

System Requirements Specification Template

The Redistrictinator

Client: Geoffrey Weiss



BRICKS

Benjamin Jeremenko

Jacob Philip

Sumanth Neerumalla

Zachary Elliott

Francis Kato

Nathaniel Fuller

Table of Contents

1. Introduction	pg. 2
1.1 Purpose of This Document	
1.2 References	
1.3 Purpose of the Product	
1.4 Product Scope	
2. Functional Requirements	pg. 3
3. Non-Functional Requirements	pg. 8
4. User Interface	pg. 9
5. Deliverables	pg. 9
6. Open Issues	pg. 9
Appendix A – Agreement Between Customer and Contractor	pg. 10
Appendix B – Team Review Sign-off	pg. 11
Appendix C – Document Contributions	pg. 12

1. Introduction

1.1 Purpose of This Document

The purpose of this document is to provide a description of the software that will be developed to redistribute Maryland's 8 districts into evenly populated sections. It is to be used by the customer and developers to define the expectations and requirements for the product.

1.2 References

(n.d.). Retrieved October 19, 2017, from <https://blocks.org/mbostock>

Olson, B. (n.d.). B-Districting. Retrieved October 19, 2017, from <http://blog.bdistricting.com/2016/>

Data.gov. (n.d.). Retrieved October 19, 2017, from <https://www.data.gov/>

Amazon Web Services (AWS) - Cloud Computing Services. (n.d.). Retrieved October 19, 2017, from <https://aws.amazon.com/>

1.3 Purpose of the Product

The purpose of this project is to create and display 8 evenly distributed districts for the state of Maryland, along with details about each district. Currently, Maryland's districts lines may be the result of gerrymandering so this product will be able to correct any lines drawn for political gain.

1.4 Product Scope

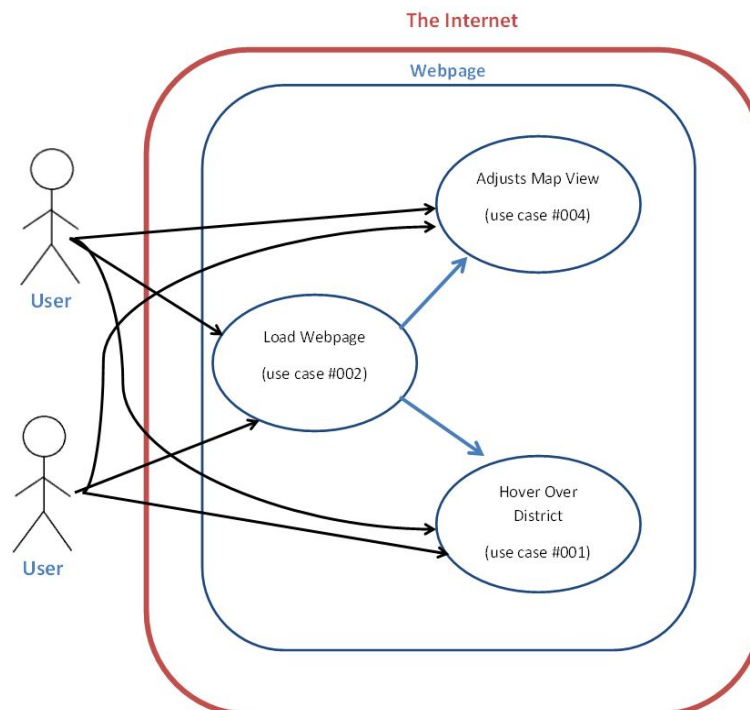


Figure 1.1

The UML diagram shows that the system created is contained completely in the system of the internet. Users will interact through the internet to use functional parts of the system such as hovering over districts and adjusting the map's viewport.

2. Functional Requirements

Our product will have several required functionalities. In general, we are to deliver a web interface product that will allow users to look at the zip codes in the state of Maryland and then generate groups of zip codes to be considered as separate districts. The districts should be generated to be as close to population as possible and be visibly distinct from each other on the web interface. We will leverage the Google Maps API and an AWS backend that will feed data to the front end as necessary.

