

Dashboard and Weather Incident Verification System

CSE Capstone Team Members:

Evan Dontje

Edward Waller

Joe Lockard

Auto-Owners Team Members:

Scott Lake

Jim Walsh

Ed Keen

February 1, 2010

# Table of Contents

[Table of Contents 2](#_Toc252790893)

[Team Introduction 4](#_Toc252790894)

[Functional Specifications 5](#_Toc252790895)

[Overview 5](#_Toc252790896)

[Features 5](#_Toc252790897)

[-Login Based Customization 5](#_Toc252790898)

[-Policy Holder’s Information 5](#_Toc252790899)

[-Search Options 5](#_Toc252790900)

[-Email 5](#_Toc252790901)

[-Mobile Optimization 6](#_Toc252790902)

[Design Specifications 7](#_Toc252790903)

[Customization 7](#_Toc252790904)

[Search Options 7](#_Toc252790905)

[Display 7](#_Toc252790906)

[Store Results 7](#_Toc252790907)

[-File 7](#_Toc252790908)

[-Email 7](#_Toc252790909)

[User Interface 8](#_Toc252790910)

[Data Flow Diagram 10](#_Toc252790911)

[Technical Specifications 12](#_Toc252790912)

[System Components 12](#_Toc252790913)

[-Hardware Platforms 12](#_Toc252790914)

[Languages and Software 12](#_Toc252790915)

[UML Diagram 13](#_Toc252790916)

[Testing 15](#_Toc252790917)

[Login Screen 15](#_Toc252790918)

[User Preferences 15](#_Toc252790919)

[Information 15](#_Toc252790920)

[Schedule 16](#_Toc252790921)

[Risks 18](#_Toc252790922)

[WSRP 18](#_Toc252790923)

[Ranking of Weather Services 18](#_Toc252790924)

[Extrapolation of Weather Data 18](#_Toc252790925)

[Weather API’s 18](#_Toc252790926)

# Team Introduction

Evan Dontje ([dontjee@msu.edu](mailto:dontjee@msu.edu)) Customer Contact

Edward Waller([wallered@msu.edu](mailto:wallered@msu.edu)) Project Manager

Joe Lockard([lockardj@msu.edu](mailto:lockardj@msu.edu)) Developer

# Functional Specifications

## Overview

This web application is for Auto-Owners Insurance claims associates that handle the enormous amount of claims that occur due to the weather. The application will allow the associate to cross check a claim with actual weather history for the date specified on the claim. It will query several of the most reliable weather history sources on the web, and display the results in an easy to read format. The associate may then make a determination as to whether or not the claim is valid. The data retrieved in this application will also be preserved, if the user wishes.

## Features

### -Login Based Customization

The user will be able to specify several things on the application. A claims worker that only handles claims in couple states, can specify that in their preferences. This will make the drop down menu for addresses in the search only show the states relevant to that worker. Likewise, the policy holder drop down menu will only display customers that fall within those states as well.

### -Policy Holder’s Information

If the user specifies a policy holder in the search, rather than an address, the policy of that customer will then be displayed on the results page, as well as the weather results. The begin date of their policy will also be listed.

### -Search Options

The user may not always want to search for ALL weather events, in a given case. For that reason, there are options for the user to search for only certain weather events. Certain weather services that will be used only search for catastrophic events, for example. If the user only wishes to search for hurricanes, for example, then it will be of benefit to not search the other web services available. Searching for specific weather events may improve speed, as well as not clutter the results.

### -Email

The user will have the option of pressing an “Email” button on the results page. This will format the results, to be sent out via email.

### -Mobile Optimization

Many claims workers who wish to use this application may not be in an office, in front of a computer. For this reason, the application will be optimized for use with mobile devices, such as Ipods and Smartphones.

