Summer 2015

Investigating Effective Methods for Aggregating, Organizing, Storing, Managing, and Disseminating Community Resilience Data

Jonathan A. Kemp Jr
Western Washington University, jonny@betathink.com

Investigating Effective Methods for Aggregating, Organizing, Storing, Managing, and Disseminating Community Resilience Data

By

Jonathan A. Kemp Jr.
Accepted in Partial Completion
Of the Requirements for the Degree
Master of Science

Kathleen L. Kitto, Dean of the Graduate School

ADVISORY COMMITTEE

Chair, Dr. Scott Miles

Dr. Michael Medler

Mr. Tyson Waldo

MASTER'S THESIS

In presenting this thesis in partial fulfillment of the requirements for a master's degree at Western Washington University, I grant to Western Washington University the non-exclusive royalty-free right to archive, reproduce, distribute, and display the thesis in any and all forms, including electronic format, via any digital library mechanisms maintained by WWU.

I represent and warrant this is my original work, and does not infringe or violate any rights of others. I warrant that I have obtained written permissions from the owner of any third party copyrighted material included in these files.

I acknowledge that I retain ownership rights to the copyright of this work, including but not limited to the right to use all or part of this work in future works, such as articles or books.

Library users are granted permission for individual, research and non-commercial reproduction of this work for educational purposes only. Any further digital posting of this document requires specific permission from the author.

Any copying or publication of this thesis for commercial purposes, or for financial gain, is not allowed without my written permission.

Jonathan Kemp

7/21/2015

Investigating Effective Methods for Aggregating, Organizing, Storing, Managing, and Disseminating Community Resilience Data

A Thesis
Presented to

The Faculty of
Western Washington University

In Partial Fulfillment
Of the Requirements for the Degree
Master of Science

By
Jonathan A. Kemp Jr
July 2015

Abstract

Currently there is no comprehensive source of community resilience data. Geographic data is collected by multiple agents and stored using different schemas. In most cases the schemas that store the data do not relate them to concepts of community resilience, or the disasters the data could be associated with. So this begs the question, how can decentralized geographic data be leveraged to facilitate data-driven decision-making about community disaster resilience? This question was answered by completing three related objectives. First a data aggregation was performed, second a schema was created to organize data with respect to components of disaster resilience, and third a data system called WISCkey was developed for storing, managing, and disseminating data over the web.

A data aggregation was performed for two case studies and was developed specifically for the variety of data related to disaster recovery. Subsequently, a schema was developed to organize aggregated data based on attributes of resilience and aggregation outcomes. Technical infrastructure was selected and configured to store, manage and disseminate the organized data.

The result of this research is a web-based application called WISCkey. WISCkey was built using MongoDB, Python, and Bottle and organizes aggregated data. It was developed to store, mange, disseminate, and provide the means to centralize a variety of resilience data. Ultimately these completed objectives permit applying community resilience theory to facilitate data-driven decision-making, and research, in a user friendly way.

Contents

List of Figures and Tables

Chapter 1: Introduction

1.1 Problem Overview

Data-driven decision-making (DDDM) has the potential to assist communities to improve their resilience to disasters, minimize losses and facilitate effective recovery. Currently there is no systematic way, or comprehensive source, for finding data to drive management decisions toward community resilience (Chang 2009; Beran et al. 2008). Recent literature has suggested the need for systems to better manage resilience data (Chang 2009; Huggins 2007; Pradhan 2007; Cutter 2003; Zhang, Zhou, and Nunamaker 2002). It is important to measure and evaluate disaster resilience to understand what went wrong, what went right, and how to evolve practices of improving resilience for future disasters.

A resilient community is one that either does not experience degradation to critical services, or recovers them to similar or better levels in a reasonable time (S. Miles and Chang 2011). In essence, community disaster resilience is the combination of loss and recovery from a stress event within a community. Unless explicitly stated otherwise, the word resilience is used synonymously in this thesis with community disaster resilience.

No organization is specifically mandated with storing, managing, and disseminating resilience data. Currently, most data associated with disaster resilience are stored and managed by the organizations that captured or commissioned those data. The decentralized storage of data within individual organizations creates a problem of knowing what each organization has and how to obtain it. The problem of decentralization makes it cumbersome to use data for decision making and research. Complications for transferring resilience data between the various organizations

arise because each has a different method of capturing and storing those data. Also, there are critical technical challenges which include proprietary, standalone, and segregated disaster management systems (Pradhan 2007; Voigt 1998).

Resilience data can vary greatly in content type, timely availability, and relative usefulness. The spatiotemporal resolution of data needed for monitoring resilience varies greatly based on both the size and intensity of a stress event. Data gathered and used towards DDDM is currently organized based on explicit traits, such as if it pertains to economics, transportation, demographics, etc. Such obvious representations leave less explicit resilience attributes of the data to be filled in ad hoc by managers and researchers of community resilience theory.

Disciplines such as communications and medicine use schemas to organize data based on relationships to illustrate what concepts are relevant to one another (F. Ye et al. 2005; Poser et al. 1983). A useful data schema which displays relationships between data does not currently exist in the study of community resilience. Establishing relationships between resilience data is a necessary first step to measure, evaluate, and research disaster resilience to inform decision making. A theoretical framework for disaster resilience provides the conceptual structure needed to create a functional schema. Such a schema would organize, manage, analyze and synthesize loss and recovery data in a systematic and meaningful way.

Some work has been completed to consolidate and organize resilience data. However, data consolidation efforts have largely been associated with the loss portion of resilience data, and not the recovery portion (National Research Council 1999). This unbalanced focus means that only a partial understanding of resilience is possible with these systems and respective data. According to Chang (2009), recovery is often cited as the least understood aspect of disasters. It is symptomatic that no comprehensive disaster resilience database of multiple disasters exists. Furthermore, several

researchers have declared a need for a comprehensive source of disaster recovery data (Pradhan 2007; Chang 2009; S. Cutter 2003). Because recovery is a large portion of resilience, and is the least understood phase of the disaster cycle, more work needs to be completed toward its understanding. This thesis focuses on the recovery side of resilience data rather than loss.

1.2 Research Question and Objectives

The research presented here seeks to fill in gaps related to work on resilience data systems. Specifically, the research is guided by the following question:

> How can decentralized geographic data be leveraged to facilitate data-driven decision-making about community disaster resilience?

This is important because as researchers and managers attempt to leverage the concepts of resilience in their work, they must be able to find data to support this work. The goal of this research was to develop a schema for organizing, and develop methods for managing resilience data. This goal was accomplished through a series of related objectives. The objectives of this thesis focus on stages of aggregation, organization, storage, management, and dissemination from the lifespan of data as shown in Figure 1. Ultimately, by focusing on these middle stages, the results from these objectives will facilitate the movement of data from capture to the final stages of analyzing, visualizing, and deciding.

- First, a data aggregation was performed on decentralized resilience data at multiple spatial and temporal scales for two case studies.
- Second, a schema was created to organize data with respect to components of disaster resilience.

- Third, a data system called WISCkey was developed which implements the schema for storing, managing, and disseminating disaster resilience data.

The case studies chosen were Hurricane Katrina, which occurred in 2005, and the 2007 Lewis county floods. These events were chosen because they offer a spectrum of size, scale, and impacts. Through the aggregation of important resilience data for each case study, it became apparent what organizations hold data at federal, state, or local levels, as well as how best to access them. By using the data aggregated from the case studies as exemplars, a schema for representing data by resilient attributes was created. The data schema was then applied to develop a prototype application called WISCkey that stores, manages, and disseminates data using concepts of resilience.

