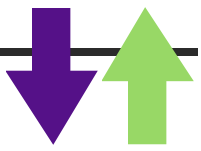
SOFTWARE DESIGN SPECIFICATION

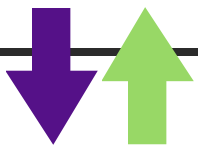
8/21/2013

Graduate Capstone

RECEIPT REWARDS

Table of Contents





1 Introduction

The purpose of this document is to provide description regarding the architecture and other design aspects of the system. The document will be used by the development team to develop the functionality. Clients can also use this to verify the agreed upon functionality.

1.1 Intended Audience

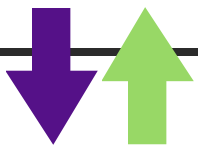
The document is intended for individuals of a medium-high technical or business background.

1.2 References

- http://web.cs.dal.ca/~hawkey/3130/SDS_outline.doc
- http://www.cs.utah.edu/~jamesj/ayb2005/docs/SDS_v2.htm
- http://www.se.rit.edu/~vdkrit/design/VDK-RIT_SDS.doc

1.3 Revision History

Name	Date	Reason For Change	Version
Andy Bottom	04/28/2013	Started compiling all architecture diagrams into this document	0.1
Andy Bottom	05/07/2013	Completed the finishing touches on this document	1.0
Andy Bottom	05/21/2013	Updated and added additional design decision including PCLs and caching	1.1



2 System Overview

2.1 Section Overview

This section is used for explanations on generically high-level overview of the entire system. This includes information regarding arts of this system.

2.2 General Constraints

The **Software Requirement Specification** for information about general constraints of the system.

2.3 Data Design

Please refer to the **Database Design Document** for a further analysis of the Data and Database Design.

2.4 Actors

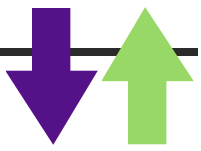
The actors are the main endpoints that utilize the functionality of the system.

2.4.1 Phone

The phones are the phone applications which the Users interact with to use the systems features.

2.4.2 Admin Website

The admin website is what the administrators interact with to manage the data in the system.



2.4.3 Database

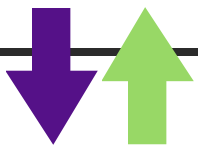
The database holds the information for the system and gets used by the other actors. All interactions with the database happen via the Web Services.

2.4.4 Web Services

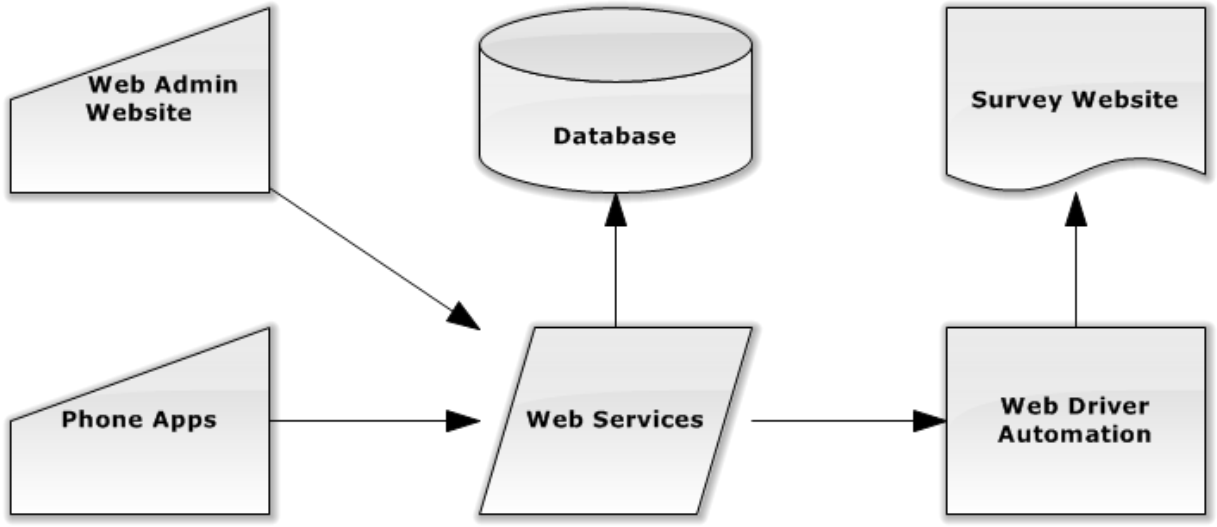
The Web Services is the way that both the Phone and Admin Website performs common logic and obtains the data. The Web Services acts as our data access layer for both the client applications to be able to get data from the database. The Web Services also acts as the kickoff for the Automation Web Driver process.

2.4.5 Survey Website

The Survey website is a pre-existing website that the company uses for the Satisfaction Surveys. Typically these websites are created by a company that offers to create these types of surveys and the origin company uses their services for this reason. It is important to note that the Company Survey Website is completely independent from our system and code base. The only interaction we have with the website is that it is the endpoint that is our Web Driver uses to perform the form automation on..



2.5 Domain Model



2.6 Goals and Guidelines

- A Large emphasis is put on convenience of the entire project. The goal is to provide an easier and more convenient way to take the surveys.
- Convenience will be done via very good Usability of the Phone Application.

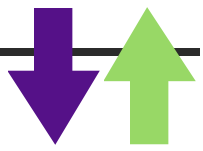
3 Architecture Design

3.1 Section Overview

The following section explains the architectural designs of the major system in the project.

3.2 Architectural Strategies

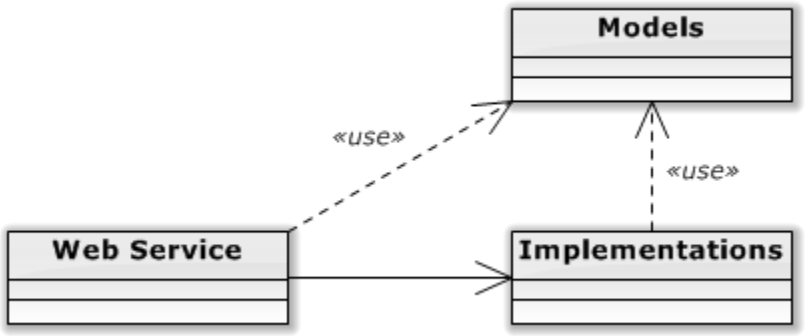
- Object Oriented Design is always the forefront of any application that I develop for. The idea of being able to get the most flexibility and reuse out of my code is always a goal that I strive for.
- The use of a DAL is implemented with the web services so that I can keep separation between the user devices and the rest of the system. It also provides reuse of all data calls.
- The automated process is also located inside the web services layer to keep abstraction between the devices and the process.
- The Web Admin site is a very object oriented approach and follows very closely along the lines of the database. The Web Admin is a Content Management System for the project.
- In the phone applications, we will create a PCL that will be referenced by the apps to incorporate a large reuse of the backend page logic.



