**Nosophobic**

**Requirements Specification**

**Version 2.0**

Owen Mitchell

Bartosz Kosakowski

Anthony Mella

Hamid Yuksel

Comp. Sci. 2XB3 – L01

**Table of Contents**

|  |  |
| --- | --- |
| Section | Page(s) |
| Domain | 2-3 |
| Functional Requirements | 3-4 |
| Non-Functional Requirements | 4-5 |
| Development Requirements | 5-6 |

**Domain**

*Nosophobic* is an application designed to provide information about chronic disease indicators (CDIs) in The United States. Using public data collected from the Centers for Disease control and prevention, CDI risk factors will be calculated and used to rank regions.

This application’s domain is primarily concerned with the public, and facilitating education about disease CDIs and their distribution with respect to geography. Stakeholders include the general public and research institutions who are seeking more information on diseases. Additionally, potential commercial stakeholders include medical community, insurance companies and/or advertisers.

Interaction between entities will be limited, as the concerns of the public will be very different from those of the academic or commercial research community. While the application is intended for public casual use, components of the functionality could be useful for the research community, and could be released as an open source tool for visualizing geographic data. However, as the target audience is the general public, their expectations will be treated with the most priority.

The expectations of the typical user include the following:

1. Application accurately show distribution of CDIs
2. Application outputs data in a manner that is easily parsable and visually appealing
3. Application runs within acceptable timescale
4. Application includes appropriate supplementary textual information for certain search criteria

**Functional Requirements**

The core functionality of *Nosophobic* is simple: the application must deliver appropriate information about the CDIs distributed across America based on user selected inputs.

**Inputs:**

The user will specify values for either, or both, of the following:

1. State (i.e. one of the 50 American states)
2. Disease Topic (e.g. Alcohol, Cardiovascular, Oral, …)

**Outputs:**

There will be two information sources outputted for every query with some modification based on the input type:

1. A graphical map of the United States colorized by state, based on the severity of CDIs, which will be referred to as the “heatmap” for the remainder of this document.
2. A table of textual information, which will be referred to as the “information table” for the remainder of this document.

The specific form of output will vary based on the set of inputs received, and will be divided into 3 cases.

**1. User selects state only**

The heatmap will be generated with only the selected state colorized. All unselected states will be colored gray. Information table will be filled with the top three most prevalent disease topics for the selected state.

**2. User selects disease topic only**

The heatmap will be generated with all states colorized on a gradient based on the severity of the CDIs for the selected disease topic.

**3. User selects state and disease topic**

The heatmap will be generated with only the selected state colorized. All unselected states will be colored gray. The information table will be filled with the three most severe CDIs for the selected disease topic and the location of the CDIs will be overlaid on the heatmap.

**4. User selects state and disease topic (safest path)**

A “safest path” will be generated by starting in the specified state and successively moving to the contiguous state with the lease prevalence of the specified disease topic.

**Non-Functional Requirements**

**Reliability:**

**Robustness**

Currently, the web app will only be compliant for US-based results. This is due to the lack of large datasets available regarding comprehensive health records. As well, our web app will be able to specify the location of disease indicators based off of longitude and latitude, making it fairly robust when measuring disease indicators in the United States.

**Testability**

We will incorporate back-end testing on our Java code through the use of junit. We will aim for 100% test coverage, and will employ many test suites to ensure our code works. As well after deployment, we can set up Google analytics to gather traffic data on our site. This data will be helpful in front-end unit testing and can contribute to any adjustments of our back-end.

**Documentation**

Javadocs will be extensively used to generate proper documentation on our back-end and ensure our API is simple to use. The front-end will feature comments relevant towards the ease of maintaining the front-end.

**Performance:**

The application must be able to complete all data manipulation and generate all graphics with 5 seconds at a minimum. Ideally, the runtime of a single query can be reduced below 2 seconds.

**Human-computer interface issues:**

**Usability**

The web app will be designed for ease of use with a simple user interface. This allows for any age group to use the website with little difficulty. Additionally, the information outputted should be easily parsable and visually pleasing.

**Portability:**

**Interoperability**

The web app will be designed to be working for Google Chrome usage at the least. Safari and Firefox full support will come second.

**Tools**

The *Nosophobic* web app will preferably be built as a dynamic web app. As such, we will use Java as our back-end language, Tomcat server, and Spring to build our MVC. For the front-end, we will use JavaScript, HTML/CSS, bootstrap, jQuery, and Ajax.

**Disaster Recovery**

The current disaster recovery plan is through our code being hosted on a remote repository. As well, our multi-branch system ensures that each collaborator has a personal branch, a test branch for the group, and a master branch for app deployment. As well, we will ensure pull requests and code reviews are done before any branch merges are made.

**Development Requirements**

**Quality Control:**

Given the space of inputs, it is feasible to exhaustively test all possible inputs, however only a portion of the functionality can be automatically tested within the timeline of this project. Therefore, the project will be tested in 3 phases:

**Basic Testing**

A testing suite will be constructed to ensure that all data manipulation methods work as intended to be verified with a subset of the data.

**Input Space and application states**

A testing suite will be constructed to ensure that all possible inputs will output without exceptions, and that all possible transitions between application states will work as intended.

**Graphics**

Within the constraints of this project there is no way to automate testing of the graphical output. As a result the validity of the graphical output will be visually tested by one or more team members.

**Functionality Priorities:**

The functionality priorities reflect the expectations of the typical user, and are ranked below.

1. Application accurately show distribution of CDIs
2. Application outputs data in a manner that is easily parsable and visually appealing
3. Application runs within an acceptable time
4. Application includes appropriate supplementary textual information for certain search criteria

**Likely changes and Maintenance:**

**Dataset changes**  
 As the data set used to facilitate this project is public and managed externally to this team, the data structure must be comprised with a separate module and designed for the modification of the data set, while maintain compatibility with the code base.

**Search Parameters**

It may be desirable to increase the specificity of search parameters in the future. To enable this the implementation should allow for the possibility of the new search criteria, by implementing a module for search parameters.

**Performance**

The processing algorithms should be modularized to allow for modification. This allows for performance improvements should they prove necessary to satisfy the performance requirements.

**Heatmap**

The construction of the heatmap should be modularized to allow for modifications in the rendering process. Possible modifications include rendering performance (space or time) to enable portability, changes in region specificity, or the addition of new countries.