1.3 Thesis Organization

The second chapter presents a literature review related to representing community resilience both conceptually and technically. The literature review provides a justification from previous research for the methods and general framework. Chapter three provides details on the case studies, presents the readily available data found, and explains the methods used to complete the research objectives. Chapter four answers the research question, focuses on the analysis, and addresses any errors that may have occurred. Chapter five provides a discussion that synthesizes insights gained from schema and data system development. In addition, the final chapter makes recommendations for future work given the utility and limitations of the research.

Chapter 2: Literature Review

2.1 Literature Overview

The growing complexity of community problems, such as improving resilience, requires knowledge, situational awareness, and more comprehensive designs for managing human activities (Dangermond 2007). A review of work meant to alleviate decentralization of data needs to be performed to evaluate current methods of storage, and understand ways to improve these methods. Part of this review involves understanding what attributes of the data need to be represented to inform decisions and research related to resilience. This understanding can be achieved by examining the data lifespan, which is the flow of data through time, from creation up until the point of informing management and research.

2.2 Community Disaster Resilience Data Lifespan

There are different systematic processes for applying data toward problem solving and decision making. This systematic process can be represented as a data lifespan with reasonably well-defined steps (Provost and Fawcett 2013). Acting (e.g., about policy) based on data is known as data-driven decision-making and is an ideal culmination of transforming data into information (Datnow and Park 2014). In order to understand how data transforms into information, a synthesis of relevant literature is prudent.

Zhang et al. 2002 states that in order to respond to a disaster quickly with management decisions, humanitarians need to collect, analyze, sort, and communicate. Huggins (2007) describes this process as Geoinformatics. By applying Geoinformatics, the turning of spatial data into spatial information by acquiring, processing, management, analysis, and visualization of data is possible. Maceachren et al. (1999) states how the steps of turning data into information are not linear.

Geoinformatics is an iterative process with feedback loops because data, and the patterns of interest in that data, are not fully known prior to being transformed to relevant information (Frawley, Piatetsky_Shapiro, and Matheus 1992; Maceachren et al. 1999). Maceachren et al. (1999) states that the transition of data into information follows iterative steps. These steps are data selection, preprocessing, transformation, data mining, and interpretation. Another systematic way of understanding the use of data through spatial problem solving is known as the Geographic Approach (Artz 2008).

The five steps in the Geographic Approach are ask, acquire, examine, analyze, and act (Artz 2008). The step ask frames the problem geographically (Artz 2008). For resilience data, the event itself provides the location and the problem for which data is required. In order to ask questions about a problem associated with a disaster recovery, it makes sense that the disaster itself first needs to occur. Once the problem is framed, determining what data is needed to answer the research question (Artz 2008). The step acquire either involves direct measurement or aggregating others' direct measurements that have already been captured. Once acquired, data must be examined to determine if it is appropriate to solve the problem asked in the first place (Artz 2008). During the examine step, data is organized, stored, managed, and maybe even disseminated in order to facilitate data analysis. The step analyze is based on a set of methods chosen to solve the problem (Artz 2008). This is when data begins to take the form of information. The final step, act involves visualizing the results from the analysis to drive decisions (Artz 2008).

For the complex task of facilitating data-driven decision making to improve resilience, the above literature can be synthesized to provide a more comprehensive representation of the data lifespan. Based on the literature, the data lifespan for this thesis is represented as ten stages, starting at creation and ending with informing decisions (Figure 1). A hazard event is the point at

which the resilience data lifespan beings. The subsequent stages are as follows: capture, aggregate, organize, store, manage, disseminate, analyze, visualize, and decide. Each stage is specifically defined in Figure 1. Data is analyzed and visualized to develop information in support of the last stage of decision making (Datnow and Park 2014; Wohlstetter, Datnow, and Park 2008; Voigt 1998). The lifespan of resilience data is linear in the sense that it moves from one start point and one endpoint, but is cyclical in the sense that certain stages can cause feedback loops with other stages within the lifespan (Maceachren et al. 1999). The overall flow of data to information, and the feedback loops that exist in the process, are represented by arrows in figure 1.

The data lifespan stages in Figure 1 are used below to organize a review of relevant literature for this study. Capture examines the phases of a disaster and how these phases fit into loss and recovery. Aggregate examines the variety of data, the spatiotemporal variability that exists throughout the data, and how to systematically gather those data. Organize examines the resilience frameworks available to aid in creating a schema. *Storage* examines the technical systems available today in data storage. Management examines best practices and other managerial standards. Disseminate examines current storage, and management systems, that help users interact with the data. The stages of event, capture, analyze, visualize, and decide are out of the scope of this thesis and will not be discussed to provide context only.

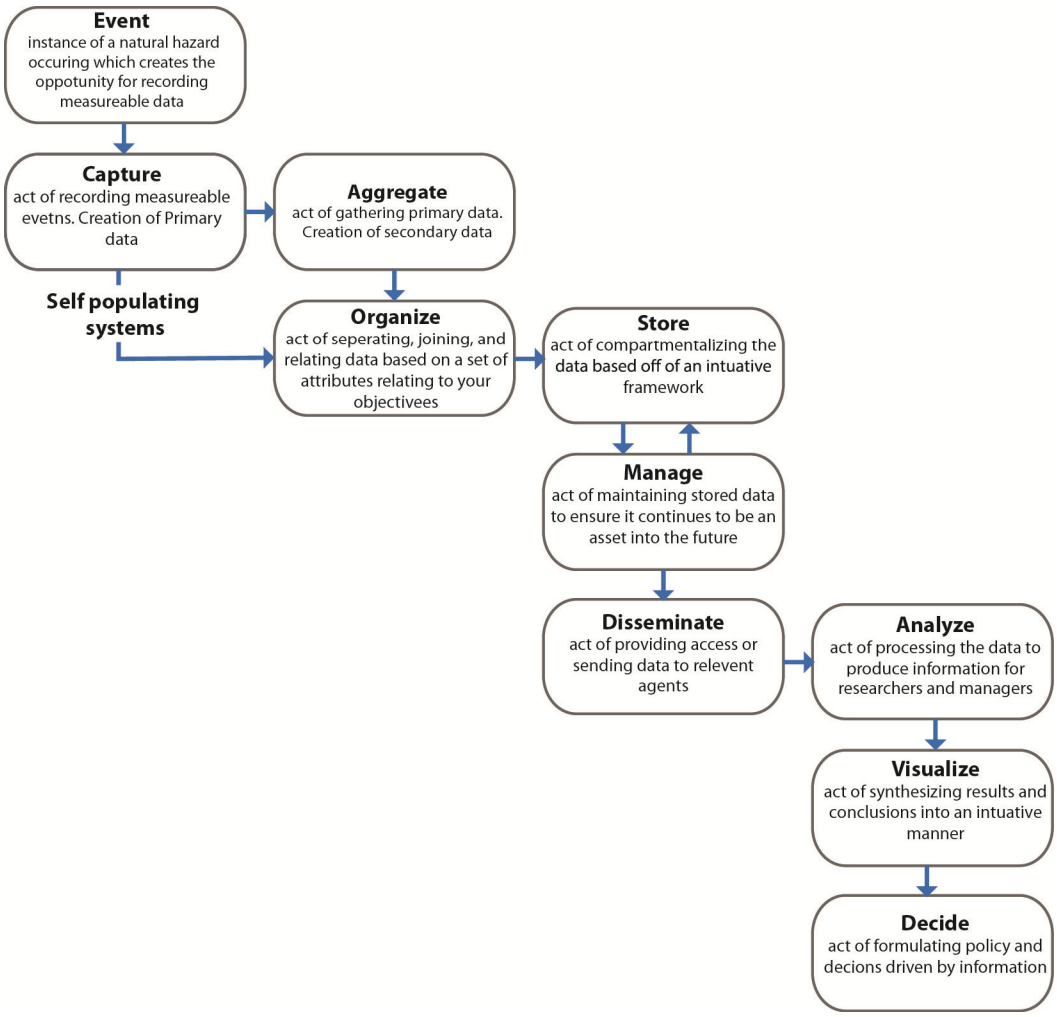

Figure 1 – A conceptual model of the data lifespan to facilitate data-driven decision-making and research about community disaster resilience.