3.3 System Architecture

3.3.1 Web Services

3.3.1.1 High-Level Class Diagram

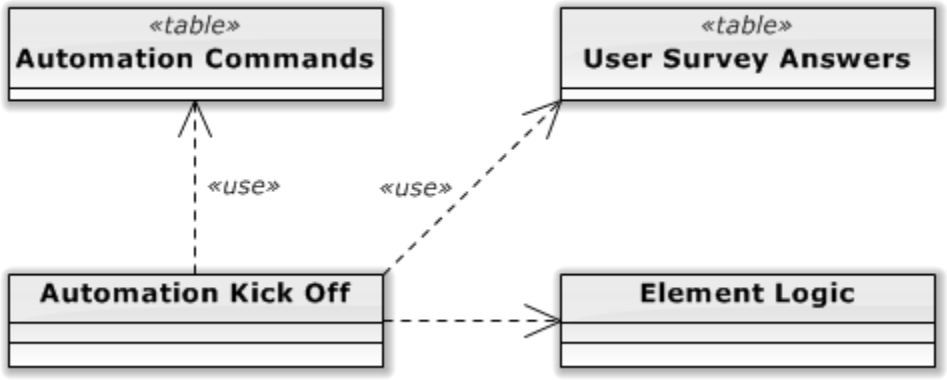


3.3.1.2 Class Diagram

@TODO: Implemented during start of Construction Phase

3.3.2 Automation Process

3.3.2.1 High-Level Class Diagram

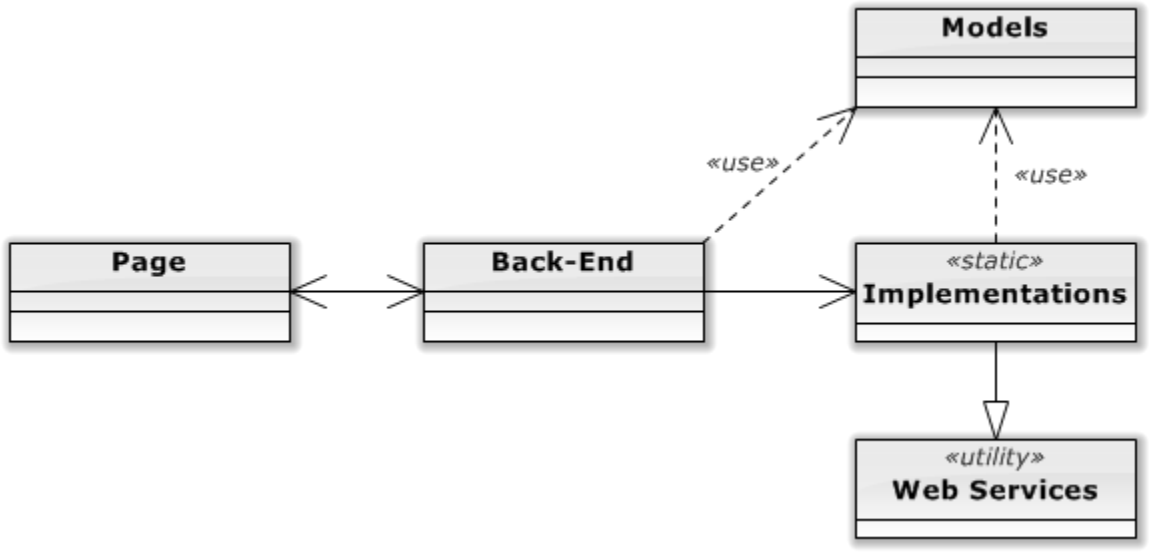


3.3.2.2 Class Diagram

@TODO: Implemented during start of Construction Phase

3.3.3 Admin CMS

3.3.3.1 High-Level Diagram



3.3.3.2 Class Diagram

@TODO: Implemented during start of Construction Phase

3.3.4 Phone Application

3.3.4.1 High Level Data Pattern

@TODO: MVVM Diagram

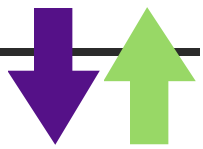
3.3.4.2 Class Diagram

@TODO: Add the class diagram

3.3.5 Reverse Proxy

3.3.5.1 High-Level Diagram

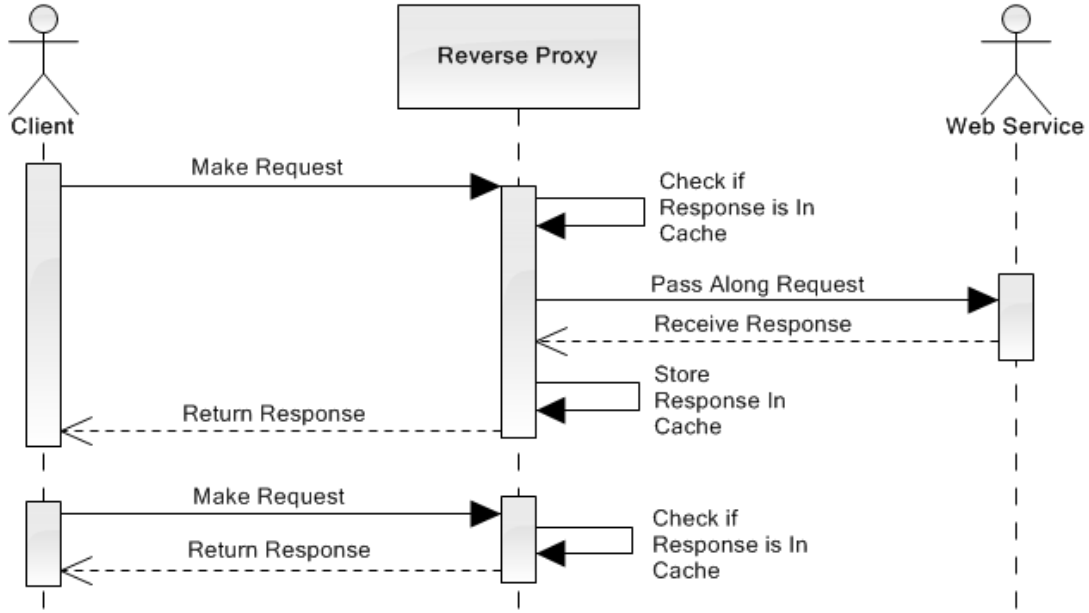
@TODO: Add the high-level diagram



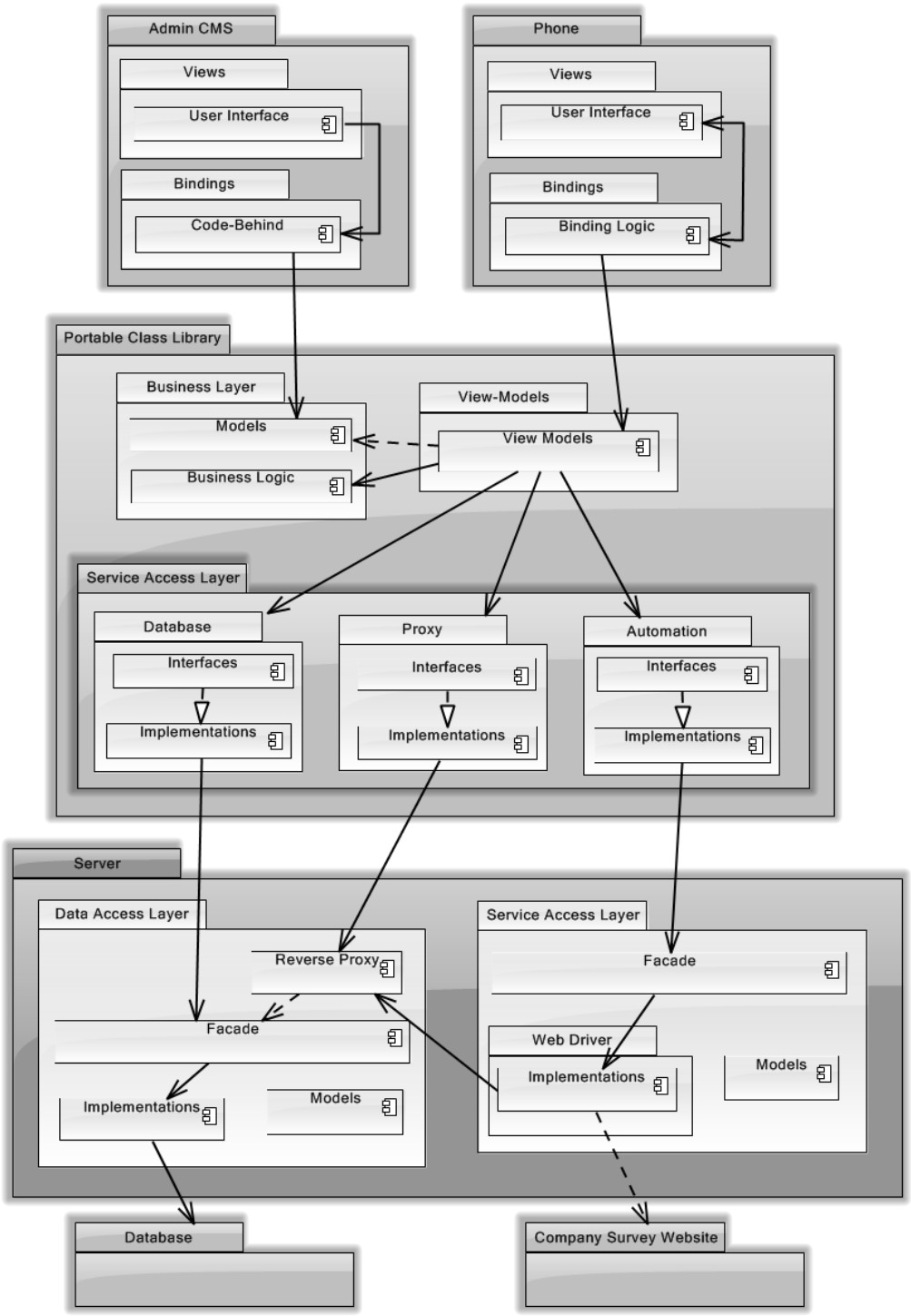
3.3.5.2 Class Diagram

@TODO: Add Class Diagram

3.3.5.3 Sequence Diagrams



3.4 Detailed System Design



4 User Interface Design

4.1 Section Overview

This section is used as a guide for designing the user interface for both the Admin CMS and the Phone Application

4.2 Interface Design Rules

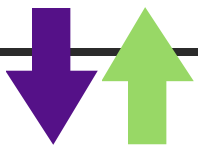
@TODO: Implemented during the start of the Construction Phase. Part of one of the iterations.

