**Requirements Analysis Document**

Road Closure Tool

CSCI 4712 Senior Capstone Project

Spring 2017

Augusta University

Augusta, GA

Date: 2/17/2017

Version 1

Team Members

JoQuan Osgood

Kyle Zercher

Alex Hutchinson

**Abstract**

This document contains the requirements, analysis, and design artifacts for the Road Closure Tool. The Road Closure tool is a webtool that will enable properly authorized users to create maps of road closures and will allow maps to be saved as a .PDF to allow for easy distribution.

The user will be identified via a unique username and password. An authenticated user will be able to access all of their previously saved maps as well as being able to edit them and create new maps. While editing a map, the user will be able to draw and annotate lines, circles, specific points, and boxes.

This document describes the requirements, analysis, and design of the Road Closure Tool. The rest of this document is structured as follows. Chapter 1 contains the introduction. This chapter presents a brief description of the system. Chapter 2 outlines the functional requirements of the system.

**Table of Contents**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | INTRODUCTION……………………………………………............. | | | 4 |
|  | 1.1 | SCOPE OF SYSTEM………………………………………………… | 4 |  |
|  | 1.2 | OVERVIEW OF DOCUMENT……………………………………… | 4 |  |
| 2 | REQUIREMENTS OF SYSTEM……………………………………. | | | 5 |
|  | 2.1 | FUNCTIONAL REQUIREMENTS………………………………….. | 5 |  |
|  | 2.2 | USE CASES………………………………………………………….. | 6 |  |
|  | 2.3 | USE CASE DESCRIPTIONS………………………………………... | 7 |  |
|  |  |  |  |  |

# Introduction

## Scope of System

Road Closure Tool is a web application for documenting road closures. The only actor is the user, who represents the person in charge of creating, editing, and deleting the documented road closures.

This system keeps track of account information, map information, and the parts of a map that has been edited. All of this is stored in the database. The user can interact with the Road Closure Tool through the web application.

The system includes functionality for verifying the user, as well as map edits such as lines that have been placed and their annotations. All data stores are internal and part of the system. The system relies on the google maps API.

## overview of document

The rest of this document is structured as follows. Chapter 2 outlines the functional requirements of the system. Within this chapter is a list of functional requirements of the Road Closure Tool. It also includes a use case model of those functional requirements. A detailed description of each functional requirement then follows.

# requirements of system

## Functional Requirements

* Login – All users of the system have to be verified before they can use the Road Closure Tool. This function verifies the user and grants them access to other functionality.
* Logout – The user will be able to clear their user session when they are done with the tool so that no one will be able to access their account when they leave.
* CreateMap – The user is able to make a new map for editing. The created map will default to the location based on the user’s ip address.
* SaveMap – This function lets users save the map to the database so that it can be retrieved at a later time. After saving, the user is still able to continue editing the map.
* EditMap – This function allows users to make changes to a map. Changes can include: drawing lines to indicate a road closure, dropping points to indicate a closed intersection, and drawing a box or circle around a large area of road closures. After every edit, the user will be able to annotate the changes.
* SelectMap – This function allows the user to select a map and view details about it.
* PublishMap. – This function is used by the user when the map is finished and ready to be distributed. The map is saved and the user is prompted to save a PDF of the finished map. The user is then redirected back to the ListMapView.
* DeleteMap – This function is for maps that are no longer useful as the maps will be permanently deleted.

## USE CASES



Figure 2.1: use case diagram for Road Closure Tool

## use case descriptions

|  |  |
| --- | --- |
| *Use case name* | Login |
| *Participating*  *actors* | Initiated by User |
| *Flow of events* | 1. The User inputs their username and password into the proper fields on the login form.    1. System responds by verifying login information and logging the user in if the information is correct and displaying the main page. |
| *Entry condition* |  |
| *Exit condition* | * The User is directed to the ListMapView. |
| *Security*  *requirements* | The characters that comprise the password should not be displayed as plaintext in the textbox during entry.   * Prevent SQL query attacks by checking for values that would not be in a user name or password. |

Figure 2.2: Use case description for Login with Success.

|  |  |
| --- | --- |
| *Use case name* | Login |
| *Participating*  *actors* | Initiated by User |
| *Flow of events* | 1. The User inputs their username and password into the proper fields on the login form, but makes a typographical error.    1. System responds by verifying login information and displays an error message indicating an incorrect username or password was used. |
| *Entry condition* |  |
| *Exit condition* |  |
| *Security*  *requirements* | The characters that comprise the password should not be displayed as plaintext in the textbox during entry.  Prevent SQL query attacks by checking for values that would not be in a user name or password. |

Figure 2.3: Use case description for Login with failure.

|  |  |
| --- | --- |
| *Use case name* | Logout |
| *Participating*  *actors* | Initiated by User |
| *Flow of events* | 1. The User selects the “Logout” button on the top right corner of the ListMapView.    1. System responds by exiting user from the system, clearing the session, and displaying the “LoginView”. |
| *Entry condition* |  |
| *Exit condition* | * The User is completely logged out of the system and LoginView is displayed. |