2.3 Capture

Capturing refers to the act of recording measurable events and is most similar to half of the stage of acquire in the geographic approach and geoinformatics. The other half of the acquire stage in the geographic approach is the act of aggregating. The logic here is that to acquire data you must gather it yourself in the field (capture) or acquire it from someone that did (aggregate). While not the first stage of the data lifespan, the stage of capture is when resilience data enters the hands of researchers, whereas the stage event represents the conception itself of resilience data. The

product of the capture stage is raw data. Capturing of resilience data can theoretically occur continuously through all phases of disaster recovery. Different types and topics of data can be captured relative to various phases of disasters.

While there is no consensus in the literature on how to represent the post-disaster phases, they still help to understand what type of data is available when. Haas et al. (1977) proposed four disaster recovery phases: response, restoration, reconstruction, and commemorative reconstruction. Haas et al. (1977) also proposed that each phase is exponentially longer than the one before it. In reality, phases are not discrete and activities can blend between them (Rodrigue 2011). So the phases of a disaster cannot necessarily be distinguished in the capturing of data. Cutter (2003) proposes that the post-disaster phases are rescue, relief, recovery, and reconstruction. The first two phases are focused on reducing loss and the latter two phases are focused on aiding recovery. These phases proposed by Cutter (2003) represent a cycle, not a linear path. Post-disaster activities oscillate between each phase, depending on attributes of the disaster, community, and length of time after the hazard event. As a disaster evolves, the amount of time spent in each disaster phase changes as well (Rodrigue 2011).

2.4 Aggregate

Aggregation refers to the collecting of items into an unorganized whole. As discussed above it is most similar to the stage of acquire in the geographic approach and geoinformatics. For data-driven decision-making, the problem is less the capturing of data and more the aggregation of it. Organizations capture data for various reporting purposes, but that data may not be easily obtainable by decision makers and researchers (Simpson and Katirai 2006). Security and privacy issues can also hinder data access for aggregation purposes (Crittenden 2013; Mills et al. 2008). Another large issue affecting aggregation is proprietary concerns of data (Mills et al. 2008).

Aggregation of data can become expensive when proprietary data must be purchased. Bharosa et al (2010) suggests that organizations are more concerned with obtaining data from others rather than providing their own data.

Pradhan (2007) argues that recovery is rarely integrated into disaster management systems. Even for one particular disaster, recovery data is typically decentralized and often not stored in any formal data system. This situation tends to necessitate a difficult process of aggregating relevant data for particular decisions or research objectives. To Pradhan (2007), what is needed is a standardized enterprise-level GIS for decision support related to disaster recovery. Standardization is ideal, but two factors make standardization complicated. First, the number of organizations involved between and within the stages of the data lifespan (Zhang, Zhou, and Nunamaker 2002; Zheng, Shen, and Tang 2012). Also, in non-disaster situations organizations tend to operate independently of one another (Bharosa, Lee, and Janssen 2010). Because of this independence, the second complication is that the various organizations use unique data standards that originate from many levels of interdependent socio-cultural decisions by organizations (Masten and Obradovic 2008).

Because of these complications, during a disaster situation the systems for data are often pieced together ad-hoc by the various organizations involved (Cutter 2003). An appropriate middleware technology could be applied to facilitate this transfer of data among organizations (Pradhan 2007). An example of middleware for data exchange is the Department of Homeland Security's Unified Incident Command and Decision Support (UCIDS) (U.S Department of Homeland Security 2013). UCIDS is middleware that stakeholders can connect to their existing data infrastructure that allows them to expose certain data to other organizations. UCIDS is geared towards the response phase after a disaster, not longer-term recovery and research.

Bizer et al. (2009) developed a system and set of best practices called the Web of Data for connecting structured data. The Web of Data links data sources through unique resource identifiers. This minimizes issues of privacy and security by creating a virtual web of linked metadata rather than providing the actual data. The Web of Data includes resource description frameworks which outline data and their sources. While exposing the actual data would be ideal, the compounding issues of data volume, variety, privacy, and security make the use of metadata a useful way to aid aggregation of decentralized data across multiple sources and organizations. Such a metadata repository is a viable first step towards creating a centralized repository for resilience data (Beran et al. 2008).

2.5 Organize

A tight relationship exists between the stage of organize and aggregate. To aggregate data in a systematic way you need to do it in an organized way. To organize data it helps to catalog what criteria was used for aggregation. The issues of spatiotemporal variability, variety of data, and capturing organizations involved with aggregation carry a direct connection with organization. In particular, the spatiotemporal attributes of data are important for the organization, because it allows decision makers and researchers to make queries based on time and location.

Geographic information is important for data-driven decision making. Chrisman (1996) breaks geographic information into three components: time, space, and attributes. Chrisman (1996) goes on to explain that in order to measure the change in one component the other 2 must be fixed. In order to understand the spatiotemporal challenges that exist for aggregation, the spatial resolution and temporal resolution must be examined (Peuquet and Duan 1995). To organize disaster resilience data the unit of analysis and the unit of observation both need to be defined. The unit of analysis is the larger phenomena that is being studied, and the unit of observation is the

scale at which the data was collected (Ye and Rey 2013). For example, it might be desired to research how an entire neighborhood is recovering after a disaster event, but the data observed is for each individual. In this case, the entire neighborhood would be the unit of analysis, but the individuals would be the unit of observation. In a way, the unit of analysis is the sum of the units of observations. Ye and Rey (2013) use the elements time, space, and attributes for the different units of analysis and uses local, meso, and global as the units of observation.

The units of observation for spatiotemporal information are important to organize and structure data systems. However, to make data useful for decision-making and research, data should be organized in a way that is tied to the problem or decision domain. This requires a domain-specific conceptual framework to determine what elements are relevant. Indicators are made up of metrics or measures derived from data (Freudenberg 2003). Indicators can be used to understand the spatio-temporal phenomenon of disaster loss and recovery through indirect measurement.

Most conceptual frameworks in the disaster literature were developed to assist in identifying what indicators, metrics, and data are important for understanding the respective topic of disasters. Simpson and Katirai (2006) developed a framework for quantifying vulnerability to hazards. Cutter et al. (2003) uses an additive model to produce composite social vulnerability scores for specific regions. Brown et al. (2010) and Chang (2009) establish indicators for evaluating recovery from disasters. Miles and Chang (2011) and Cutter et al. (2010a) propose separate frameworks for comprehensively representing resilience.

For organizing resilience data, it is important to examine the literature surrounding different conceptual frameworks of community resilience. A composite framework will provide a functional set of indicators and metrics to organize disaster resilience data. The frameworks in the literature have commonalities but also vary noticeably with respect to chosen indicators, the metrics they are

comprised of, and the data used to characterize each metric. There is not one universally accepted community resilience framework (Cutter et al. 2010b) and so a specific choice must be made from the alternatives for the purposes of organizing resilience data.