4.3 GUI Components

@TODO: Implemented during the start of the Construction Phase. Part of one of the iterations.

4.4 Detailed Description

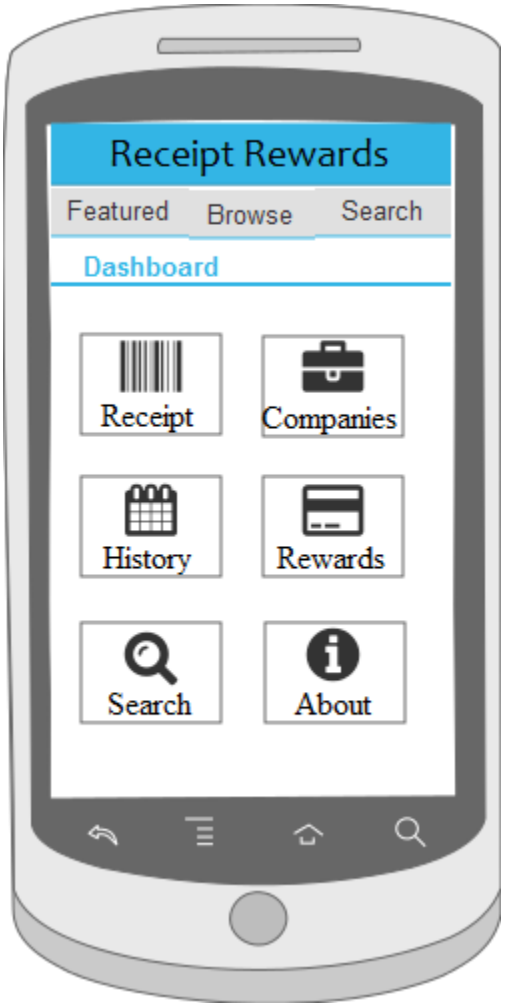
@TODO: Implemented during the start of the Construction Phase.



4.5 Prototypes and Wireframes

4.5.1 Phone Application

4.5.1.1 Dashboard / Main Menu

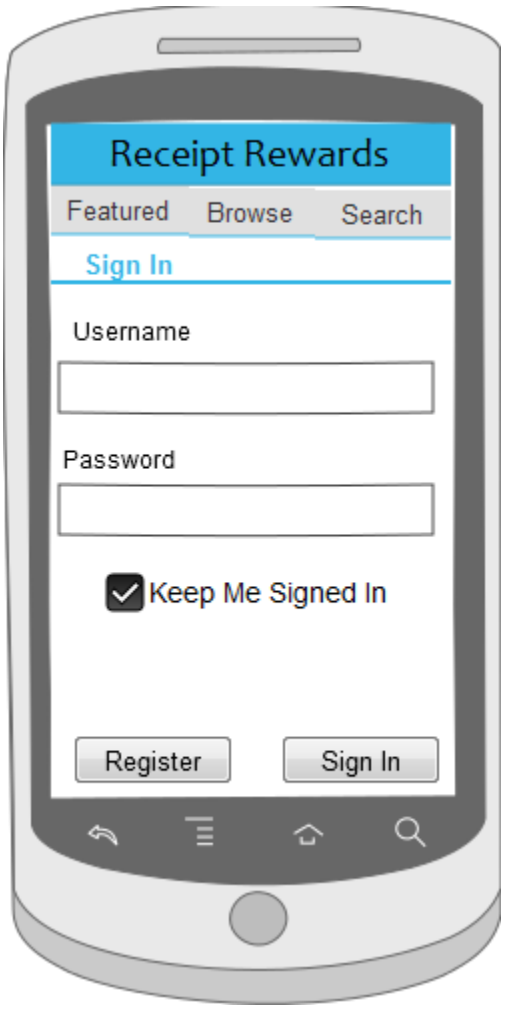


This is the dashboard or main menu page. From here, the use will be able to get to all the functionality of the system.

4.5.1.2 Register Page

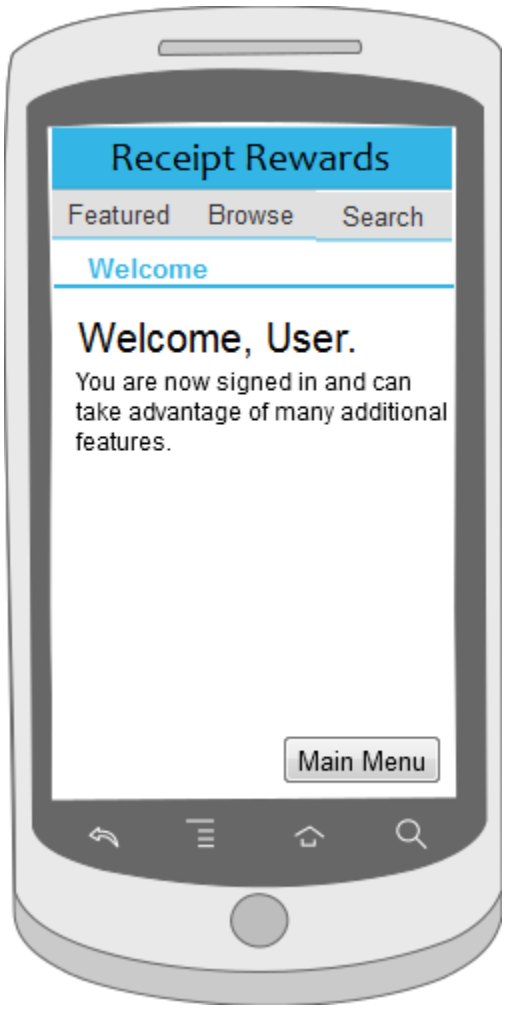
This is a register page so that an anonymous user can register and create a user account for them.

4.5.1.3 Login Page



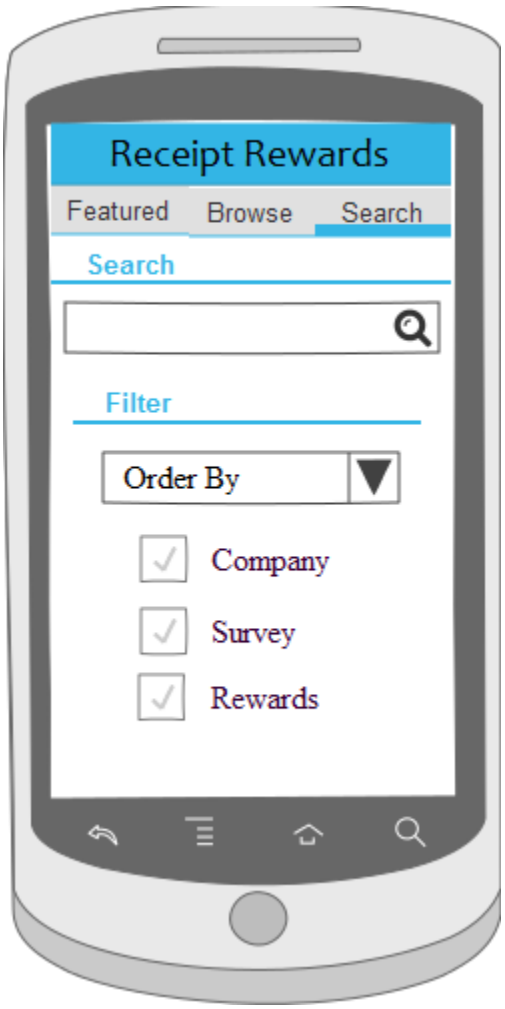
The login page provides the functionality for the user to be able to login to their user account.

4.5.1.4 Login Welcome



A welcome page for the user.

