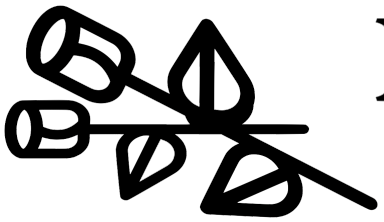
Gerrymander Application: System Requirements

Hyreus LTD.

October 19, 2017



HyreUS LTD.

Contents

1 Introduction

1.1 Purpose of This Document

The purpose of this Systems Requirements document is to be used as a guideline throughout the development of the application. It will list the functional and non-functional requirements of this application and provide detailed information for each requirement. This document will also contain a User Interface section that details the front-end creation of the application. Each deliverable will be listed along with a due date for its submission. Lastly, any open issues will be listed. These issues will be further investigated and discussed as the development process progresses.

1.2 References

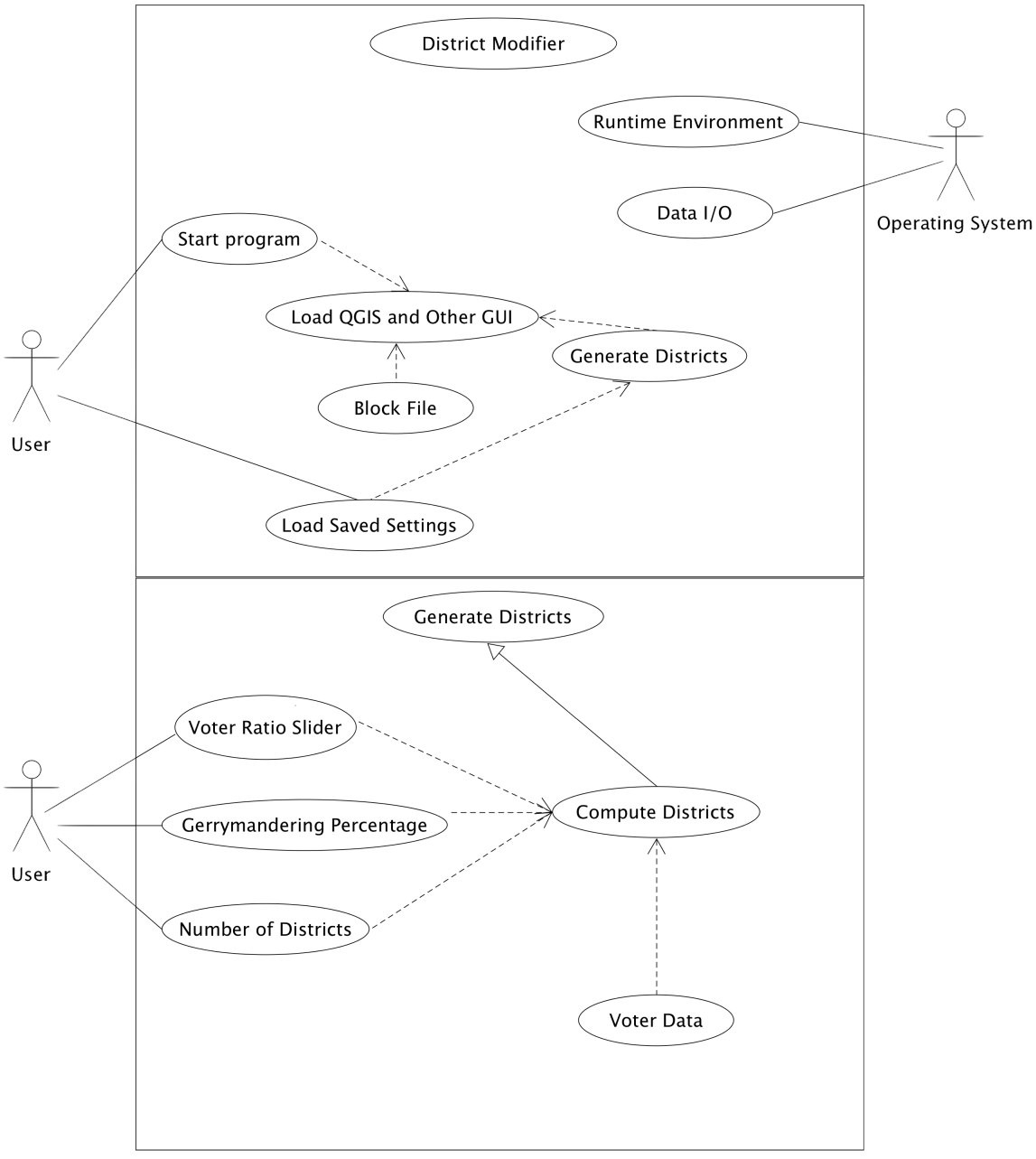
http://bdistricting.com/2010/ http://www.qgis.org/en/site/

1.3 Purpose of the Product

Since the founding of the United States, gerrymandering has been used in order to strengthen one party's chance at winning an election. By splitting up voting districts strategically, parties skew the political landscape of their specific state, giving them an advantage. In a hypothetical situation, the districts would be divided in a way that portrays the actual data collected on registered voters. This program will provide the user with a way to see this hypothetical situation as well as create their own imbalances based on the percentages of party dominance they desire.