# Design Specifications

## Customization

Customization of this program will be key for the user in terms of speed. A user may specify the areas of the country in which they handle claims. Then, if they wish to scroll through the policy holder drop down menu to search for a particular customer, they will be able to do so much faster because the list will be narrowed down considerably.

## Search Options

The user may choose to search for ALL weather occurrences in a particular area, or they may choose particular types of events. Additionally, one can search specific weather services, instead of all of them.

## Display

The results page will list all weather events that fall under the search options. Next to each result, the source will also be given. This will help the user determine the credibility of each result.

## Store Results

The search results may prove to be very valuable to the user in sorting out the claim. Since the information obtained is not guaranteed to always be there, it will be very wise to store it.

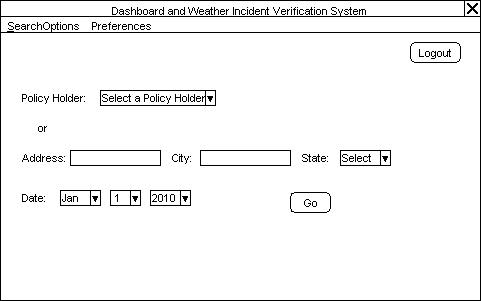
### -File

The results of a search may be stored as a file on the local machine by pressing the “Save Results” button on the search results page.

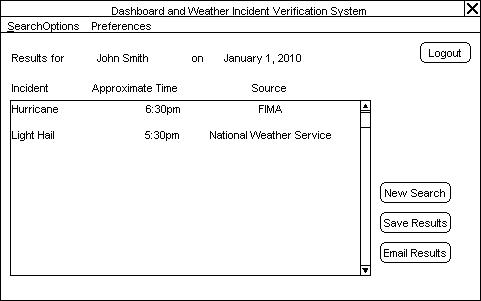
### -Email

The results of a search may also be formatted and sent via email by pressing the “Email” button on the search results page.

## User Interface

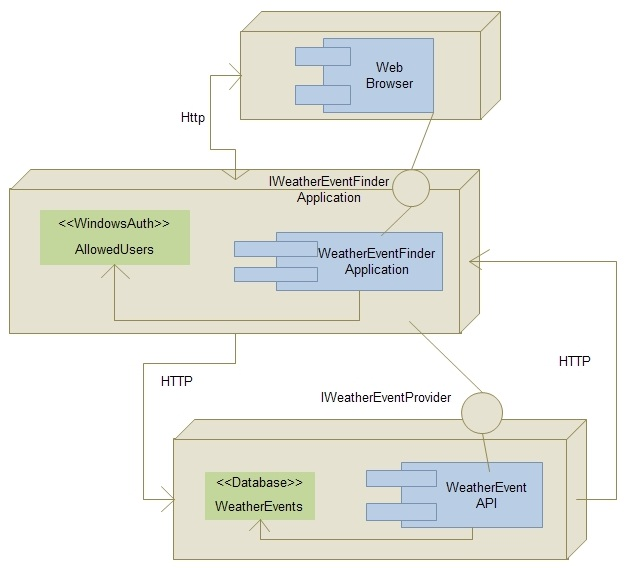


The screen shown above is what the user will see once he/she enters their username and password. In the SearchOptions menu one may specify certain weather events, to be displayed on the results page. The default is set to display all event types. In the Preferences menu, the user may enter or edit their login information, as well as the locations of their policy holders.



The results page shown above lists all events which fall under the user-specified options. The “Save Results” button allows the user to save the results to a file. Similarly, the “Email Results” button allows the user to send the results out in an email. “New Search will return the user to the previous screen, and will clear all search fields.

## Data Flow Diagram



Our architecture is made of three parts. The first is the user’s web browser. This is the interface that is used by the user to access the site. To access the site the user must have a valid username and password.

The second part of our architecture is the web server. We have an Asp.Net MVC web application running on the server which handles the user requests. User authentication is handled with windows authentication on the web server.

The third part of the architecture is the remote web services that we make API requests to for our data. This part is made up of several different websites including NCDC and Wunderground.com.