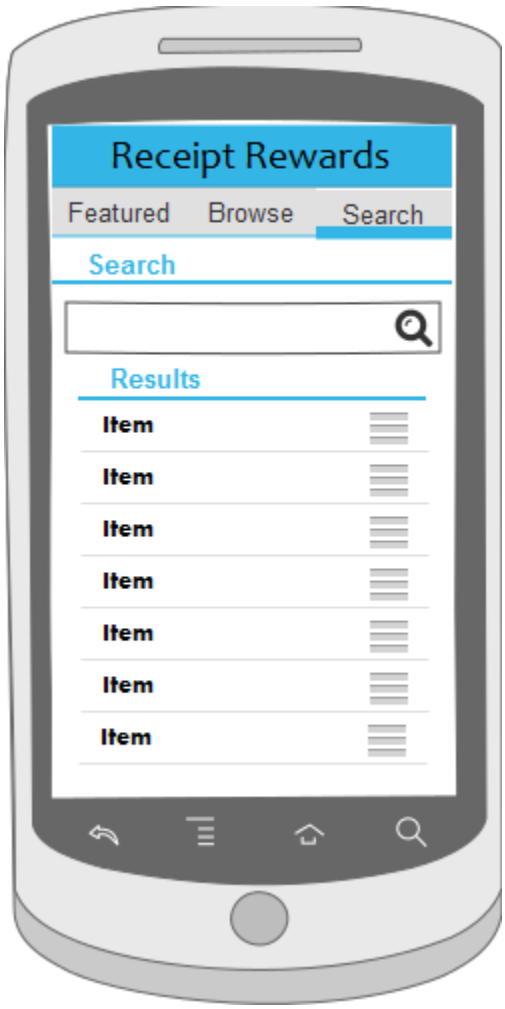
4.5.1.5 Search



The search gives the user a search box and advanced options to be able to search for companies and surveys.

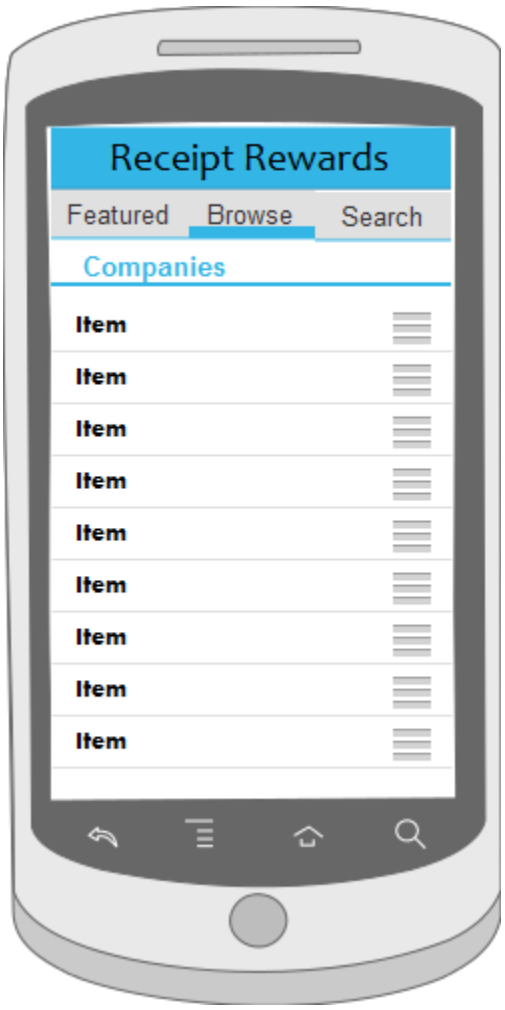
Upon searching, it will display the search result page.

4.5.1.6 Search Results



This is the search results page. The result will be of companies and surveys. They can select an item and go to either the Company Page or the Survey Page.

4.5.1.7 Company List



The company list page is used to display a list of all the companies to the user.

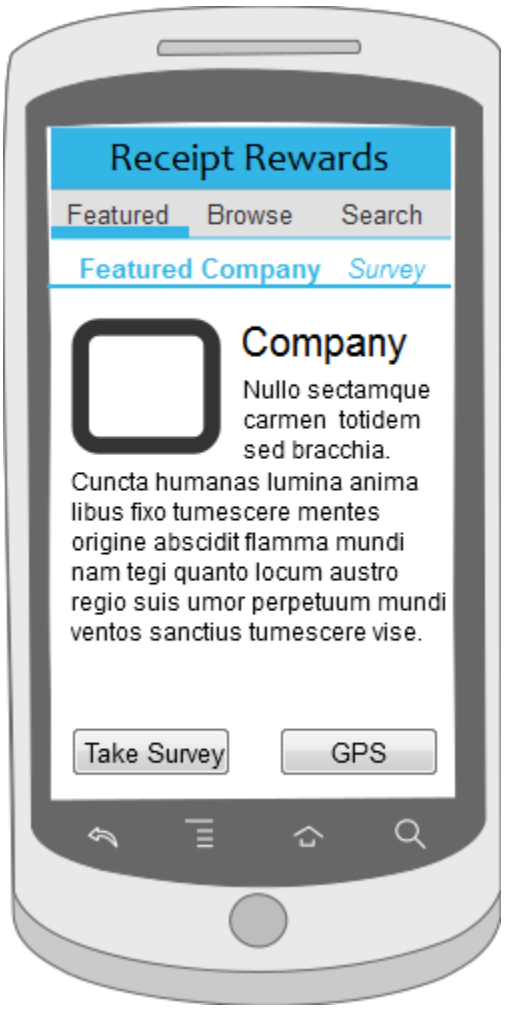
From here, a user can select a company to view which will display the company page to the user.

4.5.1.8 Company Page

The company page is the page which displays all the information about a company to the user.

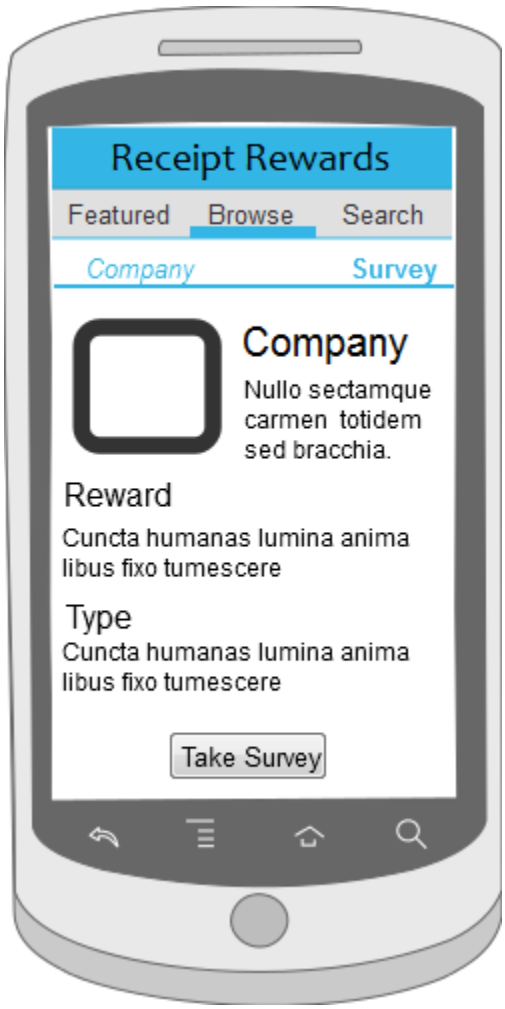
From this page, a user can also navigate the survey corresponding to that company.

4.5.1.9 Featured Company Page



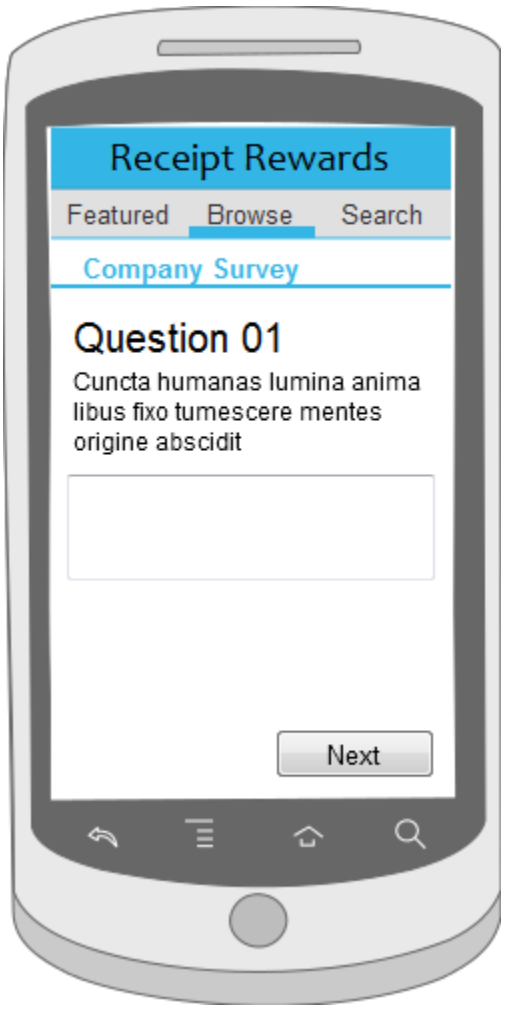
The featured company page is a place in the application where a company can pay to be listed in a special section. Kind of like a advertising spot for companies.

