# In My Words

*Project Design Document*

*CRDH Solutions*

Prepared by Anthony Haddox

Last Revised 1/15/2016

Version 1.4

# Revision History

| Date | Version | Description | Author |
|---|---|---|---|
| **12/21/2015** | 1.0 | Creation of SDD | Anthony Haddox |
| **12/24/2015** | 1.1 | Named the Project \| Updated sections 3, 5, 11 \| Created Use Cases template | Anthony Haddox |
| **12/26/2015** | 1.2 | Updated section 5 | Anthony Haddox |
| **12/28/2015** | 1.3 | Updated section 7 | Anthony Haddox |
| **1/15/2016** | 1.4 | Updated section 11 | Anthony Haddox |

# Table of Contents

# 1. Introduction

The Software Design Document (SDD) is a document to provide documentation which will be used to aid in software development by providing the details for how the software should be built. This document contains narrative documentation of the software design for the project as well as: case models, object behavior models, and other supporting requirement information.

## 1.1 Purpose

The purpose of the SDD is to provide a detailed enough description of a system's design to allow for software development to proceed with an understanding of what is to be built and how it is expected to be built. The SDD provides the information necessary to outline the details for the software and system to be built.

## 1.2 Scope

This SDD is for a lightweight web service that provides a creative writing outlet for a multi-national user base. This SDD will focus primarily on the back end design of the web service, with further design on the front end specified at a later date.

## 1.3 Definitions, Acronyms, and Abbreviations

- **IMW** - **I**n **M**y **W**ords. The working title of this software.
- **Software Design Document** - The Software Design Document (SDD) outlines the project requirements, specifications, and implementation details. It also serves as a reference to guide the development process.
- **Web Service** - A service offered by an electronic device to another device, communicating with each other via the internet. HTTP Protocols are utilized for machine-to-machine communication and to transfer file formats such as XML and JSON
- **RESTful** - REST stands for **Re**presentational **S**tate **T**ransfer, an architectural style for networked applications, primarily used to build lightweight, maintainable, and scalable web services. Services based upon REST are called RESTful services. While not dependent upon any protocol, almost every RESTful service uses HTTP as its underlying protocol.
- **SOAP** - **S**imple **O**bject **A**ccess **P**rotocol. IT is the specification for exchanging structured information in the implementation of web services in computer networks. SOAP has three major characteristics:
    1) Extensibility
    2) Neutrality (SOAP can operate over any transport protocol)
    3) Independence (SOAP allows for any programming model)
- **XML** - E**X**tensible **M**arkup **L**anguage. A simple, flexible text format designed to meet the challenges of large-scale electronic publishing. It defines a set of rules for encoding documents that is both human and machine-readable.
- **JSON** - **J**ava**S**cript **O**bject **N**otation. It uses human-readable text to transmit data objects consisting of attribute-value pairs. It is language-independent and primarily used for

asynchronous browser/server communication. JSON has steadily been replacing XML as the preferred format.

- **ASP.NET** - ASP.NET is an open source server-side web application framework, designed for web development to produce dynamic web pages. It was developed by Microsoft to allow programmers to build dynamic web sites, web applications, and web services. ASP.NET is built on the Common Language Runtime (CLR) which allows programmers to write ASP.NET code using any supported .NET language. The current stable release is version 4.6.
- **Common Language Runtime** - The Common Language Runtime (CLR) is the virtual machine component of Microsoft's .NET framework. It manage the execution of .NET programs using just-in-time compilation to convert compiled code into native machine instructions. IT provides memory management, type safety, exception handling, garbage collection, security, and thread management.
- **Model View Controller** - Model-View-Controller (MVC) is a software architectural pattern mostly for implementing user interfaces. IT divides a given software application into three interconnected parts to separate internal representations of information from the ways that information is presented to the user.
- **Entity Framework** - The Entity Framework (EF) is an object-relational mapper that enables .NET developers to work with relation data using domain-specific objects. It eliminates the need for most of the data-access code that developers usually need to write.
- **API** - **A**pplication **P**rogram **I**nterface. A set of routines, protocols, and tools for building software applications. The API specifies how software components should interact.
- **LINQ** - **L**anguage-**In**tegrated **Q**uery. LINQ is a set of features that extend the language syntax of C# and Visual Basic. LINQ introduces standard, easily-learned patterns for querying and updating data.
- **Cry** - That thing we do when we realize this **cannot** be accomplished.