Bruneau et al. (2003) put forward one of the earliest frameworks for resilience in the disaster literature. Their framework represents resilience as four components: technical, organizational, social, and economic, with each component having quantifiable traits of robustness, redundancy, resourcefulness, and rapidity. Cutter et al. (2010a) breaks resilience into components of social, economic, institutional, infrastructure, and community capital. There are multiple indicators within each component. For instance, social capital is comprised of education equity, age, transportation access, communication capacity. Alternatively Miles and Chang (2011) organize resilience into the three components of physical built environment, economics, and personal capital (S. Miles and Chang 2011). Norris et al. (2008) examines disaster resilience from a psycho-social perspective and introduces the idea of well-being as the ultimate goal of resilience. Norris et al (2008) connects personal and social well-being to Bruneau et al.'s resourcefulness, robustness, redundancy, and rapidity, while adding adaptive capacity and quality of life. Miles (2014) separates static resilience into four constructs of well-being, identity, services, or capitals (WISC), with 27 variables in total characterizing the constructs.

The WISC framework proposed by Miles (2014) is the most comprehensive of the frameworks in the literature, incorporating components of Cutter et al. (2010a), Norris et al. (2008) and Bruneau et al. (2003), among others. WISC is more complex than other frameworks in the literature and thus more difficult to operationalize. However, this complexity is necessary when trying to understand a concept as complicated as community resilience. The theoretical framework of Miles (2014) allows for the most detail and theoretical grounding for organizing resilience data,

which is essential for informing decisions and research. The framework does not currently specify a list of data sources to use as indicators of each component, whereas some frameworks do (e.g., Cutter et al. 2010a). This data flexibility allows for a more dynamic organizational schema.

2.6 Store and Manage

Store in the data lifespan refers to the implementation of an organizational schema with technical infrastructure. Manage refers to the upkeep of such a system through database administration and associated best practices. An organizational schema for disaster resilience data becomes more useful when technical infrastructure is in place to support storage and management. The stages of storage and management are cyclical in the lifespan because they are continuously occurring throughout decision-making and/or research. When data moves along to further stages, where it is transformed into information, the raw data is still (ideally) being managed. Furthermore, data may need to be reorganized and stored using a new schema as new means of representing resilience data is needed. While the stages of store and manage are technically separate in the data lifespan, because they often happen concurrently it makes the most sense to discuss them together.

A computer database can be thought of as digital data stored with an explicit data model and structure. A database management system (DBMS) is software, or a stack of software, designed to manage a database and facilitate dissemination of data. There are two main types of data models present in the literature today, relational and non-relational. Prior to discussing these alternative database models, several basic principles of DBMS should be discussed.

There are basic guiding principles involved in storage and management of data. All interactions with a database can be considered transactions in which the activities of one user are isolated from all concurrent activities (Haerder and Reuter 1983). Since applications tend to have multiple users all at once, data must be indivisible and concurrent (Haerder and Reuter 1983; Fowler

and Sadalage 2013). To enforce indivisibility traditional data models ensure that every transaction has properties of atomicity, consistency, isolation, and durability (ACID) (Haerder and Reuter 1983). ACID transactions are a classic strategy for determining if a database is effective and that each transaction is a single logical expression (Barthomew 2010). A newer counterpart to ACID is "basically available, soft state, eventual consistency" (BASE) (Fowler and Sadalage 2013; Bartholomew 2010). BASE emphasizes higher availability of data and allows for approximate outputs (Brewer 2000). ACID and BASE transactions are straightforward to enforce over a non-networked system. When designing a distributed web-service additional properties of consistency, availability, and partition tolerance (CAP) are desired in these transactions (Gilbert and Lynch 2000). CAP Theorem, also known as Brewer's Theorem, is a conjecture stating it is impossible for a database to have all three traits of CAP simultaneously (Gilbert and Lynch 2000).

For representing disaster resilience data it is important to consider what aspects of CAP are desired and whether ACIDS or BASE transactions are appropriate. For resilience data the DBMS model that makes the most sense depends on the use case. If the DBMS is to be used in addressing loss then an ACID model may be suitable. Whenever extremely valuable resources are at stake, such as money or human lives, strong consistency and accuracy are critical (Fowler and Sadalage 2013). For example, emergency responders need up to date locational information in order to find people in need of assistance. If the use case is more for the recovery side of disaster resilience, then eventually consistent data is acceptable in exchange for a simpler model. For example, for researching resilience it won't matter if the addresses of parcels are out of date at that very instant, as long as they are eventually consistent (within a reasonable timeframe). While exceptions do exist, ACID is generally associated with relational models and BASE is associated with non-relational models (Fowler and Sadalage 2013).

The long-standing approach for storage and management of data is the relational data model (Bartholomew 2010; Leavitt 2010). Codd (1970) is credited with the invention of the relational data model and the twelve traits required to be considered such. Many modern RDBMS now need only conform to at least two of the traits established by Codd to be considered relational. An RDBMS stores data in tables with rows and columns. A row is an entity, and a column is its associated attributes. Data in a table can be related to another table of data according to common attributes (ORACLE 1995; Leavitt 2010). Relational operators must also be present which enable relationships to be tested between two entities. The presence of relationships and their operators makes performing queries on the data easy and efficient. Currently the widely used language for querying a RDBMS is the Structured Query Language (SQL) (Deutsch 2013). The conformity to SQL makes moving from one type of RDBMS to another easy because the application code querying the database will largely not have to be rewritten. There are many variations of commercial and open source RDBMS such as OracleDB, SQL Server, MySQL, PostgreSQL, etc. The rigidity of the relational model is intentional and allows developers to maintain large amounts of data in persistent, concurrent, integrated, and standard ways (Fowler and Sadalage 2013). Critiques of the relational model are that it does not scale well over the web, is too rigid to handle diverse datasets (Stonebraker 2010), and the interaction style is unproductive (Fowler and Sadalage 2013).

With the surge of data that has been seen over the last several years, new types of non-relational data models are being developed and used for storage and management (Fowler and Sadalage 2013). Non-relational DBMS are most commonly referred to as NOSQL (Bartholomew 2010; Leavitt 2010). Originally this was due to the fact they did not use SQL to query the database, but now they commonly refer to non-relational DBMSs that run well on a network, are open-source, and are schemaless (Fowler and Sadalage 2013; Pokorny 2011; Leavitt 2010). It is important to note that while considered schemaless the non-relational model is not without structure, but it is just less

rigid than traditional relational models. The performance gains of NOSQL DBMSs are debated in the literature. Some researchers claim the same performance gains can be achieved using the relational model by eliminating unnecessary overhead (Stonebraker 2010; Pokorny 2011). It is possible however, that the relative performance is more dependent on the use case (Bartholomew 2010; Fowler and Sadalage 2013; Leavitt 2010), rather than being a universal trait.

Unlike the relational model, there are four alternative approaches to non-relational DBMS: key-value pair, column, document, and graph (Rebelic 2013). Table 1 lists the four approaches to non-relational data models and examples of associated DBMS software available today. A strength of the relational DBMS is derived from the fact that many operate similarly and share a common query language; the same is not true for non-relational DBMSs. Major differences exist between the alternative non-relational DBMS approaches (Bartholomew 2010). The decision as to which approach to the non-relational data model is appropriate for storing and managing disaster resilience data is important because it will be harder to switch later in development. For this reason, it is necessary to examine the literature for each approach in order to determine which one can best represent disaster resilience data.

Data Model	DBMS	Developer
Document	MongoDB	10gen
	SimpleDB	Amazon
	CounchDB	Apache
	Jackrabbit	Apache
Graph	Neo4j	Neo Technology Inc
	AllegroGraph	Franz Inc
	BigData	Mark Logic
	ArangoDB	ArangoDB
Key-Value	Aerospike	Aerospike
	Riak	Basho Technoligies
	Tokyo Cabinet	FAL Labs
	Hibari	Hibari developers
Column	Cassandra	Apache
	Hbase	Apache
	Big Table	Google

Figure 2 – Non-relational data model approaches and associated DBMS software.

