the business layer to build our business logics because the framework has already handled the communication and data access for us.

For the client part, cocoa framework is a very pure and standard MVC architecture so we can easily separate the data and operation.

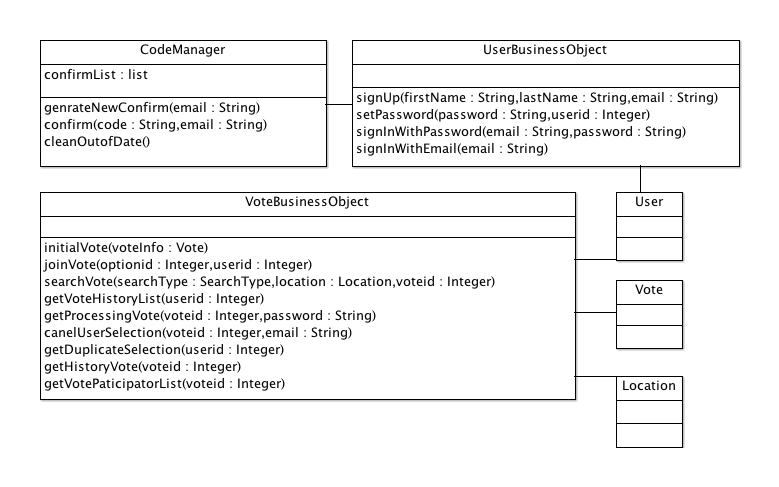
The diagram below shows the architecture of the project from a general view.



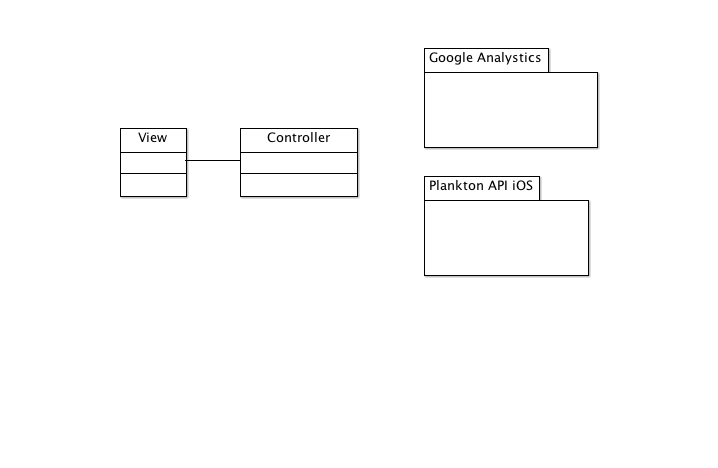
## Decomposition Description

The communication between client and server will format the data as the formation of ***Interface Document*** described.

Since we will only design the Business Logic Layer for the server, this document will only shows the class diagram of that layer. The diagram below is server’s Class Diagram.

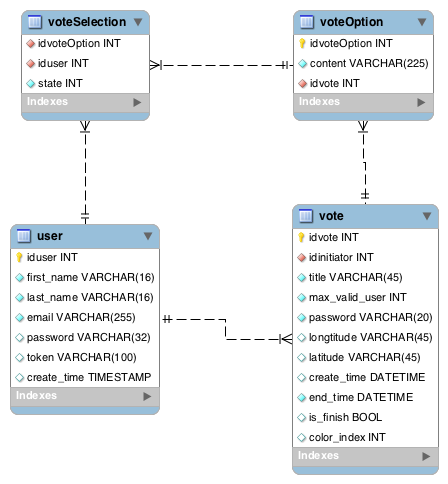


For client part, every controller will report the exception and user preference (if allowed) by using Google Analytics, and every controller will send request from Plankton API to server, so every single controller’s class diagram should looks like the image below. The detail relationship of each controller and view will show in human interface design part.



# Data Design

We use EER to describe our database design and structure.



# Human Interface Design

## Overview of User Interface

Describe the functionality of the system from the user’s perspective. Explain how the user will be able to use your system to complete all the expected features and the feedback information that will be displayed for the user.

## Screen Images

Display screenshots showing the interface from the user’s perspective. These can be hand­ drawn or you can use an automated drawing tool. Just make them as accurate as possible. (Graph paper works well.)

## Screen Object and Actions

A discussion of screen objects and actions associated with those objects.

# Appendices

DatabaseDesign.mwb (Works with MySQLWorkBench)

InterfaceDesign.xml (Works with Microsoft EXCEL)

Class Diagram (Works with ArgoUML)