## 1.4 Overview

The SDD is divided into 11 sections with various subsections. The main sections of the SDD are:

1. Introduction
2. The Problem
3. The Concept
4. Technology Overview
5. Use Cases
6. Design Overview
7. Model Overview
8. Controller Overview
9. View Overview
10. Development Timeline
11. Supplementary Documentation

# 2. The Problem

This section details the problem that CRDH Solutions is aiming to solve with this software.

## 2.1 Submitted to Microsoft

The following is part of the text submission to Microsoft outlining the problem CRDH Solutions is aiming to solve.

In an increasingly interconnected world it is important for everyone to be able to understand and empathize with people from other cultures and nations. Currently, much of the information people absorb about the world comes from media outlets or second-hand accounts through social media. Additionally, video is becoming the preferred medium for knowledge dissemination to the detriment of written words. CRDH Solutions is looking to achieve two goals: first, to expand consciousness about other cultures through first-hand accounts. Second, to improve literacy and communication skills by providing a writing and reading repository for use by anyone at any time.

## 2.2 The Problem in Detail

### 2.2.1 Identifying the issues

There are two issues CRDH Solutions has identified which can be rectified through the development of new software:

1. Cultural empathy and awareness
2. Literacy and written communication

The following subsections provide additional understanding of these issues.

### 2.2.2 Cultural Empathy and Awareness

The power and accessibility of the internet allow for the rapid exchange of ideas and information. Unfortunately, given the turmoil in the world today, some of the ideas and information spread around the internet offer a negative portrayal of other cultures and nations. Often times, the information people read about cultures and nations are from second or third-hand accounts, and nuances become lost in translation. While news media plays an important role in informing people about issues occurring in other parts of the world, it is difficult to find stories that aren't skewed against the people in those parts of the world. Because of the media portrayal, many people in the western world often have trouble empathizing with the people of other nations and cultures.

### 2.2.3 Literacy and Written Communication

The ability to not only read, but comprehend what you are reading, is an important skill for everyone to possess. While mediums such as picture and video are decent ways of self-expression, they don't facilitate the same mental exercise that results from reading and writing. To become and remain a successful person requires solid communication skills, particularly the ability to express what you are thinking; it can be difficult for a person to find ways to improve upon these skills.

# 3. The Concept

CRDH Solutions will be designing and implementing an online repository where users can submit their own stories, or anything else they write, for global viewing.

## 3.1 The Name

This project will be titled "In My Words" (IMW).

## 3.2 The Overview

IMW will serve as an online repository where users from around the world can view and submit stories or other things they have written. Users can submit stories anonymously, in their own language, at any time. A random selection of submissions will be displayed on the website's home page, with a search function for users to read a specific submission. IMW will track the number of views on every submission.

In addition to original submissions, users will also be allowed to offer translations of existing submissions. Users will have the ability to view the original and translated submissions side-by-side to promote the connection amongst languages. In the case that there are no user-submitted translations, users will have the ability to select an automatic translation, provided by the Translation API; IMW will still allow automatic translation if there are user-submitted translations. Submissions can have multiple user-submitted translations of the same language and there will be a rating system to determine which user-submitted translations are the most accurate and this may be used as a sorting criteria.

## 3.3 Submission Verification

There are no current plans to verify each user submission. In lieu of this, users will be able to report submissions if they do not meet the standards of IMW.

## 3.4 Automatic Translation

To artificially bypass the 2M characters/month limit of the free tier of Translation API, IMW will keep a database containing previous requests from Translation API. This will mean that only the first request of a language will be submitted to Translation API; if we have already submitted a request then we will check the existing database.

# 4.Technology Overview

## 4.1 Supporting Framework

This project will be implemented using Microsoft's ASP.NET Framework. Specifically, we will be using the Model View Controller (MVC) Framework to create a RESTful web service. The MVC Framework will allow for rapid development due to the automatic scaffolding it provides. In addition, the ASP.NET Framework provides many useful APIs which will decrease the necessary time for development, testing, and debugging.

## 4.2 Data Framework

In conjunction with ASP.NET, this project will utilize the Entity Framework. The Entity Framework allows for a code-first approach to database design; we will create the data structures in C# and the framework will automatically scaffold and generate the necessary code to design the supporting database. We will then be able to use LINQ queries to interact with the database through code.