4.5.1.10 Survey Page



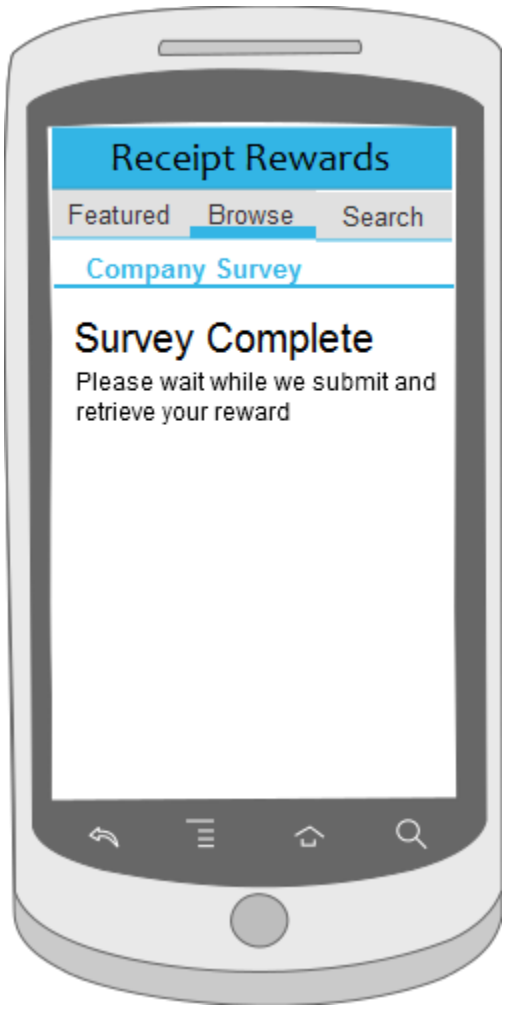
Survey page provides information regarding the survey and what the reward will be. From here, they can click take survey to move on to filling out the survey.

4.5.1.11 Survey



This is the page where they are actually taking the survey and the first question is displayed. They go through these screens until they finish.

4.5.1.12 Survey Complete



Once they finish taking the survey the will see the complete page. It is also on this page where they will receive information about how to get their reward.

5 Design decisions and tradeoffs

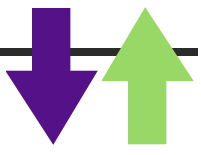
In this section, in-depth explanations containing reasoning for choices made in regard to the implementation and design of the system are explained. The options, pros and cons and overall choice and reason will be explained.

5.1 Web Services

Since web services are a vital part of our application, it was needed to be decided as to which time of web services would be implemented in the system. The choices were SOAP and RESTful. Below are a brief description of each and the choice made.

5.1.1 SOAP

Soap was the first type of web service created. I also have had experience with SOAP through an internship, so I already know the concept and how to implement them. The main part of SOAP is that it uses XML formatting and is relatively stricter in handling data and formatting. SOAP also tends to be more secure and is a better choice when using very highly sensitive data. Unfortunately, the data in our system will not be immensely sensitive.

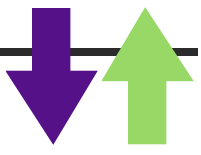


5.1.2 RESTful

After researching more about Web Services, I came upon RESTful. It seemed to me that currently, that RESTful web services seemed to be the more trendy design choice for Web Services. I thought it would be a good idea to gain experience and learn how to implement them. The conceptual flow of RESTful is extremely similar to the SOAP Web Services. The advantages for RESTful are that the data can be sent in as XML or as JSON. Also, the formatting of the system of RESTful tends to be simpler and more flexible. This causes the downside of it being less secure and not a good choice for handling highly sensitive data. Fortunately there are still other ways several designs to help make RESTful web services.. Also, one last note, which was more of a deciding factor, is that the ability to handle the data as JSON would be advantageous for mobile development because JSON is less labor intensive in compiling and parsing than the XML.

5.1.3 Decision

After weighing the pros and cons, it was decided that RESTful Web Services would the best fit for the less labor intensive fashion of dealing with JSON and that RESTful seems to be the go to choice in the future.

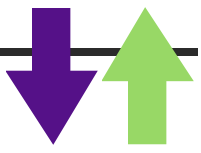


5.2 Mobile Cross-Platform Development

There was a need to develop the phone application to be cross-platform. When proposed with this problem there are relatively two directions you can take. You can implement a Web Wrapper Application or use a framework to develop the application, such as Xamarin. Below I discuss the advantages of each and the choice made.

5.2.1 Hybrid App

A hybrid app is essential a fake app. By this I mean is that it is an application that is written with HTML, CSS and JavaScript. Essentially it is a mobile website that is then "wrapped" inside a phone application so it can be "installed" on a phone. But it isn't interpretive like a native app, but instead uses the browser rendering technology to display. The advantage of this is that Hybrid Apps are highly multi-platform friendly as it uses Web Standards and not the phone to render. The disadvantage of this is that it doesn't utilize the true functionality of a phone to the extent that native apps can. Also, another major disadvantage is that the UI of the app doesn't look native. To explain further, each smartphone OS has its own unique look. Apple has the glossy rounded look where as Windows Phone has its Metro look. A hybrid app that is developed for both of the platforms will function properly, but the User Experience of the app is very different than what the user is used to. Due to the nature of how "Web Apps" look and feel, it is evident that the application was not created for that OS. Thus, they basically don't fit in.



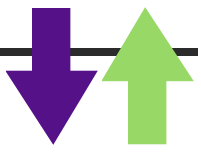
5.2.2 Xamarin

The other option is to create a Native Application. Unfortunately, all the OS use a different programming language to develop a application. This realization makes it impossible to create an application that is cross-platform. There would be no reuse of code and the developers would be stuck maintaining 3 separate code bases. This is why a use of a third-party framework can be used to create an application for all three OS but in the same language. The standout option that I selected was Xamarin.

Xamarin is a Framework that allows developers to create a .Net Native Application. The framework will then compile the application to a Native App of all the OS. The advantage of having the native app is that you can utilize a very large potential of the phone's functionality and also, the applications will be displayed using Native UI elements of the OS, thus the applications will feel very natural to the user. By using Xamarin, the UI and the Back-end Logic can be separated. By implementing in this fashion, the only customized code that needs to be developed in the UI for each operating system. The Logic on the backend will can all be reused between all OS. The disadvantage is that you will need to purchase a license to utilize Xamarin.

5.2.3 Choice

After weighing all the options, I decided to go with Xamarin to develop cross-platform Native Applications. The ability to have a native experience for the user is important and the ability to achieve this with a high percentage of code-reuse is undeniably beneficial.



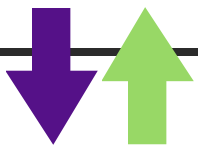
5.3 Mobile Cross-Platform Design Pattern

5.3.1 Model View View-Model (MVVM)

I initially was going to develop the apps following the MVVM pattern. To implement it this way, I found a framework which is very similar to Xamarin called MVVM Cross (MVX) which allows the user to user the MVX libraries and have a core library which could very very easily have large code reuse for all mobile operating systems. At the beginning of my phone development process, I looked very strongly into the possibility of using MVX as my framework and it looked like it would be possible. However, tricky aspect of my app would be the need to have a dynamic form displayed for the surveys. I looked at it at all angles and determined that where MVX is currently with their version 3 release, it does not have the correct capabilities yet to successfully implement all aspects of my application, thus the design decision to continue with the Xamarin Framework was made. However, I have immensely high hopes and expectations that MVX will be a great choice for app development in the future.

5.3.2 Portable Class Library (PCL)

A portable class library is essential a package of classes that will be used by all OS versions of the project and will contain all the unified functionality of the system. Unfortunately, at this moment, PCLs aren't supported by Xamarin directly, (but will be in the near future), so instead, I will be making the PCL package, and just copy the package into each project. Inside of the PCL, will be business logic, data access object for accessing the database (phone, and the remote) and a service access layer (for automation and remote database.) Models will also be defined in the PCL.



5.4 Caching

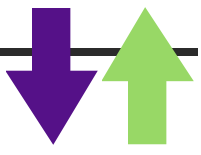
Since there will be a very heavy dependence on the web services in my application, the need to keep the web services performing very quickly and effectively will be vital. So we will help to lessen the load that actually be hitting the web services by implementing caching layers for content that is requested often.

5.4.1 Reverse Proxy

The purpose of a reverse proxy is to essentially have this proxy on our server side and have all the requests feed through it. By doing this, the one client will request from the web services and cache the content. Thus if 5 people request to see a list of companies, the first person will go through the reverse proxy and then hit the web services, the remaining four will call to the reverse proxy, but since the content would be cached after the first time, the web services will not be hit, and the cached content will be returned. Again, this won't be used for Admin Content, or for Updated of information. But mostly will be implemented for the most utilized calls to the system.