A key-value non-relational data model is simple in design. Key-value uses a hash table containing a list of keys and a value associated with that key (Rebelic 2013). Key-value is an easy design to implement, but its simplicity makes it possible to become inconsistent if your dataset grows large enough (Kumar and Checker 2013). Disaster resilience data in this model could be a particular disaster as a key and the value would be a description of the event, chunk of information, or attribute of resilience.

Column data models are similar to key-value in that they use keys, but each key points to multiple columns intended to be spread across multiple machines (Rebelic 2013). A relational DBMS uses rows for reading and writing data, but a column DBMS uses columns. Because of the way a column is stored on disk (as compared to a row) operations performed on them are much faster (Kumar and Checker 2013). Conceptually, resilience data in a column data model is similar to the relational model. There is one large table instead of multiple linked tables. Within the table each

row could be an event, with each column being an attribute of that event. This conceptual table is not actually stored as such; instead it is stored in a single long text string. For example, disaster A, disaster B, disaster C: location A, location B, location C: indicator A, indicator B, indicator C: etc.

Instead of tables with rows and columns, a graph data model uses nodes that are connected by edges. Both edges and nodes contain attributes and could be stored as entities. The graph data model structures are useful for modeling entities that have strong relationships with one another (Kumar and Checker 2013). For disaster resilience data, each element of resilience, and disaster could be a separate node. Each node would be connected to whichever other node it was relevant to, with an edge having its own attributes describing the relationships as well. This is a very intuitive approach to data management.

The document data model is very similar to key-value stores, but unlike normal key-value stores they allow for nested documents. This nesting allows for more complexity when designing an organizational schema (Rebelic 2013; Kumar and Checker 2013; Cattell 2011). DBMSs that use the document data model generally support multiple types of documents and the ability to query collections based on multiple attributes (Cattell 2011). The best use case Cattell points out for a document data model approach is one where there will be multiple objects and each object has multiple different attributes associated with them (Cattell 2011). An example for disaster resilience data could be a document for each piece of data; within each document is attributes organized by key-value pairs. Two examples of key-value pairs included in the list could be event (e.g., Hurricane Katrina) and associated characteristic of resilience (e.g., economics).

Because of its schemaless nature, a non-relational model has the most advantage in managing and storing disaster resilience data and is the approach adopted for this study. The volume of disaster resilience data is not necessarily larger than what relational models are capable

of handling. However, the variety of data contributors and the spatiotemporal complexity make the use of the non-relation model for storing and managing resilience data worthwhile. Community resilience theory is constantly changing and so are the associated frameworks. There is a variety of content types, such as jpegs, docs, pdfs, etc. and a variety of researchers and organizations capturing data, all this equates to a proliferation of data (Voigt 1998). As suggested above in subsection 2.4, the schema needs for resilience data will contain different resilience attributes. The non-relational data model is useful if a lot of semi-structured data is present, which is the case with resilience data (Bartholomew 2010; Leavitt 2010).

2.7 Disseminate

If resilience data is going to support decision making and research it will need to be readily available. Disseminate is the stage of providing easy access to stored and managed data. Effective dissemination allows researchers and decision makers to conveniently access the storage mechanisms in a variety of ways to facilitate their own data aggregation needs. According to an Executive Order issued by President Barrack Obama (2013), disseminated data should be machine readable and open to the public.

Web services are a way in which different software systems exchange information with each other (Cerami 2002). Web services tend to use a standardized XML messaging system and are not necessarily tied to any single operating system or programming language. These traits make cross platform communication of web services possible (Cerami 2002). Multiple examples of web services can currently be found to support data exchange, but none are developed for disseminating data based on its attributes of resilience. The Bureau of Economic Analysis (BEA), the Bureau of Labor Statistics (BLS) data page, and Environmental Protection Agency (EPA) Data Finder are all examples of web service functionality that can be used as examples for developing a web service for disaster

resilience data. The BEA's mission it to provide insight into the American economy by providing the most up to date statistics on economic accounts (http://www.bea.gov/itable/index.cfm). The BLS measure labor market activity, working conditions, and price changes in the economy and disseminates that information (http://www.bls.gov/data/). The EPA's goal is to protect human health and the environment. Additionally the EPA offers a data finder (http://www.epa.gov/data/).

Simple Object Access Protocol (SOAP) and Representational State Transfer (REST) are two web-service protocols used for requesting information between two client computers. Both SOAP and REST are viable options for disseminating resilience data between potential users. SOAP is the older of the two methods and essentially transmits XML-encoded messages (zur Muehlen, Nickerson, and Swenson 2005). REST exposes a specific set of resources through a series of HTTP methods such as get, post, put, or delete (Mulligan and Gracanin 2009). REST generally is well suited for basic applications, whereas SOAP offers flexibility and advanced features suited for enterprise applications (Pautasso, Zimmermann, and Leymann 2008). Infrastructure for storing and managing organized resilience data will already be in place by using a DBMS, the resources being stored will just need to be made available for dissemination. Because of the simplicity of REST web services, it is an appropriate way to easily disseminate disaster resilience data organized, stored, and managed in a chosen DMBS (Pautasso, Zimmermann, and Leymann 2008; Mulligan and Gracanin 2009; zur Muehlen, Nickerson, and Swenson 2005).

In typical use cases, a new web framework applying either SOAP or REST protocols is not developed for every application. A web framework is a collection of packages that allow developers to skip tasks like handling protocols, sockets, or other tedious code (Piercy 2014). There are two main types of web frameworks: a full-stack framework and a basic framework (Piercy 2014). The difference is that full-stack frameworks come with pre-selected technology to implement a web

service. Technology that might be included in a full-stack framework include a database, template engine, AJAX toolkit, etc. (Piercy 2014). A basic web framework runs as its own independent process allowing one to introduce other components, such as any DBMS deemed appropriate to run on top of the framework (Piercy 2014). This thesis requires flexibility in choosing specific components to alleviate decentralization of resilience data. A basic web framework is appropriate because it allows for such flexibility, and would create an application server with only the simple functionality required.

2.8 Analyze, Visualize, Decide

At the end of the dissemination stage of the data lifespan, disaster resilience data would need to be analyzed, visualized, and used to make decisions. However, these stages are out of scope for this thesis. Future work will need to be completed to build upon the understanding of these out of scope stages of the data lifespan. With an understanding based on the current literature of organizing, storing, managing, and disseminating a useful research framework was devised. This research framework was used to accomplish the research objectives in order to answer the research question.

Chapter 3: Research Approach

3.1 Research Approach Overview

The literature provides a solid foundation to better represent community resilience data. The objectives of this thesis aim to facilitate the lifespan of resilience data leading to information. The research question of how to better represent community resilience data in a centralized way can be answered by completing the three subsequent research objectives.

1. *Aggregation:* A data aggregation was performed for two case studies and was developed specifically for the variety of data related to disaster recovery. The aggregation reveals a systematic way to gather data at various spatiotemporal resolutions, from various sources, and in various formats.
2. Organization: Subsequently, a schema was developed to organize aggregated data based on attributes of resilience and aggregation outcomes. The schema represents the data with respect to components of community resilience. More specifically the schema represents data in terms of the WISC community resilience framework (Miles 2014), which is the most comprehensive such framework to date. The schema also facilitates characterizing and structuring data to add meaning with respect to important elements of communities, disasters, and resilience.
3. Storage, Management, and Dissemination: Technical infrastructure was selected and configured to store, manage and disseminate the organized data. The DBMS software MongoDB was used for storing and managing data. Dissemination of data is facilitated using a Bottle web framework. Both systems are combined into an application called WISCkey which is described in this chapter. WISCkey provides a non-relational data model with principles of BASE being served via a REST service and coded with the Python programming language. WISCkey earns its name by being a key step in representing decentralized data by applying the schema for operationalizing the theoretical framework WISC.