## 4.3 Translation API

Microsoft provides a Translator API which can be integrated with existing codebases for quick use. The free subscription to the Translator API has a constraint of two (2) million characters per month. To make use of the API we will need to register our application and create credentials for it. At that point we will be able to obtain an access token through the HTTP POST method when we need to make use of the Translator API.

## 4.4 Web Service Hosting

Microsoft Azure will be used to host the web service application. There is a free tier of use that allows for development and testing. Azure allows for conversion to another plan without any delay or downtime. While the costs of the next tier of service are relatively low, the plan is to spend nothing to reduce development overhead.

## 4.5 Open Source Hosting

In the interests of keeping this project open source, CRDH Solutions will be creating a code repository on GitHub for use by other developers.

# 5. Use Cases

## 5.1 Create Submission

| | | |
|---|---|---|
| **Use case name:**<br>Create Submission | **ID:**<br>CSub | **Priority:**<br>Medium |
| **Primary actor:**<br>User | **Use case type:**<br>Click here to enter text. | **Level:**<br>Overview |
| **Brief description:**<br>The user creates and submits content. | | |
| **Goal:**<br>• Add submission content to database<br>• Redirect user to submission display (VSub) | | |
| **Success Measurement:**<br>• Database is updated<br>• Automatic page redirection | | |
| **Precondition:**<br>• None.<br><br>**Trigger:**<br>• The user clicks the "Create Story" link/button | | |
| **Relationships:**<br>**Include:** Click here to enter text.<br>**Extend:** Click here to enter text.<br>**Depends on:** Click here to enter text. | | |
| **Flow of Events:**<br>1. The user clicks the "Create Story" link/button<br>2. The user is sent to the CSub page<br>3. The user fills out all necessary fields<br>4. The user clicks the "Publish Story" link/button<br>5. The database is updated<br>6. The user is redirected to the VSub page | | |
| **Assumptions:** Click here to enter text. | | |
| **Implementation Constraints and Specifications:** Click here to enter text. | | |

## 5.2 View Submission

| **Use case name:**<br>View Submission | **ID:**<br>VSub | **Priority:**<br>Medium |
|---|---|---|
| **Primary actor:**<br>User | **Use case type:**<br>Click here to enter text. | **Level:**<br>Overview |
| **Brief description:**<br>The website displays user-specified content | | |
| **Goal:**<br>• Display submission content | | |
| **Success Measurement:**<br>• User is shown content<br>• Successful HTTP GET | | |
| **Precondition:**<br>• Content exists<br><br>**Trigger:**<br>• The user clicks the "View Story" link/button | | |
| **Relationships:**<br>**Include:** Click here to enter text.<br>**Extend:** Click here to enter text.<br>**Depends on:** Database populated with content | | |
| **Flow of Events:**<br>1. The user clicks the "View Story" link/button<br>2. The content is displayed | | |
| **Assumptions:** There is preexisting content | | |
| **Implementation Constraints and Specifications:** Click here to enter text. | | |

## 5.3 Report Submission

| **Use case name:** Report Submission | **ID:** RSub | **Priority:** Medium |
|---|---|---|
| **Primary actor:** User | **Use case type:** Click here to enter text. | **Level:** Overview |
| **Brief description:** The user reports a submission inconsistent with IMW standards | | |
| **Goal:**<br>• Populate data base with report details | | |
| **Success Measurement:**<br>• Database is populated with submission details and reporter's details (tentative) | | |
| **Precondition:**<br>• Content exists<br><br>**Trigger:**<br>• The user clicks the "Report Story" link/button | | |
| **Relationships:**<br>**Include:** Click here to enter text.<br>**Extend:** Click here to enter text.<br>**Depends on:** Database populated with content | | |
| **Flow of Events:**<br>1. The user clicks the "Report Story" link/button<br>2. If there is no existing hash in the report database: create database entry<br>3. If there is an existing hash in the report database: update database entry | | |
| **Assumptions:** There is preexisting content | | |
| **Implementation Constraints and Specifications:** Click here to enter text. | | |