# Technical Specifications

## System Components

### -Hardware Platforms

#### -PC

The majority of the associates using the program will be running the application on a PC.

#### -Mobile Device

The application will be optimized for mobile device use as well, if the user wishes.

### Languages and Software

#### -C#

The application will be written in the C# language, as this is the most convenient choice, given the software used.

#### -Visual Studio 2008

The IDE used for this application will be Visual Studio 2008. It is required in order to run ASP.NET MVC.

#### -ASP.NET MVC

This application will be built using ASP.NET MVC, a Microsoft framework used for building applications.

#### -Windows Server 2003

Windows Server 2003 will be used for the server.

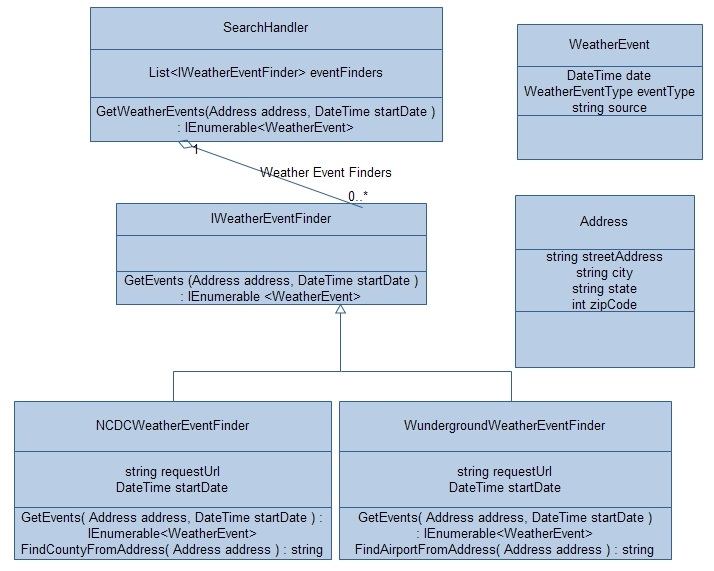
#### -Windows 7

The PC’s that this application will be built on will be using Windows 7 for their operating system.

#### -GIT SCM

GIT SCM is the version control system that will be used for developing this application.

## UML Diagram



SearchHandler

The SearchHandler will contain a list of implementations of the IWeatherEventFinder class. When a request is made using the GetWeatherEvents method, the list will be iterated through and each implementation will have the GetEvents method called on it. Each method returns a group of weather events, which is sorted and returned. To call the method, an address is passed in, and so is a start date. It is assumed that the request will return WeatherEvents from the start date until the current date.

IWeatherEventFinder

Interface that contains one method, GetEvents. Implementations of the interface will take an address and a start date and return an enumerable list of WeatherEvents.

NCDCWeatherEventFinder

This class implements the IWeatherEventFinder interface. It queries the National Climactic Data Center website to get a list of recent weather events. There is no public API so this is done by making an http POST to the website and parsing the html that is returned. The class has two private members, requestUrl which is the url to make the http request to, and startDate which is the date to start getting weather events from.

The class has two methods. GetEvents is public and its parameters are an address and a startDate. The class also has a private method GetCountyFromAddress. This method analyzes an address to find which County that address is located in. This information is required for the GetEvents call because the NCDC stores events based on the County in which they occurred.

WundergroundWeatherEventFinder

This class implements the IWeatherEventFinder interface. It queries Wunderground.com to get a list of weather events. An API call is made, and a csv file is returned which is then parsed to get a list of WeatherEvents. The class has two private members, requestUrl which is the url to make the http request to, and startDate which is the date to start getting weather events from.

The class has two methods. GetEvents is public and its parameters are an address and a startDate. The class also has a FindAirportByAddress method which is used to find the closest airport to a given address. This is then used in the GetEvents call.