5.4.2 Phone Caching

There is no phone caching done with our phone application. The only caching that is done is the default caching of each operating system.

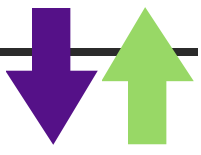


5.5 Database Access

The next decision was how the phone applications will access the database on the server. After researching, it has been advised by many that for security and performance reasons, having the application directly make a connection to a database is not a good idea.

5.1 Data Access Layer through Web Services

We will be implementing a Data Access Layer through which all database calls are made. This way the database side will be separated from the rest of the code bases. To access the data layer, it will be hooked up to web services that can be called and depending on the request parameters, can perform any CRUD functionality. This way the Admin CMS and the phone applications can all get to the database in a way that is secure and reused.



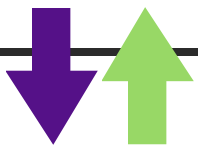
6 Pseudocode for System

Below are template files to be used to view the syntax and design to be followed in files of a similar nature.

6.1 Admin CMS

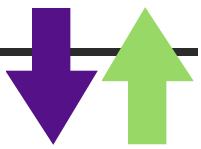
6.1.1 Element Manager Page (Back-End)

```
using ReceiptReward;
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class Entity_Manager : System.Web.UI.Page {
    // Global Variables
    private int surveyId = 0;
    private Survey survey = null;
    private Company company = null;
    private string type = "survey";

    protected void Page_Load(object sender, EventArgs e) {

        // Call Methods
        getVariables();
        if (! this.IsPostBack) {
            getTitle();
            getList();
        }
    }


    /**
     * Get The Data for the page
     */
    private void getVariables() {
        if (Request.QueryString["surveyId"] != null) { surveyId = int.Parse(Request.QueryString["surveyId"]); }
        survey = SurveyImpl.getBySurveyId(surveyId);
        company = CompanyImpl.getById(survey.companyId);
        lblScripts.Text = "<script type=\"text/javascript\"> var bigId = " + surveyId + "; bigType = \"" + type + "\";</script>";
    }


    /**
     * Get the title of the page
     */
    private void getTitle() {
        lblTitle.Text = company.name + ": " + survey.name;
        txtEditName.Text = survey.name;
    }

```

```
45
46
47      /**
48       * Creates a new element
49       */
50      protected void Add_Click(Object sender, EventArgs e) {
51          Revision revision = new Revision();
52          revision.revisionId = 0;
53          revision.active = chkAddActive.Checked;
54          revision.automation = null;
55          revision.dateCreated = DateTime.Now;
56          revision.form = null;
57          revision.revisionNumber = txtAddRevisionNumber.Text;
58          revision.surveyId = survey.surveyId;
59          RevisionImpl.create(revision);
60      }
61
62
63      /**
64       * Edit element
65       */
66      protected void Edit_Click(Object sender, EventArgs e) {
67          Survey surveyEdit = new Survey();
68          surveyEdit.surveyId = int.Parse(hdnSaveId.Value);
69          surveyEdit.name = txtEditName.Text;
70          SurveyImpl.update(surveyEdit);
71      }
72
73
74      /**
75       * Delete element
76       */
77      protected void Delete_Click(Object sender, EventArgs e) {
78          if (hdnDeleteType.Value == "survey") {
79              SurveyImpl.removeBySurveyId(int.Parse(hdnDeleteId.Value));
80          } else if (hdnDeleteType.Value == "revision") {
81              RevisionImpl.removeByRevisionId(int.Parse(hdnDeleteId.Value));
82          }
83      }
```

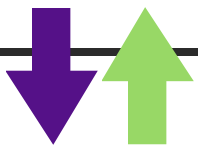
```
85
86      public void getList() {
87          // Instantiate Variables
88          string html = "";
89          int counter = 0, childrenCount = 0;
90          List<Revision> list = RevisionImpl.getBySurveyId(survey.surveyId);
91
92          // Iterate through list
93          foreach (Revision item in list) {
94
95              // Instantiate Variables
96              childrenCount = 0; //Survey.containsNumOfQuestionGroups(surveyID);
97
98              html += "" +
99                  "<div style='position: absolute; top: " + (counter * 25) + "px; left: 0px; height: 50px; width: 960px;'>" +
100                     "<div class='TableElementDiv' style='width: 200px;'>" + CommonUtils.trim(item.revisionNumber, 20) + "</div>" +
101                     "<div class='TableElementDiv' style='width: 100px;'>" + item.active + "</div>" +
102                     "<div class='TableElementDiv' style='width: 200px;'>" + item.dateCreated.ToShortDateString() + "</div>" +
103                     "<div class='TableElementDiv' style='width: 200px;'>" +
104                         "<a href='Revision.aspx?revisionId=" + item.revisionId + "'>Edit</a> | " +
105                         "<a class='isALink' onclick='$(\"#dlgDelete\").dialog(\"open\"); $(\"#hdnDeleteId\").val(\"" + item.revisionId + "\");
                            $(\"#hdnDeleteType\").val(\"" + "revision" + "\");'>Delete</a>" +
106                     "</div>" +
107                 "</div>";
108             counter++;
109         }
110         // Add More to the text
111         html = "<div id='movingObjectsHolder' style='position: relative; width: 960px; height: " + ((counter - 1) * 25) + "px;'>" + html + "</div>";
112         html = "" +
113                 "<div class='TableElementDiv' style='width: 200px;'><strong>Revision Number</strong></div>" +
114                 "<div class='TableElementDiv' style='width: 100px;'><strong>Active</strong></div>" +
115                 "<div class='TableElementDiv' style='width: 200px;'><strong>Date Created</strong></div>" +
116                 "<div class='TableElementDiv' style='width: 200px;'><strong>Options</strong></div>" + html;
117
118         // Display Text
119         lblList.Text = html;
120     }
121
122 }
```

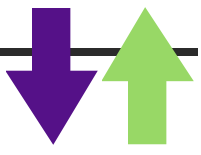
6.1.2 Element Implementation

```
using System;
using System.Collections.Generic;
using System.Web;
using System.Configuration;
using System.Data.SqlClient;
using System.Net;
using System.IO;
using Newtonsoft.Json.Linq;
using System.Diagnostics;


namespace ReceiptReward
{

    public class SurveyImpl
    {
        // Base URL
        private const string BASE_URL = "http://140.104.69.94:8080/RestfulWebService/bti";
        private const string DIRECTORY = SurveyImpl.BASE_URL + "/Survey";

        // GET Methods
        private const string GET_ALL = SurveyImpl.DIRECTORY + "/All";
        private const string GET_BY_COMPANY_ID = SurveyImpl.DIRECTORY + "/ById/Company";
        private const string GET_BY_SURVEY_ID = SurveyImpl.DIRECTORY + "/ById/Survey";

        // POST Methods
        private const string CREATE = SurveyImpl.DIRECTORY + "/Create";

        // UPDATE Methods
        private const string UPDATE = SurveyImpl.DIRECTORY + "/Update";

        // DELETE Methods
        private const string REMOVE_BY_COMPANY_ID = SurveyImpl.DIRECTORY + "/ById/Company";
        private const string REMOVE_BY_SURVEY_ID = SurveyImpl.DIRECTORY + "/ById/Survey";


        /**
         * Obtains a list of all surveys in the database
         * Calls webservice
         * @return A list of surveys objects
         */
        public static List<Survey> getAll()
        {
            // Instantiate Variables
            string json = "";
```

