Rest APIs Application

Project Proposal Forum

CNLjunggren

<https://github.com/CNLjunggren>

Revision 1.0

Date: 2/18/2020

**Table of Contents:**

[**Project Overview & Objectives:** 2](#_Toc33878255)

[State the Problem: 2](#_Toc33878256)

[Background: 2](#_Toc33878257)

[Project Objectives: 2](#_Toc33878258)

[Challenges: 3](#_Toc33878259)

[Benefits & Opportunities: 3](#_Toc33878260)

[**Project Scope:** 3](#_Toc33878261)

[In-Scope: 3](#_Toc33878262)

[Out of Scope: 4](#_Toc33878263)

[Stakeholders 4](#_Toc33878264)

[Work Breakdown Chart (Continued on next page): 4](#_Toc33878265)

[*Legend:* 4](#_Toc33878266)

[*Chart:* 5](#_Toc33878267)

[**Project Completion:** 6](#_Toc33878268)

[Project Completion Criteria 6](#_Toc33878269)

[Assumption and Constraints 7](#_Toc33878270)

[**Project Controls:** 8](#_Toc33878271)

[Risk Management 8](#_Toc33878272)

[Change Control Log 8](#_Toc33878273)

[Roles and Responsibilities 8](#_Toc33878274)

[**Project Schedule:** 9](#_Toc33878275)

[Project Schedule 9](#_Toc33878276)

[Project Schedule (Starting February 18th) 9](#_Toc33878277)

[**Cost Estimate:** 9](#_Toc33878278)

[**Issue Log:** 10](#_Toc33878279)

# **Project Overview & Objectives:**

## State the Problem:

In the job market, many fresh out of college programmers have a hard time differentiating themselves from other programmers. Besides the projects from my experience while getting my degree at Grand Canyon University, I have little to add to my existing Project Portfolio. To solve this, I propose making a web application project that demonstrates my Object Orientated Programming (OOP) Language, project management, documentation, database, rest service, and API implementation skills regarding web application development.

## Background:

This project is being undertaken because of my need to differentiate myself from other programmers in the job market and to accurately demonstrate my OOP Language, project management, documentation, database, rest service, and API implementation skills regarding web application development. The goal of this project is to create a web application that demonstrates several uses of third-party APIS including OpenWeather and PayPal along with at least one demonstration of using a REST service to be communicated with to gather information from an API. Upon successful completion of this project, this web application will allow users to select an API service they wish to use, make a request to the given API, and receive data or results to view. Any results or API implementation that require tables of data to be shown will involve the data being stored in a standard SQLServer database within Visual Studio, while other requests not requiring long-term data storage will be directly shown to the user in a user-friendly manner.

## Project Objectives:

* This project will be a functioning web-application that utilizes at least two APIs sources and a database that can hold information passed from the APIs to the web app. The two API providers that this web-application will utilize are:
  + OpenWeather: A very extensive website that provides weather data to users in multiple different formats. Two of the in-scope APIs from OpenWeather that will be used are:
    - “Current Weather” call.
    - “5 Day/3 Hour Forecast” call.
  + PayPal: A well established enterprise for various financial services and stands as a widely accepted payment alternative. In scope for this project the web application will support:
    - REST API calls with PayPal’s Sandbox.
* The OpenWeather page will demonstrate my ability to utilize the OpenWeather API to create a web application capable of calling and receiving data from an API that can be displayed to the user and save to a database.
  + Libraries dedicated to weather data from OpenWeather will support CRUD functionality to allow the user to create/upload, view (read), update, and delete weather entries after logging in as an admin.
* PayPAl page will demonstrate my ability to use Javascript, REST services, and the PayPal REST API (Sandbox).
* Website will support standard user registration and login, with admin accounts able to access CRUD-based web pages to manage the database data.

## Challenges:

* Most of my REST services and API experience is from within the Java language/frameworks. It will be a challenge to learn how to fully utilize both with C# and .NET MVC.
* I have never utilized the PayPal APIs offered through PayPal Developer and will have to learn how to deserialize the API’s response into data and save it into a database.

## Benefits & Opportunities:

* This project will continue to demonstrate my potential as a developer and demonstrate that I can utilize databases, CRUD functionality, REST services, and several APIs within a web application.
* I will gain more insight and knowledge relating to REST services and API technology by doing this project.

# **Project Scope:**

This project is an account-based web-application where users can log in and register to the site. Once a user has logged into an account the website will allow users to make requests for weather information to OpenWeather through the OpenWeather page and allow for mock transactions and other API requests to PayPal through the PayPal page. It is important that features are laid out as either in-scope or out of scope for successful completion. A list of both in-scope and out of scope features are listed below.