## 5.4 Delete Submission

| **Use case name:** Delete Submission | **ID:** DSub | **Priority:** Medium |
|---|---|---|
| **Primary actor:** Administrator | **Use case type:** Click here to enter text. | **Level:** Overview |
| **Brief description:** The administrator deletes a submission from all databases | | |
| **Goal:**<br>• All database entries of the submission are removed | | |
| **Success Measurement:**<br>• Successful database update | | |
| **Precondition:**<br>• Content exists<br>• Administrative access to database<br><br>**Trigger:**<br>• Click here to enter text. | | |
| **Relationships:**<br>**Include:** Click here to enter text.<br>**Extend:** Click here to enter text.<br>**Depends on:** Database populated with content | | |
| **Flow of Events:**<br>1. The administrator removes relation from the report databse<br>2. The administrator removes relation from the submission database | | |
| **Assumptions:** There is preexisting content | | |
| **Implementation Constraints and Specifications:** Click here to enter text. | | |

## 5.5 Automatic Translation

| **Use case name:**<br>Automatic translation | **ID:**<br>ATrans | **Priority:**<br>Medium |
|---|---|---|
| **Primary actor:**<br>User | **Use case type:**<br>Click here to enter text. | **Level:**<br>Overview |
| **Brief description:**<br>The website contacts Translation API and displays translated user content | | |
| **Goal:**<br>• Redirect user to translated submission display (VTrans) | | |
| **Success Measurement:**<br>• User is shown translated content<br>• Successful HTTP GET | | |
| **Precondition:**<br>• Content exists<br>• First language request to Translation API<br><br>**Trigger:**<br>• The user clicks the "Translate" link/button | | |
| **Relationships:**<br>**Include:** Click here to enter text.<br>**Extend:** Click here to enter text.<br>**Depends on:** Database populated with content, Translation API | | |
| **Flow of Events:**<br>1. The user selects the desired translation target language<br>2. The user clicks the "Translate" link/button<br>3. The Translation API is accessed<br>4. The translated results are displayed along with the original submission content (VTrans) | | |
| **Assumptions:** There is preexisting content, successful GET from Translation API | | |
| **Implementation Constraints and Specifications:** Under the 2M character/month limit for Translation API | | |

## 5.6 Submit Translation

| **Use case name:**<br>Submit translation | **ID:**<br>STrans | **Priority:**<br>Medium |
|---|---|---|
| **Primary actor:**<br>User | **Use case type:**<br>Click here to enter text. | **Level:**<br>Overview |
| **Brief description:**<br>The user submits a translation of existing content | | |
| **Goal:**<br>• Redirect user to translated submission display (VTrans) | | |
| **Success Measurement:**<br>• Database is updated | | |
| **Precondition:**<br>• Content exists<br><br>**Trigger:**<br>• The user clicks the "Submit Translation" link/button | | |
| **Relationships:**<br>**Include:** Click here to enter text.<br>**Extend:** Click here to enter text.<br>**Depends on:** Database populated with content | | |
| **Flow of Events:**<br>1. The user clicks the "Submit Translation" link/button<br>2. The user inputs the translation target language<br>3. The user offers their translation<br>4. The user clicks the "Submit" button/link<br>5. The database is updated<br>6. The user is redirected to translated submission display (VTrans) | | |
| **Assumptions:** There is preexisting content | | |
| **Implementation Constraints and Specifications:** Click here to enter text. | | |

## 5.7 View Translation

| **Use case name:**<br>View Translation | **ID:**<br>VTrans | **Priority:**<br>Medium |
|---|---|---|
| **Primary actor:**<br>User | **Use case type:**<br>Click here to enter text. | **Level:**<br>Overview |
| **Brief description:**<br>The website displays translated user content | | |
| **Goal:**<br>• Redirect user to translated submission display | | |
| **Success Measurement:**<br>• User is shown translated content and original content | | |
| **Precondition:**<br>• Content exists<br>• Translation exists<br><br>**Trigger:**<br>• STrans or ATrans events | | |
| **Relationships:**<br>**Include:** Click here to enter text.<br>**Extend:** Click here to enter text.<br>**Depends on:** Database populated with content, Translation API | | |
| **Flow of Events:**<br>1. The translated results are displayed along with the original submission content | | |
| **Assumptions:** There is preexisting content, successful GET from Translation API | | |
| **Implementation Constraints and Specifications:** Click here to enter text. | | |

## 6. Design Overview

Content here...