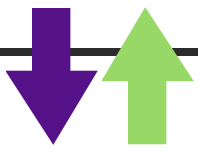
```
            string json = "";
            List<Survey> surveys = new List<Survey>();

            // Set up Request
            HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create(SurveyImpl.GET_ALL);
            httpWebRequest.ContentType = "text/json";
            httpWebRequest.Method = "GET";

            // Get the Response
            HttpWebResponse httpResponse = (HttpWebResponse)httpWebRequest.GetResponse();
            using (StreamReader streamReader = new StreamReader(httpResponse.GetResponseStream()))
            {
                json = "{\"data\":" + streamReader.ReadToEnd() + "}";
            }

            // Convert JSON to object
            JObject jObject = JObject.Parse(json);
            JToken jCodes = jObject["data"];
            foreach (JToken jCode in jCodes.Children())
            {
                Survey survey = new Survey();
                survey.surveyId = (int)jCode["surveyId"];
                survey.companyId = (int)jCode["companyId"];
                survey.name = (string)jCode["name"];
                surveys.Add(survey);
            }

            // Return
            return surveys;
        }


        /**
         * Obtains a Survey by Company Id from the database
         * Calls webservice
         * @return A Survey object
         */
        public static List<Survey> getByCompanyId(int companyId)
        {
            // Instantiate Variables
            string json = "";
            List<Survey> surveys = new List<Survey>();

            // Set up Request
            HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create(SurveyImpl.GET_BY_COMPANY_ID + "/" + companyId);
```

```
            HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create(SurveyImpl.GET_BY_COMPANY_ID + "/" + companyId);
            httpWebRequest.ContentType = "text/json";
            httpWebRequest.Method = "GET";

            // Get the Response
            HttpWebResponse httpResponse = (HttpWebResponse)httpWebRequest.GetResponse();
            using (StreamReader streamReader = new StreamReader(httpResponse.GetResponseStream()))
            {
                json = "{\"data\":" + streamReader.ReadToEnd() + "}";
            }

            // Convert JSON to object
            JObject jObject = JObject.Parse(json);
            JToken jCodes = jObject["data"];
            foreach (JToken jCode in jCodes.Children())
            {
                Survey survey = new Survey();
                survey.surveyId = (int)jCode["surveyId"];
                survey.companyId = (int)jCode["companyId"];
                survey.name = (string)jCode["name"];
                surveys.Add(survey);
            }

            // Return
            return surveys;
        }


        /**
         * Obtains a Survey by Id from the database
         * Calls webservice
         * @return A Survey object
         */
        public static Survey getBySurveyId(int surveyId)
        {
            // Instantiate Variables
            string json = "";
            Survey survey = new Survey();

            // Set up Request
            HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create(SurveyImpl.GET_BY_SURVEY_ID + "/" + surveyId);
            httpWebRequest.ContentType = "text/json";
            httpWebRequest.Method = "GET";

            // Get the Response
            // Get the Response
            HttpWebResponse httpResponse = (HttpWebResponse)httpWebRequest.GetResponse();
            using (StreamReader streamReader = new StreamReader(httpResponse.GetResponseStream()))
            {
                json = "{\"data\":" + streamReader.ReadToEnd() + "}";
            }

            // Convert JSON to object
            JObject jObject = JObject.Parse(json);
            JToken jCodes = jObject["data"];
            foreach (JToken jCode in jCodes.Children())
            {
                survey.surveyId = (int)jCode["surveyId"];
                survey.companyId = (int)jCode["companyId"];
                survey.name = (string)jCode["name"];
            }

            // Return
            return survey;
        }


        /**
         * Create a new entry in the Database
         * Calls webservice
         */
        public static void create(Survey survey)
        {
            // Set up Request
            HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create(SurveyImpl.CREATE);
            httpWebRequest.ContentType = "application/json";
            httpWebRequest.Method = "POST";

            // Send Request Data
            using (StreamWriter streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))
            {
                string json = "{\"surveyId\":\"" + survey.surveyId + "\",\"companyId\":\"" + survey.companyId + "\",\"name\":\"" + survey.name + "\"}";
                Debug.WriteLine(json);
                streamWriter.Write(json);
                streamWriter.Flush();
                streamWriter.Close();
            }

            // Get the Response
            HttpWebResponse httpResponse = (HttpWebResponse)httpWebRequest.GetResponse();
```

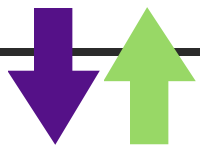
```
            HttpWebResponse httpResponse = (HttpWebResponse)httpWebRequest.GetResponse();
        }


        /**
         * Update entry in the Database
         * Calls webservice
         */
        public static void update(Survey survey)
        {
            // Set up Request
            HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create(SurveyImpl.UPDATE);
            httpWebRequest.ContentType = "application/json";
            httpWebRequest.Method = "PUT";

            // Send Request Data
            using (StreamWriter streamWriter = new StreamWriter(httpWebRequest.GetRequestStream()))
            {
                string json = "{\"surveyId\":\"" + survey.surveyId + "\",\"name\":\"" + survey.name + "\"}";
                Debug.WriteLine(json);
                streamWriter.Write(json);
                streamWriter.Flush();
                streamWriter.Close();
            }

            // Get the Response
            HttpWebResponse httpResponse = (HttpWebResponse)httpWebRequest.GetResponse();
        }


        /**
         * Removes an entry in the Database by Company Id
         * Calls webservice
         */
        public static void removeByCompanyId(int companyId)
        {
            // Set up Request
            HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create(SurveyImpl.REMOVE_BY_COMPANY_ID + "/" + companyId);
            httpWebRequest.ContentType = "application/json";
            httpWebRequest.Method = "DELETE";

            // Get the Response
            HttpWebResponse httpResponse = (HttpWebResponse)httpWebRequest.GetResponse();
        }
        /**
         * Removes an entry in the Database by Survey Id
         * Calls webservice
         */
        public static void removeBySurveyId(int surveyId)
        {
            // Set up Request
            HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create(SurveyImpl.REMOVE_BY_SURVEY_ID + "/" + surveyId);
            httpWebRequest.ContentType = "application/json";
            httpWebRequest.Method = "DELETE";

            // Get the Response
            HttpWebResponse httpResponse = (HttpWebResponse)httpWebRequest.GetResponse();
        }


        /**
         * Get the breadcrumbs
         */
        static public string getBreadcrumbs(int id)
        {
            // Instantiate Variables
            string tempStr = "";
            Survey survey = SurveyImpl.getBySurveyId(id);
            Company company = CompanyImpl.getById(survey.companyId);

            // Survey
            tempStr = "<a class='breadcrumbCurrent' href='Survey.aspx?surveyId=" + survey.surveyId + "'>" + survey.name + "</a>" + tempStr;

            // Company
            tempStr = "<a class='breadcrumbNotCurrent' href='Company.aspx?companyId=" + company.companyId + "'>" + company.name + "</a> >> " + tempStr;

            // Homepage
            tempStr = "<a class='breadcrumbNotCurrent' href='Index.aspx'>Home</a> >> " + tempStr;

            // Output Links
            return tempStr;
        }

    }
}
```

6.2 Web Services

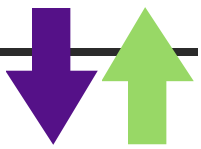
6.2.1 Service

```
package com.bti.ws.controller;

import com.bti.ws.implementations.SurveyImpl;
import com.bti.ws.model.Survey;
import java.util.List;
import org.springframework.http.HttpStatus;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.bind.annotation.ResponseStatus;

@Controller
@RequestMapping("/Survey")
public class SurveyController {


    @RequestMapping(value = "/All", method = RequestMethod.GET)
    @ResponseBody
    List<Survey> getAll() {
        return SurveyImpl.findAll();
    }

    @RequestMapping(value = "/ById/Company/{companyId}", method = RequestMethod.GET)
    @ResponseBody
    List<Survey> getByCompanyId(@PathVariable("companyId") int companyId) {
        return SurveyImpl.findByCompanyId(companyId);
    }

    @RequestMapping(value = "/ById/Survey/{surveyId}", method = RequestMethod.GET)
    @ResponseBody
    List<Survey> getBySurveyId(@PathVariable("surveyId") int surveyId) {
        return SurveyImpl.findBySurveyId(surveyId);
    }

    @RequestMapping(value = "/Create", method = RequestMethod.POST)
    @ResponseStatus(HttpStatus.OK)
    void create(@RequestBody Survey survey) {
        SurveyImpl.create(survey);
    }

    @RequestMapping(value = "/Update", method = RequestMethod.PUT)
    @ResponseStatus(HttpStatus.OK)
    void update(@RequestBody Survey survey) {
        SurveyImpl.update(survey);
    }

    @RequestMapping(value = "/ById/Survey/{surveyId}", method = RequestMethod.DELETE)
    @ResponseStatus(HttpStatus.OK)
    void removeBySurveyId(@PathVariable("companyId") int surveyId) {
        SurveyImpl.deleteBySurveyId(surveyId);
    }

    @RequestMapping(value = "/ById/Company/{companyId}", method = RequestMethod.DELETE)
    @ResponseStatus(HttpStatus.OK)
    void removeByCompanyId(@PathVariable("companyId") int companyId) {
        SurveyImpl.deleteByCompanyId(companyId);
    }

    @ExceptionHandler(NotFoundException.class)
    @ResponseStatus(value = HttpStatus.NOT_FOUND, reason="Code Type not found")
    void handleNotFound(NotFoundException exc) {}

    class NotFoundException extends RuntimeException {
        private static final long serialVersionUID = 1L;
    }
}