## In-Scope:

* The web application will support user account-based browsing, with normal users unable to access administration pages and for no access to be allowed on anything but the home, login, and registration pages unless they log in.
* Navigation bar will be a partial page that allows for user-friendly navigation of the site. This navigation bar will also be adaptable, so it displays the correct options based on a user’s access level.
* A web page for OpenWeather’s API usage where users can make requests for “Current Weather” and “5-Days/3Hours” weather data calls from OpenWeather. This page must include:
  + Data validation for each field, ensuring that malicious and/or erroneous inputs cannot be submitted.
  + Methods to select the measurement system, location, and type of call the user is requesting.
* A web page for PayPal’s REST API and at least one other API usage where users can make mock transactions and request some form of data (likely mock data) from PayPal. This page must include the following:
  + Data validation for each field, ensuring that malicious and/or erroneous inputs cannot be submitted.
* Web page for the viewing all weather data in the database that supports admin functionality by checking if an admin is accessing the page. Only admins will be able to delete weather data from the database utilizing CRUD functionality on this page.
* An admin only web page for the viewing all transaction data in the database. This page will allow for admins to view non-sensitive information about each transaction and delete transactions from the database utilizing CRUD functionality.
* An admin only web page for the viewing all user accounts in the database. This page will allow for admins to view non-sensitive information about each user and delete them from the database utilizing CRUD functionality.
* Business and Data (DAO) services for each API call/request that successfully takes user inputs and makes appropriate API calls to either OpenWeather or Paypal, saves the API responses into readable/processable data, and save data to the database, directly display data, and/or display error or confirmation messages back to the user.

## Out of Scope:

* The ability to receive weather alerts and or summaries of months and years in a given location are considered out of scope as I already will show I can utilize OpenWeather’s API services.
* Allowing users to make actual transactions through PayPal is out of scope as I would need a specific developer role from PayPal to utilize actual transaction. Another reason this feature is considered out of scope is because there is no services or products on the site to monetize.

|  |  |  |
| --- | --- | --- |
| Stakeholders | | |
| Stakeholder Name | Role(s) | Responsibilities |
| Caleb Ljunggren | Developer | Developing |
| OpenWeather | API/weather data provider | Taking requests for weather data and returning an API response. |
| PayPal (PayPal Developer) | API/mock transaction provider/handler | Taking and partially handling mock transaction from the user. |

## Work Breakdown Chart (Continued on next page):

### *A picture containing meter Description automatically generatedLegend:*

### A picture containing meter Description automatically generated*Chart:*

# **Project Completion:**

By the completion of this project there will be;

* A home page that explains the functionality of the web application and have some basic information about the creator.
* A functioning database that can store user accounts, transaction data, weather data, and any other functional data tables for API functionality.
* Functional user account restrictions and user registration/login that prevents non-admins from accessing admin webpages and functions. This also will restrict people from accessing the APIs if they have not logged in.
* An OpenWeather page that supports request form/s for the “Current Weather” and “5-Days/3-Hours” APIS provided by OpenWeather.
* Functionality in the form of controllers, models, views, and business and data services that take the user requests for OpenWeather’s weather data and create the appropriate API call. Once the call is sent, the application will either convert this data and save/display it to the user or display an error message.
* A PayPal page that supports request form/s for the REST API “Sandbox” mock transactions and at least one other API provided by PayPal.
* Functionality in the form of controllers, models, views, and business and data services that take the user requests for PayPal’s APIs/ mock transactions and create the appropriate API call. Once the call is sent, the application will either convert this data and save/display it to the user or display an error message.
* Data Validation for all forms on either page to prevent users from inputting malicious/erroneous data inputs.
* Page security that prevents users from bypassing roles through entering direct webpage https.
* Navigation bar that allows for user-friendly navigation of the site that displays the correct options based on a user’s access level.
* Web pages for viewing all data from the database, with all but the weather data tables being completely locked behind admin roles.
* CRUD functionality locked behind admin-access that allows admins to remove and/or edit data from the database.