1.4 Product Scope

This application will be connected to a database that will contain voter information for the state of Maryland. The database will be created to allow room for more states to be added, but we will be specifically be focusing on the state of Maryland. Using research on gerrymandering, we will use Python coding not only to efficiently gerrymander a state, but also to display it on Google Earth. Once displayed, the user will be given the option to use 2 sliders that will greatly effect the gerrymander display. The first slider will decide whether the user wants the map to be more or less gerrymandered. The second slider will decide whether you want the gerrymander to favor republicans or democrats.


District Modifier
Runtime Environment
Data I/O
Operating System
Start program
Load QGIS and Other GUI
Generate Districts
Block File
User
Load Saved Settings
Generate Districts
Voter Ratio Slider
Gerrymandering Percentage
Number of Districts
Compute Districts
User
Voter Data

2 Functional Requirements

Number	1	
Name	Adjusting to More or Less Gerrymandered	
Summary	This slider will allow the user to dictate the amount of Gerrymandering that they desire (less gerrymandered to more gerrymandered)	
Priority	5	
Preconditions	Working gerrymandering algorithm, database up and running, and connected to User Interface	
Postconditions	User will be able to dictate how much gerrymandering they want	
Primary Actor	User	
Secondary Actors	Database and User Interface	
Trigger	User moves the slider from its default position	
Main Scenario	**Step**	**Action**
	1	User moves slider
	2	Algorithm runs in the background to produce new gerrymandered map
	3	Map display correctly on the interface
Extensions	**Step**	**Branching Action**
	1a	None
Open Issues	1) Setup of Database with state information 2) Gerrymandering Algorithm 3) User Interface Design	

Number	2	
Name	Adjusting to Favor Republicans or Democrats	
Summary	This slider will allow the user to whether they want the gerrymander to favor Republicans or Democrats	
Priority	5	
Preconditions	Working gerrymandering algorithm, database up and running, and connected to User Interface	
Postconditions	User will be able to dictate which party the gerrymander will favor	
Primary Actor	User	
Secondary Actors	Database and User Interface	
Trigger	User moves the slider from its default position	
Main Scenario	**Step**	**Action**
	1	User moves slider
	2	Algorithm runs in the background to produce new gerrymandered map
	3	Map display correctly on the interface
Extensions	**Step**	**Branching Action**
	1a	None
Open Issues	1) Setup of Database with state information 2) Gerrymandering Algorithm 3) User Interface Design	

Number	3	
Name	Adjusting number of districts (between 10-100)	
Summary	This slider will allow the user to choose how many districts they want displayed on the map	
Priority	5	
Preconditions	Working gerrymandering algorithm, database up and running, and connected to User Interface	
Postconditions	User is able to dictate number of districts	
Primary Actor	User	
Secondary Actors	Database and User Interface	
Trigger	User moves the slider from its default position	
Main Scenario	**Step**	**Action**
	1	User moves slider
	2	Algorithm runs in the background to produce new gerrymandered map
	3	Map display correctly on the interface
Extensions	**Step**	**Branching Action**
	1a	None
Open Issues	1) Setup of Database with state information 2) Gerrymandering Algorithm 3) User Interface Design	

Number	4	
Name	Working algorithm in the background	
Summary	The algorithm should be accurate yet efficient, producing a result within 400-500ms	
Priority	5	
Preconditions	Working gerrymandering algorithm, database up and running, and connected to User Interface	
Postconditions	Algorithm will return an accurate display of the gerrymandered districts	
Primary Actor	User	
Secondary Actors	Census data included in the Database	
Trigger	User moves the slider from its default position	
Main Scenario	**Step**	**Action**
	1	User moves slider
	2	Algorithm runs in the background to produce new gerrymandered map
	3	Algorithm returns correct results
Extensions	**Step**	**Branching Action**
	1a	Research algorithm, code in python, and test it using census data
Open Issues	1) Setup of Database with state information 2) Gerrymandering Algorithm 3) User Interface Design	

Number	5	
Name	Saving of slider settings	
Summary	Users should be able to save whichever slider settings they desire (up to 3)	
Priority	2	
Preconditions	Users find a combination of slider settings that they desire, and click save	
Postconditions	They will now be able to reload up to three previous slider settings	
Primary Actor	User	
Secondary Actors	The database and user interface, along with working algorithm	
Trigger	User attempts to save slider settings	
Main Scenario	**Step**	**Action**
	1	User finds desired settings for each slider
	2	User clicks save button on screen and names slider settings
	3	Settings are saved and now able to be reloaded on the screen
Extensions	**Step**	**Branching Action**
	1a	None
Open Issues	None	