WeatherEvent

This class has no public methods, but does have 4 public properties. They are a date, the type of weather event, and the source of the data.

Address

This class has no public methods, and has three public properties. They are a street address, city, state, and zip code.

# Testing

## Login Screen

Security testing must be done with regard to the login screen, so that only the authorized Auto-Owners claims associates can log in. Since this program will be linked to their database of policy holders, this testing will be crucial.

## User Preferences

We must test the program to ensure that all user preferences will be saved from the last time the user was logged in. These preferences make the program more user friendly and can help save the user valuable time, so they are very important.

## Information

Information accuracy without a doubt the most important aspect of this program. The user will rely on this information to make a determination which can largely affect a customer life, so testing must be done to ensure that they are making the correct decision.

## Schedule

* Schedule
  + Overview
    - (2/22-3/03) Alpha Demonstration
    - (4/5-4/14) Beta Demonstration
    - (4/26-4/28) Project Video
    - (4/30) Design Day
  + Detailed
    - Jan. 25 – Jan. 31 (Week 1)
      * Server
        + Prototype communication with weather services.
      * User Interface
        + Create mockup of basic user interface.
      * User Interaction
        + Research how to request and display weather information with ASP.NET MVC
    - Feb. 1 – Feb. 7 (Week 2)
      * Server
        + Parse weather data into a consistent format
      * User Interface
        + Implement user interface as valid xhtml that communicates with the web server correctly.
      * User Interaction
        + Display parsed result data when a query is submitted.
    - Feb. 8 – Feb. 14 (Week 3)
      * Server
        + Add first weather service
      * User Interface
        + Add buttons for sorting and organizing resulting data
      * User Interaction
        + Implement the actual sorting and changing of date ranges and displaying the results
    - Feb. 15 – Feb. 21 (Week 4)
      * Server
        + Add second weather service
      * User Interface
        + Organize interface so it’s easy to use and add data displaying which weather service the data is from and selection of weather services to use
      * User Interaction
        + Filter by the selected weather services and add AJAX responses
    - Feb. 22 – Feb. 28 (Week 5)
      * Server
        + Login Authentication
      * User Interface
        + Login page or form
      * User Interaction
        + Login cookies and success and failures
    - Mar. 1 – Mar. 7 (Week 6)
      * Server
    - Mar. 8 – Mar. 14 (Week 7)
    - Mar. 15 – Mar. 21 (Week 8)
    - Mar. 22 – Mar. 28 (Week 9)
    - Mar. 29 – Apr. 4 (Week 10)
    - Apr. 5 – Apr. 11 (Week 11)
    - Apr. 12 – Apr. 18 (Week 12)
    - Apr. 19 – Apr. 25 (Week 13)
    - Apr. 26 – May 2 (Week 14)
    - May 3 – May 9 (Week 15)
      * May 4
        + Watch Project Videos

# Risks

## WSRP

WSRP standard stands for Web Services for Remote Portlets. It is a network protocol standard that is designed for communication with remote portlets. None of us have used this standard before, and so far can only find limited information related to it. We should be able to mock it out for now, and continue on with the project, as it seems to be a relatively small risk.

## Ranking of Weather Services

Certain weather services that are available are more credible than others. We must research all of them and determine which ones should be used. Auto-Owners claims associates will probably have knowledge of which ones are credible, so we may have to speak with a few of them to find out.

## Extrapolation of Weather Data

Sometimes weather incidents will have occurred near, but not right at certain locations of interest. Decisions must be made as to whether or not the event was close enough to have affected the location that was searched for. Additionally, a weather event may have occurred a two locations on either side of the location searched for, so perhaps some guessing may have to take place using a midpoint formula.

## Weather API’s

One big risk that we have is the fact that we will be gathering information from several different sources. Each one of these sources uses a different weather API. We will certainly pick the most important one out for now, and try to use it with our program. Later though, we must figure out how to query the rest of them as well, and make sense of the information returned.