Figure 2.4: Use case description for Logout.

|  |  |
| --- | --- |
| *Use case name* | CreateMap |
| *Participating*  *actors* | Initiated by User and Google Maps JavaScript API |
| *Flow of events* | 1. The User selects the “CreateMap” function from the “ListMap” view. 2. The Systemprompts the to enter a name and description for the map. 3. The User enter the name and description of the new map then clicks “submit". 4. The System creates a map object and saves to database then opens the “EditMapView” page and calls the Google Maps JavaScript API. 5. Google Maps JavaScript API creates an empty map that is displayed on the “EditMapView” page. The User’s map centers to the User’s current locations. |
| *Entry condition* |  |
| *Exit condition* |  |

Figure 2.5: Use case description for CreateMap

|  |  |
| --- | --- |
| *Use case name* | SaveMap |
| *Participating*  *actors* | Initiated by User |
| *Flow of events* | 1. The User selects the “SaveMap” function. 2. The Systemsaves appropriate data to the database. |
| *Entry condition* |  |
| *Exit condition* |  |

Figure 2.6: Use case description for SaveMap

|  |  |
| --- | --- |
| *Use case name* | EditMap |
| *Participating*  *actors* | Initiated by User and Google Maps JavaScript API |
| *Flow of events* | 1. The User selects a drawing on the map.    1. Google Maps JavaScript API makes the drawing editable and highlighted.   3. The User clicks the erase button.  4. Google Maps JavaScript API removes the drawing on the Map. The system removes the data associated with drawing form the Map object.  . |
| *Entry condition* | * The User is directed to EditMapView |
| *Exit condition* |  |

Figure 2.7: Use case description for EditMap (Erase Tool).

|  |  |
| --- | --- |
| *Use case name* | EditMap |
| *Participating*  *actors* | Initiated by User and Google Maps JavaScript API |
| *Flow of events* | 1. The User selects the Annotate tool from the toolbar.    1. Google Maps JavaScript API responds by changing the panning tool to the annotate tool while mouse cursor is on the map.   3. The user clicks any area on the map to make an annotation.  4. Google Maps JavaScript API draws a small textbox with a cursor inside for the user add text.  5. The User can then enter in text in the textbox and selects the Annotate tool from the toolbar to finalize annotation.  6. System creates a row with details about the Annotate with a unique identifier in the MarkerListBox. |
| *Entry condition* | * The User is directed to EditMapView |
| *Exit condition* |  |
| *Security Requirements* | * Prevent SQL query attacks by checking input from User |

Figure 2.8: Use case description for EditMap (Annotate Tool).

|  |  |
| --- | --- |
| *Use case name* | EditMap |
| *Participating*  *actors* | Initiated by User |
| *Flow of events* | 1. The User selects a “unlock” Button on the MarkerListBox for a specific Marking.    1. System changes the lock icon next to Marker selected to the unlock icon. “Edit” buttons appear in all label columns except the “Type” label. “Delete” button appears next to “unlock” button. The edited labels in the MarkerListBox include “Name”, “Start Time”, “End Time”, and “Description”.   3. The User clicks on any of the “Edit” buttons in the columns on the Marker selected.  4. System pops up a modal with a text box for editing.  5. The User then adds the desired text and clicks the “Save” button.  6. System responds by saving the data entered to the specific Marker in its designated Row.  7. The User selects the “unlock” Button to finalize changes to the Marker.  8. System removes the “Edit” buttons and changes the unlock icon to the locked icon. |
| *Entry condition* | * The User is directed to EditMapView and Marker is created |
| *Exit condition* |  |
| *Security Requirements* | * Prevent SQL query attacks by checking input from User |

Figure 2.9: Use case description for EditMap (MarkerListBox-Edit with Save).

|  |  |
| --- | --- |
| *Use case name* | EditMap |
| *Participating*  *actors* | Initiated by User |
| *Flow of events* | 1. The User selects a “unlock” Button on the MarkingListBox for a specific Marking.    1. System changes the lock icon next to Marker selected to the unlock icon. “Edit” buttons appear in all label columns except the “Type” label. “Delete” button appears next to “unlock” button. The edited labels in the MarkerListBox include “Name”, “Start Time”, “End Time”, and “Description”.   3. The User clicks on the “Name” column on the Marking selected.  4. System pops up a modal with a text box for editing.  5. The User then adds the desired text and clicks the “Cancel” button.  6. System closes the modal.  7. The User then selects the “unlock” Button to finalize changes to Marking  8. System removes the “Edit” buttons and changes the unlock icon to the locked icon. |
| *Entry condition* | * The User is directed to EditMapView and Marker is created |
| *Exit condition* |  |

Figure 2.10: Use case description for EditMap (MarkerListBox-Edit with Cancel).