Number	001	
Name	Mouse-over District	
Summary	The user hovers their mouse over a district on the map and a popup displaying the population, racial diversity, and zip codes missing of that certain district	
Priority	3	
Preconditions	The webpage loaded properly The map loaded properly with the calculated districts drawn	
Postconditions		
Primary Actor	User	
Secondary Actors	Internet	
Trigger	The user moves their mouse over a district	
Main Scenario	Step	Action

	1	The system registers which district is being moused-over and determines the population, racial diversity, and missing zip codes of that district
	2	The system displays the information in a graphic for the user to see
Open Issues	< list of issues awaiting decisions that affect the use case >	

Number	002	
Name	Loads Webpage	
Summary	The user enters the URL to the project webpage and the web page is loaded into the internet browser properly	
Priority	5	
Preconditions	Internet access	
Postconditions	The webpage will have loaded and the generation of the new district lines on the Google Maps map will have begun	
Primary Actor	User	
Secondary Actors	Internet	
Trigger	The user entered the URL to the project webpage	
Main Scenario	Step	Action
	1	The webpage loads into the web browser
	2	The webpage displays the title, descriptions, and area where the Google Maps map will be generated
Open Issues	< list of issues awaiting decisions that affect the use case >	

Number	003	
Name	Loses Internet Access	
Summary	The user at some point after loading the webpage loses internet access	
Priority	1	
Preconditions	The webpage loaded properly	
Postconditions		
Primary Actor	User	
Secondary Actors	Internet	
Trigger	The user loses internet connection	
Main Scenario	Step	Action
	1	The web page and map will still be up but hovering over districts for details will no longer be possible
	2	The user must reconnect to an internet connection
	3	The user must refresh the page and reload the web app
Extensions	Step	Branching Action
	1a	< condition causing branching > : < action or name of sub use case >
Open Issues	< list of issues awaiting decisions that affect the use case >	

Number	004
---------------	-----

Name	Zoom in on Map	
Summary	The user uses the mouse to zoom in and out of the map of the redistricted lines of Maryland, going from street level to being able to see the whole continent	
Priority	2	
Preconditions	The webpage loaded properly The map loaded properly with the calculated districts drawn	
Postconditions	The map's view is adjusted	
Primary Actor	User	
Secondary Actors	Internet	
Trigger	The user scrolls or right clicks or uses the zoom feature present on the Google Maps map	
Main Scenario	Step	Action
	1	The system registers whether the user is attempting to zoom in or out and adjusts the view for the new window to the land
	2	The system displays the map with the new window coordinates
Extensions	Step	Branching Action
	1a	< condition causing branching > : < action or name of sub use case >
Open Issues	< list of issues awaiting decisions that affect the use case >	

Number	005
---------------	-----

Name	Hosting Multiple Users	
Summary	Multiple users loading the webpage at the same moment	
Priority	2	
Preconditions	Internet access	
Postconditions	The webpage loaded properly	
Primary Actor	User	
Secondary Actors	Internet	
Trigger	Each user loads the webpage into their respective web browsers	
Main Scenario	Step	Action
	1	The system treats each user as separate and loads the webpage for each user
	2	Each map is loaded with redistricting lines on the state of Maryland
Extensions	Step	Branching Action
	1a	< condition causing branching > : < action or name of sub use case >
Open Issues	< list of issues awaiting decisions that affect the use case >	

Test 001: Hover your mouse over each district and determine if each district shows a popup menu showing the population, racial diversity, and missing zip codes if any.

Test 002: Go onto a computer and determine that the web page loads properly with the Google Maps map showing when you enter the URL.

Test 004: Enter the webpage and determine that the Google Maps map is capable of zooming in and out of the view initially shown on load.

Test 005: Determine that the web page supports multiple users simultaneously by using two separate computers to enter the webpage and use the functions on the Google Maps map such as zooming and hovering over the districts.