3.2 Aggregation

As mentioned in the literature review, one of the most comprehensive community resilience frameworks to date is the WISC framework (Miles 2014). Due to its comprehensiveness WISC was chosen as the mechanism to guide aggregation of resilience data. More specifically, data

representing capital and it's variables of social, political, human, economic, built, and natural capitals was the focus of aggregation. These variables of capital represent half of the community resilience component of infrastructure (Miles 2014) The construct of capital was chosen because services derived from capital create the foundation for achieving the end goal of community well-being (Miles 2014). Because capitals can be viewed as the foundation of communities and resilience, it is an appropriate starting construct for guiding the aggregation of resilience data.

A capital within the Framework of WISC is a community asset which is used to subsequently obtain additional assets, or derives a service. In a sense, capital is the building block of community resilience. Seven variables of the construct of community capital were adopted in WISC by synthesizing other community resilience frameworks. These community capitals are social, political, human, economic, built, natural, and cultural.

Social capital refers to the bonds that hold people and groups together. Social capital is directly linked to cultural capital, which is the traditions and languages within a community, and together these capitals act as a cohesive force. Political capital reflects on the access to power via political channels and the ability to contribute to the community. Human capital is the skill and abilities of the people inside a community, especially those in leadership positions. Natural capital refers to the natural resources such as amenities, natural beauty, timber, oil, etc. Economic capital is the resources that can be used to invest in the development of businesses, and supports civic and social entrepreneurship. Built Capital is the infrastructural interface between natural capital and all other capital variables, built capital controls the access to natural capital from all other capitals.

Together the variables for the construct of capital, provides a reasonable stepping stone in understanding community resilience. Another construct of the WISC framework is geography—space, time, and scale. Geography strongly influences the processes and outcomes of community

resilience, and therefore the resilience data that can be aggregated. A proper aggregation will search for community capitals by resolutions of space, time, and scale.

For community resilience data, aggregation can happen at various spatial resolutions. As seen in the literature, units of observation are proposed as individual, local, meso and global (Ye and Rey 2013). These categories proposed by Ye and Rey (2013) are not finite enough and can be interpreted differently by an individual's perception of scale. More specific units of observation are proposed for this aggregation, and later on for organization. These units include a combination of established jurisdictional boundaries and census units. The units are national, state, metropolitan statistical area, county, city, neighborhood, census designated area, and individual (e.g., households or businesses). Under Ye and Rey's classifications county might be considered local by one researcher and meso by another. By explicitly stating the boundaries themselves as the units this interpretation is taken out of the mix.

Much the same way space was delineated into human constructed segments, time in the aggregation was divided into units of observation of increasing resolution. These units of observation can either be taken as totaled, yearly, quarterly, monthly, weekly, daily, hourly, or more than once per hour. These temporal breaks are self-evident, except for perhaps totaled resolution. Totaled resolution data represents a cumulative snapshot of an event. For example one might provide a cumulative dollar amount associated with damage from an event since it occurred.

The spatiotemporal resolutions provide a structured way to aggregate data based on similar units of observation to ensure the aggregation is systematic across time and space. The aggregation was used to gather resilience data for two very different scale natural disaster events. Through aggregating resilience data for the two events, insight was gained on how data access changes

based on event size. After aggregation was complete it was organized and disseminated, completing the second and third objectives of this research.

The two events chosen for aggregation were the 2007 Lewis County flood and Hurricane Katrina in 2005. These two events were chosen for several reasons. First, these disasters occurred at different spatiotemporal scales, allowing for an evaluation of what types of resilience data are available for disasters of differing sizes. For example, Hurricane Katrina was an event which grabbed national attention, whereas the 2007 Lewis County Floods were more a regional event. Selecting two events of such differing sizes helps to reveal what attributes of data are present for large and small events. Second, both disasters occurred within 2 years of each other, which should mitigate for any change in data monitoring or dissemination technologies that occur over time. Third, enough time has elapsed from both events that each should have had opportunity for organizations to capture disaster resilience data along all the phases of the disaster cycle. It is easy to see the stark contrast of both event sizes and economic impacts associated with them. This difference between case studies was intentional so that nuances of data availability could be discovered throughout the aggregation process. The dataset gathered represents a potentially high variability in spatiotemporal and resilience attributes, which required an organizational schema.

To aggregate resilience data for every WISC capital, at each spatial scale affected for both events, was out of scope for this thesis. For this reason, one example jurisdiction was chosen for aggregating data at each defined spatial resolution. This process was applied to both events using a spatial resolution hierarchy. For Hurricane Katrina data aggregation was attempted for each WISC capital for the United States, Louisiana, New Orleans-Metairie-Kenner Metropolitan Statistical Area Orleans Parish, New Orleans city proper, Uptown, and an individual that lives within Uptown. For the Lewis County flood, an aggregation was attempted for each WISC capital for the United States,

Washington, Lewis County, Centralia and an individual that lives in Centralia proper. For both events certain spatial resolutions were intentionally left out. For Lewis County there is no metropolitan statistical area or neighborhood, because of the city's small size. To better understand the context of the data about the two disasters, a closer look at each's geography and characteristics is described below.

In December of 2007, record rainfall occurred in the Willapa Hills from a storm coming in from the Pacific Ocean causing flooding throughout Lewis County, Washington. Preliminary damage from the 2007 floods totaled $166.1 million, with the cost rising to $510 million a year after the event which made up about 25% of the total FEMA claims for the state of Washington in 2007. Despite being a devastating event for the county, flooding is not uncommon in the area, with records showing significant floods at least seven times since the 1800s, of which the 1933, 1986, 1997 and 2007 floods all exceeded the 100 year flood level (Lewis County 2014).

Lewis County is a rectangular county in the southwest of Washington State. With a land area of 2,402.80 square miles and is ranked as the 6th largest county in Washington as shown in Figure 3. Lewis County's population is roughly 76,000, giving it a population density of 31.63 people per square mile. This makes Lewis County the 22nd most densely populated county in Washington State out of 39 total counties. The largest town within Lewis County is Centralia with a population of 16,440. The largest job sectors in descending order within Lewis County are government, manufacturing, healthcare and retail. When added with all other smaller business sectors Lewis County paid $787,405,854 in wages during 2010 (State of Washington 2011).

Table 1– Population statistics for Lewis County from the 2011 Washington Databook

Components of Population Change		City	Population
2010-2011		Centralia	16,440
Estimated Births	956	Chehalis	7,310
Estimated Deaths	778	Morton	1,125
Natural Increase	179	Mossyrock	760
Net Migration	366	Napavine	1,780
Total Population Change	545	Pe Ell	635
Marriages in 2010	522	Toledo	725
Divorces in 2008	325	Vader	625
		Winlock	1,340

Table 2- Employment statistics for Lewis County from the 2011 Washington Databook

Industry	Employers	Average # of Employees	Percent of Total	Wages Paid	Percent of Total
Agricuture, Forestry, Fishing & Hunting	139	1,246	5	38,569,968	5
Mining	14	181	1	9,033,319	1
Utilities	-	-	-	-	-
Construction	236	865	4	34,188,104	4
Manufacturing	119	2,772	12	124,143,602	16
Wholesale/Retail Trade	337	3,850	17	100,938,460	13
Transportation & Warehousing	89	1,028	4	35,278,968	5
Information	18	173	1	6,815,485	1
Financial, Industrial, Real Estate	114	630	3	18,204,228	2
Professional & Tech Services	99	391	2	13,935,748	2
Mgmnt of Companies and Enterprises	-	-	-	-	-
Administrative & Waste Services	74	691	3	16,799,575	2
Educational Services	16	85	0	1,851,021	0
Health Care and Social Assistance	144	2,862	12	109,428,362	14
Arts, Entertainment, Recreation	26	320	1	3,863,787	1
Axxomodation and Food Services	178	1,882	8	28,282,998	4
Other Services	770	1,112	5	18,132,687	2
Government	90	4,926	21	195,973,743	25
Not Elsewhere Classified	14	369	2	31,965,799	4
Total	2,474	23,380	100	787,405,854	100

