Rapid Storage Reporting Tool Project

Technology Feasibility

October 22, 2013

Chad Dulake
Nakai McCarty
Forrest Townsend

Table of Contents

Introduction

This document provides a brief overview of our approach to the project. This includes various technologies which we anticipate will be either necessary or potentially useful in the course of production. A brief discussion of how these technologies will be integrated follows this listing. Finally, the document concludes with a GUI mockup for the report generation interface and a two architectural diagrams illustrating data and control flow in the system respectively. These diagrams serve as proof of concept for the project.

Technology Overview

Languages: C#, CSS, HTML, Javascript, Linq, XML, XSLT
Libraries: ASP.NET MVC5, Entity Framework, JQuery, KnockoutJS, Node.js, Typescript, Windows Authentication
Graphical Libraries: d3.js, Twitter Bootstrap
System-Type: Centralized system; Local intranet
Data-Transfer Protocols: SOAP for exchanging information, since we are using XML this works great.
Database Systems: Microsoft SQL Server
Browsers: Internet Explorer 10, Chrome, Firefox, and Safari

Technology Integration

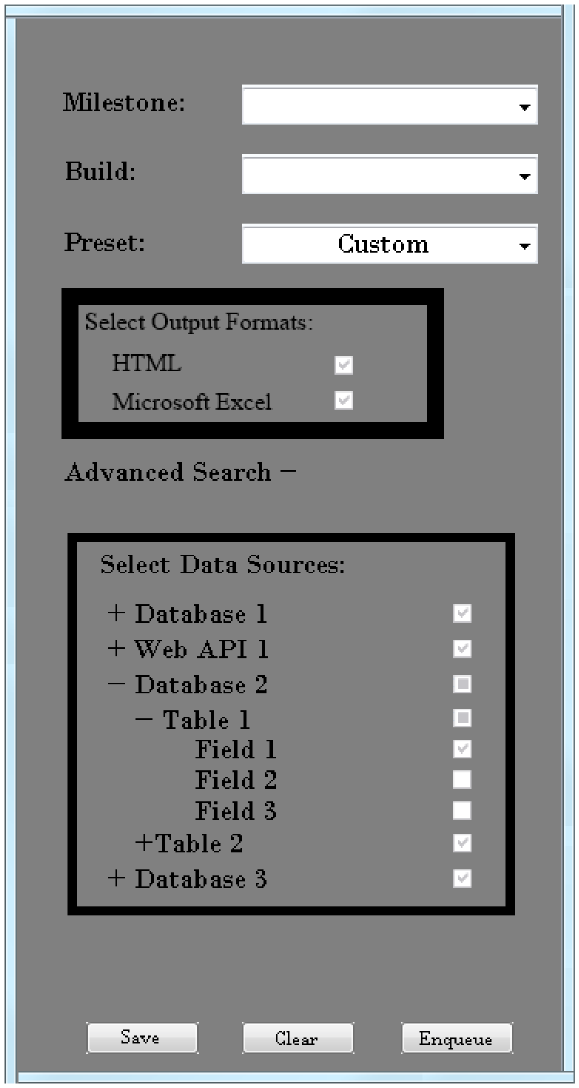
Inputs handled by the tool will consist of formatting and content request instructions given by the user, query results from Microsoft SQL Servers, and the results from other input modules which may be developed at a later date or by a third party. The tool will process input data and produce XML output which will be further processed by various output modules into more human readable formats like HTML or Excel data. Final outputs can be downloaded and HTML outputs can be previewed once a report has been generated. The users will interact with a single web interface. This interface allows the user to select which data sources they would like to aggregate into a report. Stylistically, the interface will be compatible with the four major browsers: Internet Explorer, Chrome, Firefox and Safari.

The libraries that are listed above, will help us functionally reach completion as well as to provide us a “niceness” sense to our tool. The graphical libraries that we are going to explore include the Twitter Bootstrap and d3.js library. Twitter Bootstrap is a way to give your site a common looking sense of completeness. The d3.js library is used to model data is a graphical and creative way. Node.js may be used if we want to do JavaScript processing on the server side.

One of the key data communication languages we are using is XML. For our output formats, we will use XML to prepare each result. Both outputs that we intend in using, Microsoft Excel and HTML, both translate from XML incredibly well. A great feature of using XML throughout is that we can manually generate test XML data to throw into each module individually for testing.

Proof of Feasibility

These are quick design mockups of the graphical user interface in our web application.