# 7.Model Overview

## 7.1 Submission

| **Class Name**: Submission | |
|---|---|
| **Brief Description**: Data container for user submissions. | |
| **Attributes (fields)** | **Attribute Description** |
| DateTime datePublished | This variable tracks when the submission was published to the IMW database. |
| string text | This variable contains the "story" submitted by the user. |
| Dictionary<string, Translation> automaticTranslations | This dictionary contains all of the translations performed by Translation API. The language, in string format, serves as the key in this key-value pairing. |
| ArrayList<Translation> userTranslations | This array list contains all of the user submitted translations. The data structure used for this field is subject to change or removal. |
| int views | This variable tracks the number of views this submission has received. |
| string userHostAddress | This variable contains the IP address of the user. |
| **Methods (operations)** | **Method Description** |
| | |

## 7.2 Translation

| **Class Name**: Translation | |
|---|---|
| **Brief Description**: Data container for translations of submission text. | |
| **Attributes (fields)** | **Attribute Description** |
| string submissionHash | This variable contains the hash of the submission being translated for. |
| string text | This variable contains the translation submitted by the user. |
| string language | This string contains the language of the translation. |
| string userHostAddress | This variable contains the IP address of the user. |
| **Methods (operations)** | **Method Description** |
| | |

## 7.3 Submissions (For EF)

| **Class Name**: Submissions | |
|---|---|
| **Brief Description**: Data structure to create and populate EF tables. | |
| **Attributes (fields)** | **Attribute Description** |
| string submissionHash | This variable is a hash of the submission. This will allow for O(1) lookup times. Will serve as the primary key of the relation. |

| | |
|---|---|
| DateTime datePublished | This variable tracks when the submission was published to the IMW database. |
| string text | This variable contains the "story" submitted by the user. |
| int views | This variable tracks the number of views this submission has received. |
| string userHostAddress | This variable contains the IP address of the user. |
| **Methods (operations)** | **Method Description** |
| | |

## 7.4 Translations (For EF)

| **Class Name**: Translations | |
|---|---|
| **Brief Description**: Data structure to create and populate EF tables. | |
| **Attributes (fields)** | **Attribute Description** |
| string submissionHash | This variable is a hash of the submission. This will allow for O(1) lookup times. Will serve as the primary key of the relation. |
| DateTime datePublished | This variable tracks when the submission was published to the IMW database. |
| string text | This variable contains the translation submitted by the user. |
| string language | This variable contains the language of the translation |
| string userHostAddress | This variable contains the IP address of the user. |
| **Methods (operations)** | **Method Description** |
| | |

## 8. Controller Overview

Content here...

## 9. View Overview

Content here...

## 10. Development Timeline

| Week | Tasks |
|---|---|
| **Week 0 (12/21 - 12/25)** | • Create SDD<br>• Download and install Microsoft Visual Studio<br>• Complete the tutorial (See section 11.1 MVC Framework Tutorial) to get acclimated to the MVC Framework<br>• Create GitHub Organization<br>  o Invite team members to organization<br>  o Create project repository<br>• Name project<br>• Write section "5. Use Cases" of the SDD |
| **Week 1 (12/28 - 1/1)** | • Write section "7. Model Overview" of the SDD<br>• Write section "8. Controller Overview" of the SDD<br>• Begin work on data structures<br>• Begin work on controllers |
| **Week 2 (1/4 - 1/8)** | • Write Section "9. View Overview" of the SDD<br>• Begin work on views<br>• Begin work on user groups and authentication (optional)<br>• Implement Translator API |
| **Week 3 (1/11 - 1/15)** | • Finalize software Readme |
| **Week 4 (1/18 - 1/19)** | • Final stage of testing and debugging<br>• Create submission video<br>• Draft written submission<br>• Cry. |

# 11. Supplemental Documentation

## 11.1 MVC Framework Tutorial

The MVC Framework tutorial can be found here: http://www.asp.net/mvc/overview/getting-started/introduction/getting-started. There are 11 parts in the tutorial but not everything will be applicable.

## 11.2 Translator API

The link to sign up for the Translator API: https://www.microsoft.com/en-us/translator/getstarted.aspx

## 11.3 Deployment to Azure

Similar to the MVC Framework tutorial, this shows how to deploy the project to Microsoft Azure. https://azure.microsoft.com/en-us/documentation/articles/web-sites-dotnet-get-started/

## 11.4 SDD Example

The example this SDD is based upon: https://www.oasis-open.org/committees/download.php/24846/Example-SoftwareDesignDocument-LegalXMLUtility.pdf

## 11.5 MD5 Hashing

This is a simple way to generate an MD5 Hash: http://blogs.msdn.com/b/csharpfaq/archive/2006/10/09/how-do-i-calculate-a-md5-hash-from-a-string_3f00_.aspx