3. Non-Functional Requirements

- **NFR1:** The algorithm for generating the new district lines must be run only after loading the web page {3}.
 - **Test1:** Load the webpage and determine if the generated map loads only after the webpage has loaded.
- **NFR2:** The system must be hosted using a web app {5}.
 - **Test2:** Check that the source code uses a web app to link to the website.
- **NFR3:** The new districts must have boundaries {4}.
 - **Test3:** Visually confirm that the districts are bordered after loading the web app.
- **NFR4:** The new districts must each have a different color {2}.
 - **Test4:** Visually confirm that the districts all have different colors after loading the web app.
- **NFR5:** The algorithm must run to completion in less than a minute {3}.
 - **Test5:** Add a timer in the algorithm code to verify its run time .
- **NFR6:** The web page must use CSS {3}.
 - **Test6:** Check the HTML for the website to determine that CSS is used.
- **NFR7:** There must only be 8 districts drawn {5}.
 - **Test7:** Visually confirm that only 8 districts are shown on the generated map
- **NFR8:** The districts drawn must be contiguous {5}.
 - **Test8:** Visually confirm that each district on the map is conjoined and not separated.
- **NFR9:** The data necessary for the redistricting algorithm is hosted on a web server
 - **Test9:** Confirm that a web server is used through the source code.
- **NFR10:** The populations for each districts should be as close as possible and must not vary by more than 10%.
 - **Test10:** Using the website, determine that the smallest district population divided by the largest district population is no less than 0.90.

4. User Interface

See *User Interface Design Document* for The Redistrictinator.

5. Deliverables

Hard copies of each of the following:

- Systems Requirement Specification - Oct 19
- System Design Document - Oct 24
- User Interface Design Document - Oct 26
- User Manual - (Canceled on schedule)
- Administrator Manual - Dec 5
- Copies of all Biweekly Status Reports - Oct 5, Oct 19, Nov 2, Nov 16,

An electronic file containing the following:

- Systems Requirement Specification - Oct 19
- System Design Document - Oct 24
- User Interface Design Document - Oct 26
- User Manual - (Canceled on schedule)
- Administrator Manual - Dec 5
- All source code - Dec 12
- The executable program - Dec 12
- Any other software required for installation and execution of the delivered program- Dec 12

6. Open Issues

We currently have some open issues that are technical. Since it is the early stages, they mainly involve deciding on the approach we will be using to come up with a product for the client.

1. The details of the algorithm that will divide Maryland into 8 separate, contiguous districts are still unknown. **Estimated Resolution Date: 10/25/2017**
2. We must find a comprehensive and complete source of data that we will use to perform our calculations for this project. **Estimated Resolution Date: 10/22/2017**

Appendix A – Agreement Between Customer and Contractor

Team Members:

Benjamin Jeremenko: Benjamin Jeremenko Date: 10/18/17
 Jacob Philip: Jacob Philip Date: 10/17/17
 Sumanth Neerumalla: Sumanth Neerumalla Date: 10/19/17
 Zachary Elliott: Zachary Elliott Date: 10/19/17
 Francis Kato: Francis Kato Date: 10/19/17
 Nathaniel Fuller: Nathaniel Fuller Date: 10/19/17

Customer:

Geoff Weiss: Geoff Weiss Date: 10/18/17
 Comments: _____

The system to be implemented will be a web app that displays 8 districts in Maryland that have been redistricted to create evenly populated districts. The functional and non-functional requirements are accurate and determine the guidelines of this system. Team 17 and the customer, Geoff Weiss, all agree to the terms provided in this document.

If a change to this document is necessary, the following procedure will be used. The edit to the document will first be discussed in the biweekly meetings. If agreed upon by both parties, the new document will be drafted by the development team, Team 17. Once completed, the finished document will be sent to the customer to be resigned and then resubmitted to the github account.

Appendix B – Team Review Sign-off

All of the following team members have reviewed this document and agree to the content and format.

Team Members:

Benjamin Jeremenko:  Date: 10/18/17

Comments: _____

Jacob Philip:  Date: 10/18/17


Comments: _____

Sumanth Neerumalla:  Date: 10/19/17

Comments: _____

Zachary Elliott:  Date: 10/19/17

Comments: 

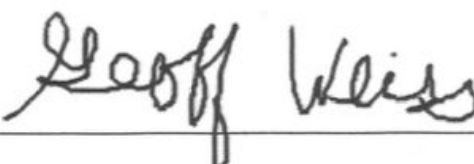
Francis Kato:  Date: 10/19/17

Comments: _____

Nathaniel Fuller:  Date: 10/19/17

Comments: _____

Customer:

Geoff Weiss:  Date: 10/18/17

Comments: _____

Appendix C – Document Contributions

Benjamin Jeremenko: Formatting - User Interface	10%
Jacob Philip: Introduction	20%
Sumanth Neerumalla: Functional Requirements	20%
Zachary Elliott: Non - Functional Requirements	20%
Francis Kato: Deliverables	15%
Nathaniel Fuller: Open Issues	15%