Milestone:

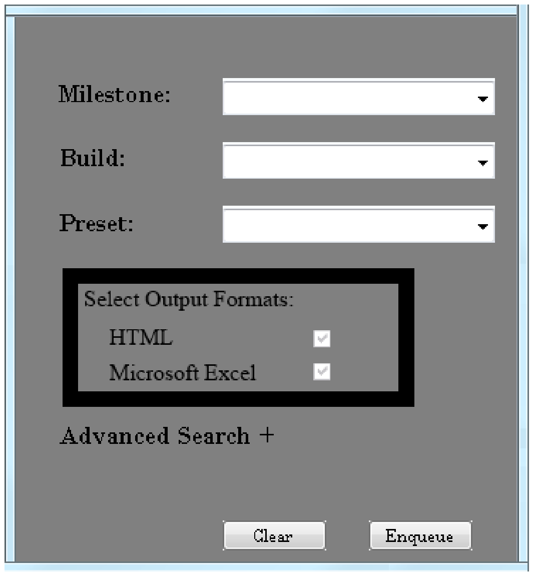
Build:

Preset: Custom

Select Output Formats:
- HTML ✓
- Microsoft Excel ✓

Advanced Search −

Select Data Sources:
- + Database 1 ✓
- + Web API 1 ✓
- − Database 2 ▪
  - − Table 1 ▪
    - Field 1 ✓
    - Field 2 ☐
    - Field 3 ☐
  - +Table 2 ✓
- + Database 3 ✓



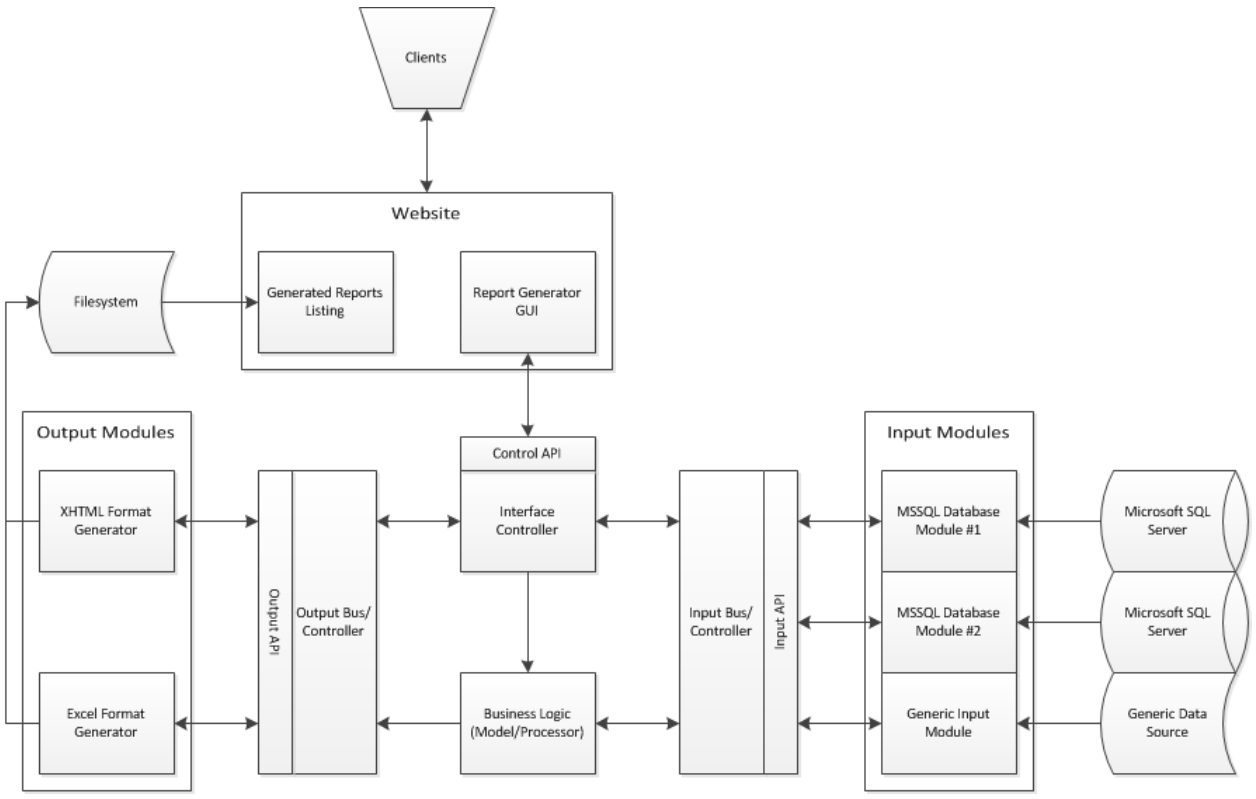
Milestone:

Build:

Preset:

Select Output Formats:
- HTML ✓
- Microsoft Excel ✓

Advanced Search +



This is a draft of our architecture for this project; it is an MVC (Model-View-Control) architecture with an addition of event busses to add modularity and anonymity between the system and its inputs & outputs. Each input module registers itself with the input bus via the input API while providing its available fields and tables (and/or subfields and subtables). Likewise the output generators register themselves with the output bus via the output API and thus introduce themselves to the interface controller for use. Only the control API does not provide a method for registering a listener, instead it’s a simple web API that any system (but in this case our website) could interact with.

All the data passed throughout the system is in an XML format due to the heavy use of Microsoft’s products and their excellent built in XML processing libraries. It was only natural to choose SOAP as our protocol for all application interfaces due to it being lightweight, well defined protocol for transmitting XML over HTTP.

While other languages and forms of data are used throughout the system, the *only* data interchange is done with XML.