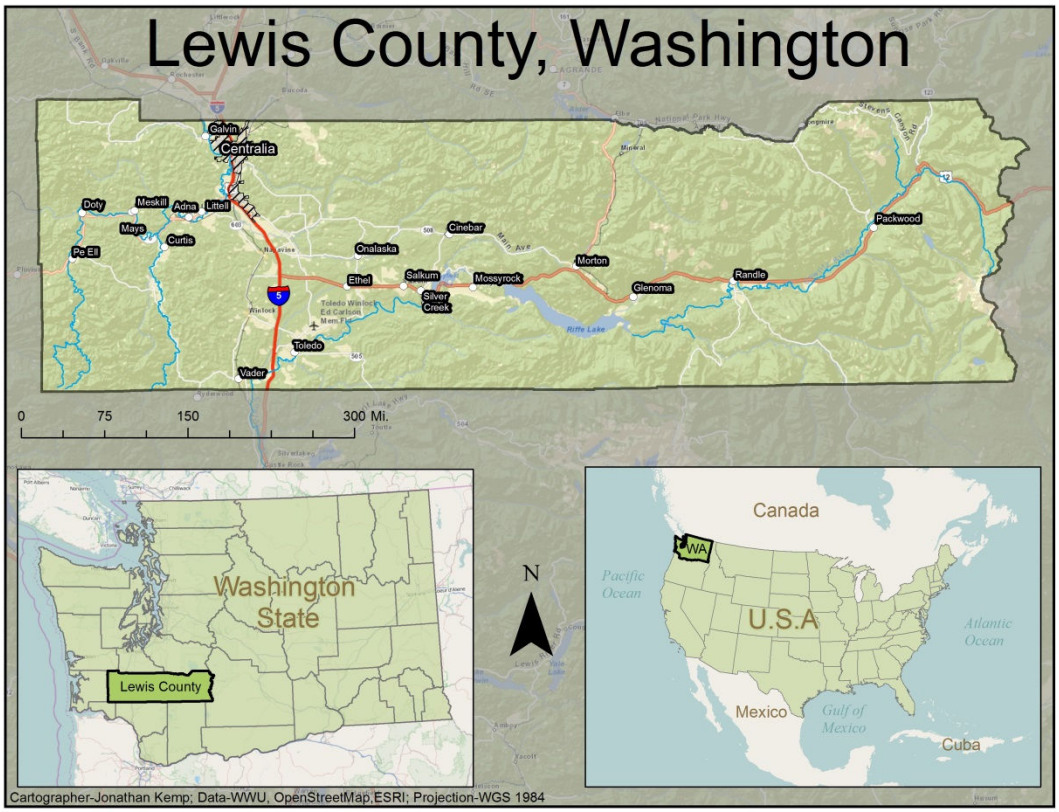

Figure 3 - Lewis County, Washington

Hurricane Katrina started as a tropical depression on August 23rd 2005. The Storm quickly turned into a Category 5 hurricane in the Gulf of Mexico and by Monday August 29th was only 90 miles away from New Orleans. Although the storm slowed down to a Category 3 as it came ashore, there were still predictions of storm surges up to 28 feet (Graumann et. al, 2005). These conditions were well above the severity needed to overwhelm the 125 miles of hurricane reduction system (Army Corps of Engineers, 2005). Although Katrina was not the largest Hurricane to strike the Gulf Coast, the destruction was some of the worst ever seen (Graumann et al, 2005). Figure 4 depicts the path Katrina took leading up to major landfall. While the effects of Katrina were wide spread, the combination of hazard conditions and human vulnerability made New Orleans one of the hardest hit

areas. Within Orleans Parish alone the cost of Hurricane Katrina is estimated around $40-$50 billion dollars (Kates et al. 2006).

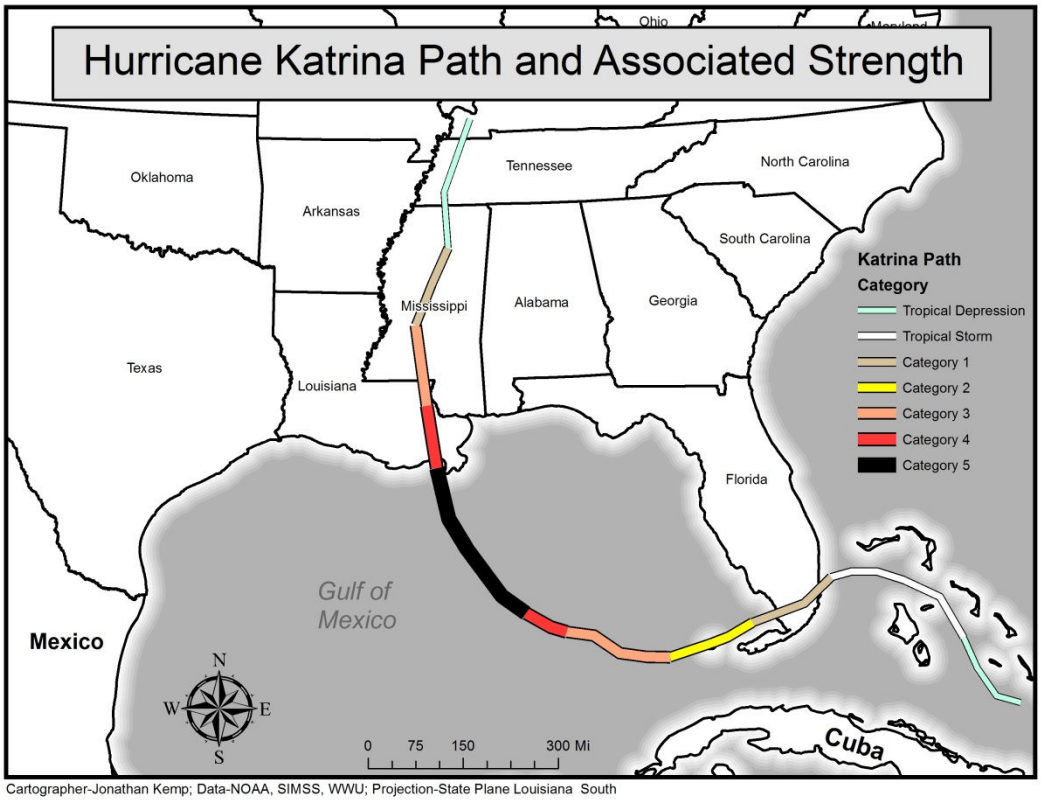

Figure 4 - Hurricane Katrina path and associated strength

New Orleans as a city has an estimated population of 369,250 and is located on the southern tip of Louisiana in the Orleans Parish. The city is below sea level and is sandwiched in between Lake Pontchartrain and the Gulf Coast as shown in Figure 5**.** The land area of New Orleans is 169.42 square miles with a density of 2,030 people per square mile (US Census 2013)**.**

Table 3- Population numbers for New Orleans from the 2010 United States Census (quickfacts.census.gov/qfd/states/22000.html)