Number	6	
Name	Working application through loss of internet connection	
Summary	Application should be fully functional without initial internet connection, and during loss of connectivity	
Priority	2	
Preconditions	Connection is lost or there was never an internet connection	
Postconditions	Regardless, application is still fully functional	
Primary Actor	User	
Secondary Actors	Every part of the system	
Trigger	User loses connectivity	
Main Scenario	**Step**	**Action**
	1	User loses connectivity or starts application without internet connectivity
	2	Application should proceed to keep running efficiently after loss of connectivity
	3	User should notice no difference in using the application
Extensions	**Step**	**Branching Action**
	1a	None
Open Issues	None	

Number	7	
Name	Graceful Failure of Application	
Summary	This slider will allow the user to whether they want the gerrymander to favor Republicans or Democrats	
Priority	5	
Preconditions	Working gerrymandering algorithm, database up and running, and connected to User Interface	
Postconditions	User will be able to dictate which party the gerrymander will favor	
Primary Actor	User	
Secondary Actors	Database and User Interface	
Trigger	User moves the slider from its default position	
Main Scenario	**Step**	**Action**
	1	User moves slider
	2	Algorithm runs in the background to produce new gerrymandered map
	3	Map display correctly on the interface
Extensions	**Step**	**Branching Action**
	1a	None
Open Issues	1) Setup of Database with state information 2) Gerrymandering Algorithm 3) User Interface Design	

3 Non-Functional Requirements

1. Response Time : The application should be able to give a response to the user's input within 400-500ms. The NFR rating is a 5. We will test this by running our animation in concert with a python clock package to test how fast the algorithm runs as well as how fast the display appears.

2. Effectiveness : The performance of our algorithms will produce accurate results in a timely manner. The results will be displayed correctly on the application with the use of sliders. There will be extensive research done on the best algorithms, and using the results of this research, we will come up with the best way to decide how to gerrymander a region. The NFR rating is a 5.

3. Documentation : With each paper requirement needed, it will be reviewed by the team and the customer before being pushed to our shared GitHub repository. Once pushed to the repository, we will submit the documents officially on blackboard as well as emailing the documents to the customer. The NFR rating is a 4.

4. Reliability : The application should to withstand rapid changes to the sliders by the users while being able to display the correct data

corresponding to those changes. Crashes should be handled gracefully. This NFR rating is a 4. We will test this by stress testing the database as well as finding beta users to test the user interface.

5. Scalability : While our main focus is the state of Maryland, the application should provide the ability to add additional states and be able to run the same algorithms on those added areas. The NFR rating is a 3. We will test this by testing the database and application with additional states.

6. Usability : The application will be not only efficient as far as showing the user how to effectively gerrymander a state, but also be educational. User's will find it easy to use the sliders and understand what they are affecting. The NFR rating is a 3. We will test this by doing beta testing and using the feedback to better our project.

7. Transparency : There will be an open dialogue weekly with the customer where problems are discussed as well as progress made on whichever action has been assigned. The NFR rating is a 3. If the point of contact for our group can't physically make the meeting, an email will be sent with all of the information that would be discussed in the "in-person" meeting.

8. Quality : Code written should lead to no fatal errors. The code will be written under camelCase coding standards, as well as using tabs. The NFR rating is a 2.

9. Readability : While the code will be written by six people, it will look seamless as if one person wrote it. There will be consistent coding standards throughout as well as useful comments. The NFR rating is a 2.

10. Security : In our case security will not be as big of a focus, since there will be no personally identifiable information released, as well as this is a very small scale project. The NFR rating is a 1.

4 User Interface

See the User Interface Design Document for the Gerrymandering Application.

5 Deliverables

Hard copies of each of the following:

- Systems Requirement Specification (Due October 19)

- System Design Document (Due October 26)
- User Interface Design Document (Due October 26)
- User Manual (Due November 30)
- Administrator Manual (Due December 5)
- Copies of all Biweekly Status Reports (Bi-Weekly)

An electronic file containing the following:

- Systems Requirement Specification (Due October 19)
- System Design Document (Due October 26)
- User Interface Design Document (Due October 26)
- User Manual (Due November 30)
- Administrator Manual (Due December 5)
- All source code (Due December 12)
- The executable program (Due December 12)
- Any other software required for installation and execution of the delivered program (Due December 12)

6 Open Issues

1. The specific details on the gerrymandering algorithm (equations, run time, etc...).
2. The collection and analysis of the voter census data.
3. Specifics on the aesthetics of the user interface.
4. Whether or not to have a backup for the database, and if so how that will be implemented.
5. The number of fields for each table in the database.

7 Appendix A - Agreement Between Customer and Contractor

See attached document.

8 Appendix B - Team Review Sign-Off

There are no comments as there were no disagreements. See attached document.

9 Appendix C - Document Contributions

1. Jamal Savoy : I took ownership over this document so a bulk of the document is my work. After learning LaTex, I created the outline and began filling it out with the information our team discussed with the customer at our first meeting. I worked on all of the sections and would estimate I did around 90% of the document.

2. Austin DeLauney : I created the Logo, and the UML Diagrams, and helped create high resolution renders of the tables that Jamal created.

3. Ben Kolarik :

4. Brad Harmening : I reviewed the document and made the necessary spelling and grammatical adjustments.

5. Damien Overton :

6. Dylan Demchuk :