```

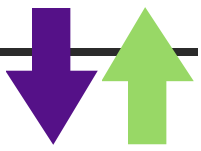



6.2.2 Implementation

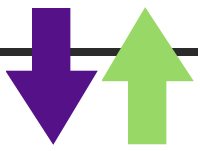
```java
package com.bti.ws.implementations;

import com.bti.ws.common.DBConnection;
import com.bti.ws.model.Survey;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;

public class SurveyImpl {

    private static final String FIND_ALL = "SELECT * FROM survey;";
    private static final String FIND_BY_SURVEY_ID = "SELECT * FROM survey WHERE survey_id = ?;";
    private static final String FIND_BY_COMPANY_ID = "SELECT * FROM survey WHERE company_id = ?;";
    private static final String FIND_NEXT_ID = "SELECT TOP 1 survey_id FROM survey ORDER BY survey_id DESC;";
    private static final String CREATE = "INSERT INTO survey (survey_id, company_id, name) VALUES (?, ?, ?);";
    private static final String UPDATE = "UPDATE survey SET name = ? WHERE survey_id = ?;";
    private static final String DELETE_BY_SURVEY_ID = "DELETE FROM survey WHERE survey_id = ?;";
    private static final String DELETE_BY_COMPANY_ID = "DELETE FROM survey WHERE company_id = ?;";


    /***** Find Methods *****/
    public static List<Survey> findAll() {
        // Instantiate Variables
        Connection con = null;
        PreparedStatement ps = null;
        ResultSet rs = null;
        List<Survey> surveys = new ArrayList<Survey>();

        try { // Get List of All
            con = DBConnection.getConnection();
            ps = con.prepareStatement(SurveyImpl.FIND_ALL);
            rs = ps.executeQuery();
            while(rs.next()) {
                surveys.add(new Survey(rs.getInt("survey_id"), rs.getInt("company_id"), rs.getString("name")));
            }
            rs.close();
            ps.close();
        } catch(Exception ex) {
            System.out.println("Error: " + ex.getMessage());
            // @TODO: Handle Exception
        } finally {
            try {
                if(con != null) {
                    con.close();
                    con = null;
                }
            } catch(Exception ex){}
        }

        // Return
        return surveys;
    }

    public static List<Survey> findByCompanyId(int companyId) {
        // Instantiate Variables
        Connection con = null;
        PreparedStatement ps = null;
        ResultSet rs = null;
        List<Survey> surveys = new ArrayList<Survey>();
```

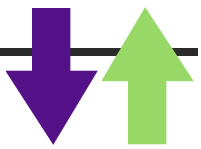
```
63          try { // Get List By Id
64              con = DBConnection.getConnection();
65              ps = con.prepareStatement(SurveyImpl.FIND_BY_COMPANY_ID);
66              ps.setInt(1, companyId);
67              rs = ps.executeQuery();
68              while(rs.next()) {
69                  surveys.add(new Survey(rs.getInt("survey_id"), rs.getInt("company_id"), rs.getString("name")));
70              }
71              rs.close();
72              ps.close();
73          } catch(Exception ex) {
74              System.out.println("Error: " + ex.getMessage());
75              // @TODO: Handle Exception
76          } finally {
77              try {
78                  if(con != null) {
79                      con.close();
80                      con = null;
81                  }
82              } catch(Exception ex){}
83          }
84
85          // Return
86          return surveys;
87      }
88
89      public static List<Survey> findBySurveyId(int surveyId) {
90          // Instantiate Variables
91          Connection con = null;
92          PreparedStatement ps = null;
93          ResultSet rs = null;
94          List<Survey> surveys = new ArrayList<Survey>();
95
96          try { // Get List By Id
97              con = DBConnection.getConnection();
98              ps = con.prepareStatement(SurveyImpl.FIND_BY_SURVEY_ID);
99              ps.setInt(1, surveyId);
100             rs = ps.executeQuery();
101             while(rs.next()) {
102                 surveys.add(new Survey(rs.getInt("survey_id"), rs.getInt("company_id"), rs.getString("name")));
103             }
104             rs.close();
105             ps.close();
106         } catch(Exception ex) {
107             System.out.println("Error: " + ex.getMessage());
108             // @TODO: Handle Exception
109         } finally {
110             try {
111                 if(con != null) {
112                     con.close();
113                     con = null;
114                 }
115             } catch(Exception ex){}
116         }
117
118         // Return
119         return surveys;
120     }
121
```

```
121
122     public static int findNextId() {
123         // Instantiate Variables
124         Connection con = null;
125         PreparedStatement ps = null;
126         ResultSet rs = null;
127         int surveyId = 1;
128
129         try { // Get All the Code Types
130             con = DBConnection.getConnection();
131             ps = con.prepareStatement(SurveyImpl.FIND_NEXT_ID);
132             rs = ps.executeQuery();
133             while(rs.next()) {
134                 surveyId = rs.getInt("survey_id") + 1;
135             }
136             rs.close();
137             ps.close();
138         } catch(Exception ex) {
139             System.out.println("Error");
140             // @TODO: Handle Exception
141         } finally {
142             try {
143                 if(con != null) {
144                     con.close();
145                     con = null;
146                 }
147             } catch(Exception ex){}
148         }
149
150         // Return
151         return surveyId;
152     }
153
154
155     /***** Create Methods *****/
156     public static void create(Survey survey) {
157         // Instantiate Variables
158         Connection con = null;
159         PreparedStatement ps = null;
160
161         try { // Set ID
162             survey.setSurveyId(SurveyImpl.findNextId());
163
164             // Create New Entry
165             con = DBConnection.getConnection();
166             ps = con.prepareStatement(SurveyImpl.CREATE);
167             ps.setInt(1, survey.getSurveyId());
168             ps.setInt(2, survey.getCompanyId());
169             ps.setString(3, survey.getName());
170             ps.executeUpdate();
171             ps.close();
172         } catch(Exception ex) {
173             System.out.println("Error: " + ex.getMessage());
174             // @TODO: Handle Exception
175         } finally {
176             try {
177                 if(con != null) {
178                     con.close();
179                     con = null;
180                 }
181             } catch(Exception ex){}
```

```
181             } catch(Exception ex){}
182         }
183     }
184
185
186     /***** Update Methods *****/
187     public static void update(Survey survey) {
188         // Instantiate Variables
189         Connection con = null;
190         PreparedStatement ps = null;
191
192         try { // Update Existing Entry
193             con = DBConnection.getConnection();
194             ps = con.prepareStatement(SurveyImpl.UPDATE);
195             ps.setString(1, survey.getName());
196             ps.setInt(2, survey.getSurveyId());
197             ps.executeUpdate();
198             ps.close();
199         } catch(Exception ex) {
200             System.out.println("Error: " + ex.getMessage());
201             // @TODO: Handle Exception
202         } finally {
203             try {
204                 if(con != null) {
205                     con.close();
206                     con = null;
207                 }
208             } catch(Exception ex){}
209         }
210     }
211
212     /***** Delete Methods *****/
213     public static void deleteBySurveyId(int surveyId) {
214         // Instantiate Variables
215         Connection con = null;
216         PreparedStatement ps = null;
217
218         try { // Delete Existing Entry
219             con = DBConnection.getConnection();
220             ps = con.prepareStatement(SurveyImpl.DELETE_BY_SURVEY_ID);
221             ps.setInt(1, surveyId);
222             ps.executeUpdate();
223             ps.close();
224         } catch(Exception ex) {
225             System.out.println("Error: " + ex.getMessage());
226             // @TODO: Handle Exception
227         } finally {
228             try {
229                 if(con != null) {
230                     con.close();
231                     con = null;
232                 }
233             } catch(Exception ex){}
234         }
235     }
236
237     public static void deleteByCompanyId(int companyId) {
238         // Instantiate Variables
239         Connection con = null;
240         PreparedStatement ps = null;
241
242         try { // Delete Existing Entry
243             con = DBConnection.getConnection();
244             ps = con.prepareStatement(SurveyImpl.DELETE_BY_COMPANY_ID);
245             ps.setInt(1, companyId);
246             ps.executeUpdate();
247             ps.close();
248         } catch(Exception ex) {
249             System.out.println("Error: " + ex.getMessage());
250             // @TODO: Handle Exception
251         } finally {
252             try {
253                 if(con != null) {
254                     con.close();
255                     con = null;
256                 }
257             } catch(Exception ex){}
258         }
259     }
260 }
261
```

7 Software Analysis

This section contains areas that after researching and analysis of parts of the system, that an improvement or change in the design is needed. The reason for the change can vary for many reasons such as bugs, performance issues, code reuse, etc…

7.1 Multiple Flows

@TODO:

7.2 Total PCL Library Integration

@TODO: Added pcl to admin website for reuse

7.3 Database Improvement

@TODO: improve Database connections

7.4 Revision Object Improvement

@TODO: Took a different approach to making the revision object.

7.4 JSON Improvement

@TODO: JSON files too big.