|  |  |
| --- | --- |
| *Use case name* | EditMap |
| *Participating*  *actors* | Initiated by User |
| *Flow of events* | 1. The User selects a “unlock” Button on the MarkingListBox for a specific Marking.    1. System changes the lock icon next to Marking selected to the unlock icon and allows changes on all fields except the “Type” label. “Delete” button appears next to “unlock” button. The edited labels in the MarkingListBox include “Name”, “Start Time”, “End Time”, and “Description”.   3. The User clicks the “Delete” button.  4. System removes the Marker from the MarkerListBox. |
| *Entry condition* | * The User is directed to EditMapView and Marker is created |
| *Exit condition* | * Marker is Deleted from MarkingListBox |

Figure 2.11: Use case description for EditMap (MarkerListBox-Edit with Delete).

|  |  |
| --- | --- |
| *Use case name* | EditMap |
| *Participating*  *actors* | Initiated by User and Google Maps JavaScript API |
| *Flow of events* | 1. The User selects the line tool from the toolbar.    1. Google Maps JavaScript API responds by changing the panning tool to the line tool while the cursor is on the map.   3. The User clicks one end of the closed road.  4. Google Maps JavaScript API draws a line as long, changing the direction of the line as the cursor moves.  5. The User then clicks again at the end of the road.  6. System responds by leaving the line on the map creating a marker object in the MarkerListBox.  7. The User drags the cursor off the map.  8. Google Maps JavaScript API changes the back to the panning tool. |
| *Entry condition* | * The User is directed to EditMapView |
| *Exit condition* |  |

Figure 2.12: Use case description for EditMap with Line.

|  |  |
| --- | --- |
| *Use case name* | EditMap |
| *Participating*  *actors* | Initiated by User and Google Maps JavaScript API |
| *Flow of events* | 1. The User selects the circle tool from the toolbar.    1. Google Maps JavaScript API responds by changing the panning tool to the circle tool while the cursor is on the map.   3. The User clicks the center of the area they want to encircle.  4. System draws a circle that becomes larger as the cursor is dragged away from the initial click.  5. The User then drags the circle to the proper size and clicks again.  6. System responds by leaving the circle on the map and creating a marker object in the MarkerListBox.  7. The User drags the cursor off the map.  8. Google Maps JavaScript API changes the back to the panning tool. |
| *Entry condition* | * The User is directed to EditMapView |
| *Exit condition* |  |

Figure 2.14: Use case description for EditMap-Circle.

|  |  |
| --- | --- |
| *Use case name* | EditMap |
| *Participating*  *actors* | Initiated by User and Google Maps JavaScript API |
| *Flow of events* | 1. The User selects the box tool from the toolbar.    1. Google Maps JavaScript API responds by changing the panning tool to the box tool while the cursor is on the map.   3. The User clicks the center of the area they want to enclose.  4. Google Maps JavaScript API draws a box that becomes larger as the cursor is dragged away from the initial click.  5. The User then drags the box to the proper size and clicks again.  6. System responds by leaving the box on the map and creating a marker object in the MarkerListBox.  7. The User drags the cursor off the map.  8. Google Maps JavaScript API changes the back to the panning tool. |
| *Entry condition* | * The User is directed to EditMapView |
| *Exit condition* |  |

Figure 2.15: Use case description for EditMap-Box.

|  |  |
| --- | --- |
| *Use case name* | SelectMap |
| *Participating*  *actors* | Initiated by User |
| *Flow of events* | 1. The User selects a map in the Listbox on the ListMapView. 2. System retrieves the data about the selected map from the database and displays details in a container on the ListMapView.   . |
| *Entry condition* | * The User is verified. |
| *Exit condition* |  |

Figure 2.16: Use case description for SelectMap.

|  |  |
| --- | --- |
| *Use case name* | PublishMap |
| *Participating*  *actors* | Initiated by User and Google Static Maps API |
| *Flow of events* | 1. The User selects the “Publish” button.   2. System saves the map to the database and converts the map into an image using the Google Static Maps API and then adds the image to a pdf and then prompts the user to download the generated pdf and then changes to the ListMapView. |
| *Entry condition* | * The User is directed to the EditMapView. |
| *Exit condition* |  |

Figure 2.17: Use case description for PublishMap (EditMapView).

|  |  |
| --- | --- |
| *Use case name* | DeleteMap |
| *Participating*  *actors* | Initiated by User |
| *Flow of events* | 1. The User selects a map within a Listbox on the ListMapView to delete.   2. System responds by presenting an indicator on the ListMap View of the map selected.   1. The User selects the “Delete” button.   4. System deletes selected map from database, generates a pop up to let User know delete was successful, and refreshes the ListMapView for Listbox update. |
| *Entry condition* | * The User is logged into System. |
| *Exit condition* | * A refreshed ListMapView is displayed with selected map deleted. |

Figure 2.18: Use case description for DeleteMap.