|  |
| --- |
| Project Completion Criteria |
| 1. A user will be able to log in and register to the web application. |
| 1. There will be a functioning database that stores data for users, weather data, mock transactions, and any other PayPal related data required for their APIs. |
| 1. Users will be able to request the “Current Weather” and “5-Day/3-Hour” data and forecast information from the website. The web app will validate the inputs, make an API request, then save and display this data to the user. |
| 1. Users will be able to make mock transactions and use at least one other API from PayPal. The web app will validate user inputs, make an API request, then save and/or display this data to the user. |
| 1. Admins can log in and access several pages not available to other user roles/nonusers. |
| 1. Users can view a paginated and sortable table of weather data from OpenWeather. |
| 1. Admins can view tables showing a list of all users, list of weather data from OpenWeather, and a limited table view of all mock transactions from PayPal. These tables can be revised through CRUD functionality, allowing admins to remove data from each database table. |
| 1. Everyone can access a home page that explains the web application’s functions and provides some information about the author. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Assumption and Constraints | | | | | |
| ID | **Description** | **Comments** | **Type** | **Status** | **Date Entered** |
| 1 | This project requires a database capable of storing and tracking users, weather, and mock transaction data. | Database containing tables for users, weather data, and mock transactions. | A | **In Scope:** Project will include a database with at least 3 tables. (one for each data/model type) | 2/29/2020 |
| 2 | This project has no budget, and thus is limited in what type of OpenWeather APIs are available. | OpenWeather has most APIs locked behind a paywall. | C | **In-Scope:** Project will only use the “Current Weather” and “5-Days/3-Hours” APIs. | 2/29/2020 |
| 3 | This project has no budget, and thus is limited to Sandbox mock transaction calls from PayPal. | PayPal only has mock transactions for free under Sandbox. | C | **In-Scope:** Project will only use PayPal’s Sandbox to make REST API mock transaction calls. | 2/29/2020 |
| 4 | There will need to be measures to prevent users and admins from accessing data they should not. | Program must have role security and hide certain fields of data. | A | **In-Scope:** Project will only show non-sensitive data to admins and users. | 2/29/2020 |
| 5 | Completed project will have too much data to nicely show on a single page. | Pagination, search and sorting functionality can mitigate this. | A/C | **In-Scope:** Project will include paginated tables. | 2/29/2020 |

# **Project Controls:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Risk Management | | | | |
| **Event Risk** | **Risk Probability** | **Risk Impact** | **Risk Mitigation** | **Contingency Plan** |
| Licensing issue | Low | High | The project will not allow the transferring, sharing, or saving of any data on the application. | If a licensing issue comes up, immediately multiple trusted individuals will be contacted for help. |
|  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Change Control Log | | | | | | | | | |
| **ID** | **Change Description** | **Priority** | **Originator** | **Date Entered** | **Date Assigned** | **Evaluator** | **Status** | **Date of Decision** | **Included in Rev. #** |
| 1 |  |  |  |  |  |  |  |  |  |
| 2 |  |  |  |  |  |  |  |  |  |
| 3 |  |  |  |  |  |  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Roles and Responsibilities | | | |
| **Name** | **Team** | **Project Role** | **Responsibility** |
| Caleb Ljunggren | Developers | Developer | Developing, research, coding, testing. |

# **Project Schedule:**

|  |
| --- |
| Project Schedule |
| 1. Initial Project Proposal (February 18th) |
| 1. Project Requirements (February 29th) |
| 1. Architectural Plan & Project Development Starts (March 14th) |
| 1. Application Testing (March 21st) |
| 1. Project Completion & Presentation (April 1st) |

|  |
| --- |
| Project Schedule (Starting February 18th) |
| 1. Environment and database setup. (week 1) |
| 1. Header/footer/body, navigation bar, information related pages. (week 1) |
| 1. OpenWeather pages with Data Validation. (week 2) |
| 1. User registration and login functionality (week 3) |
| 1. User table page with CRUD functionality. (week 3) |
| 1. Weather table page with CRUD functionality. (week 3) |
| 1. PayPal pages with Data Validation. (week 4) |
| 1. Mock transaction table page with CRUD functionality. (week 4) |
| 1. Final test cases & traceability matrixes. (week 5) |
| 1. Final error handling and bug fixing before presentation. (week 6) |

# **Cost Estimate:**

There are currently no expected expenses in developing this software unless issues relating to the database or licensing come up.

# **Issue Log:**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Issues Log | | | | | | | | | |
| **ID** | **Issue Description** | **Project Impact** | **Action Plan/Resolution** | **Owner** | **Importance** | **Date Entered** | **Date to Review** | | **Date Resolved** |
| 1 | Bootstrap is not working with razor tags for form creation. | High | Combine both Bootstrap with Razor form elements instead of solely Bootstrap. | Caleb Ljunggren | High | 2/19/2020 | | 2/20/2020 | 2/20/2020 |
| 2 | Unable to connect with OpenWeather’s APIs. | High | Solved by seeking help from various trusted sites. Will connect using HttpWebRequests. | Caleb Ljunggren | High | 2/20/2020 | | 2/21/2020 | 2/21/2020 |
| 3 | Unable to successfully deserialize JSON API responses. | High | Solved by seeking help from various trusted sites. Create model with helper classes to deserialize into a model. | Caleb Ljunggren | High | 2/20/2020 | | 2/22/2020 | 2/23/2020 |
| 4 | Error messages using ValidationSummary was not properly working. | Low | Will use ValidationMessageFor Razor tags for form data validation messages and used ViewBag to pass custom error messages to views. | Caleb Ljunggren | Med. | 2/23/2020 | | 2/25/2020 | 2/25/2020 |