People QuickFacts	New Orleans	Louisiana
Population, 2012 estiamate	369250	4602134
Population, 2010 (April 1) estimates base	343829	4533372
Population, percent change, April 1, 2010 to July 1, 2012	7.4	1.5
Population, 2010	343829	4533372
Persons under 5 years, percent, 2010	6.4	6.9
Persons under 18 years, percent, 2010	21.3	24.7
Persons 65 years and over, percent, 2010	10.9	12.3
Female persons, percent, 2010	51.6	51
White alone, percent, 2010	33	62.6
Black or African American alone, percent, 2010	60.2	32
American Indian and Alaska Native alone, percent, 2010	0.3	0.7
Asian alone, percent, 2010	2.9	1.5
Native Hawaiian and Other Pacific Islander alone, percent, 2010	-	0
Two or More Races, percent, 2010	1.7	1.6
Hispanic or Latino, percent, 2010	5.2	42
White alone, not Hispanic or Latino, percent, 2010	30.5	60.3

Table 4- Business numbers for New Orleans from the 2010 United States Census (quickfacts.census.gov/qfd/states/22000.html)

Business QuickFacts	New Orleans	Louisiana
Total number of firms. 2007	27166	375808
Black-owned firms, percent, 2007	28.9	15.9
American Indian- and Alaska Native-owned firms, percent, 2007	0.7	0.7
Asian-owned firms, percent, 2007	5.2	2.8
Native Hawaiian and Other Pacific Islander-owned firms, percent, 2007	-	0
Hispanic-owned firms, percent 2007	4.1	2.9
Women-owned firms, percent, 2007	30.4	27.4
Manufacturers shipments, 2007 ($1000)	3088945	205054723
Merchant wholesaler slaes, 2007 ($1000)	1938441	51415553
Retail sales, 2007 ($1000)	2718026	56543203
Retail sales per capita, 2007	9434	12921
Accommodation and food services sales, 2007 ($1000)	2148176	9729869

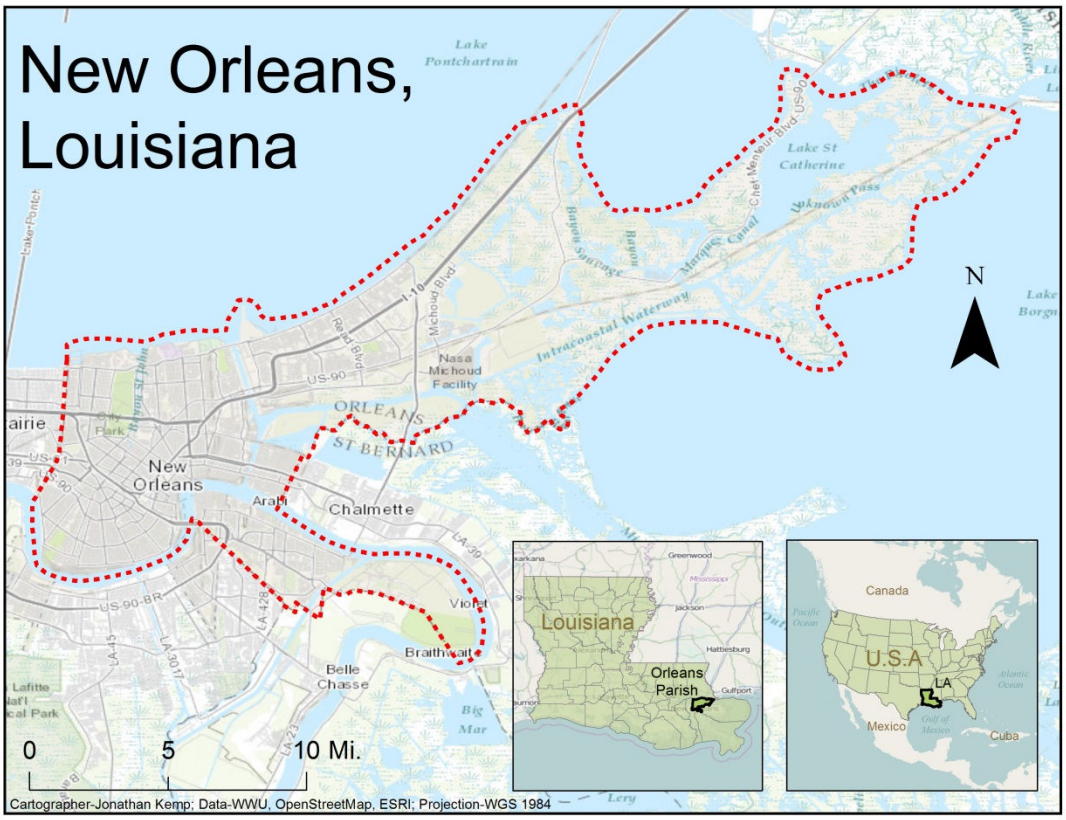

Figure 5 - New Orleans City Proper, Louisiana

3.3 Organization

By aggregating data for each of the two described case studies, certain data use case scenarios became apparent. Specifically, two use case scenarios were common for how the schema could organize resilience data. The first scenario was that an organization holds resilience data and offers access to that data readily. The second scenario was that an organization holds data, but does not offer access to it readily. In scenario one, the resilience data was aggregated and needed to be organized as both an item and the metadata associated with that item. An item could be an excel file, word doc, pdf, csv, picture, etc. In the second scenario, metadata for resilience data was the only thing to aggregate, with no particular item associated with it. The lack of an item was because

of privacy or security reasons previously stated in the literature review (Crittenden 2013; Mills et al. 2008). An effective schema for resilience data should be able to organize both scenarios in a meaningful way. To accomplish organization under both scenarios, the schema design takes on a metadata style structure. Such a structure in both scenarios holds metadata, but also provides the opportunity to organize an item alongside it when available.

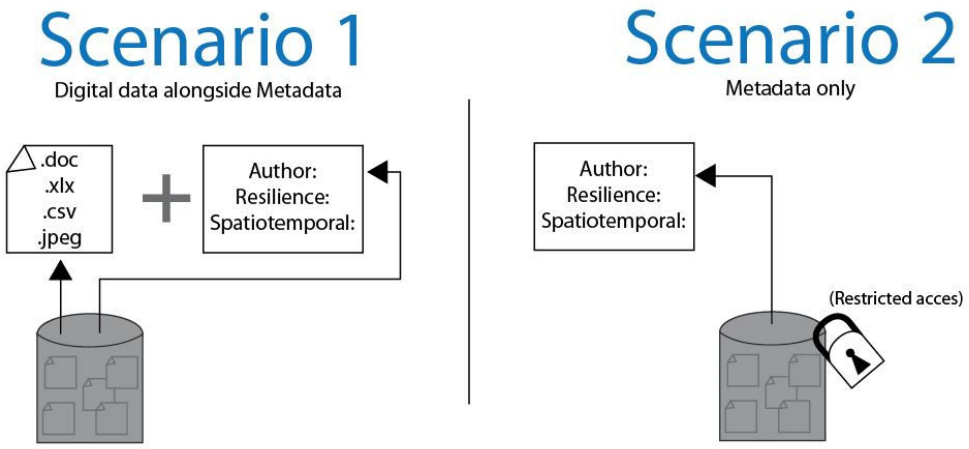

Figure 6 - Possible scenarios for data availability

The schema for resilience data is specified using the attributes of resilience itself in order to store, manage and disseminate those data more effectively. As well as attributes of community resilience, data is also associated with attributes about itself (metadata), including the data source, disaster event, spatiotemporal resolution, and data category. Some of the metadata are attributes used for aggregation described above.

Data source was included because it allows a user of that data to trace it backward. In scenario 1, the source helps verify the legitimacy of the data, as well as makes it easy to discover any recent updates, or related items. The source of resilience data is important when organizing data, but especially so if storing only metadata under scenario 2. If no item was obtained during the aggregation process, than the source metadata is important for managers and researchers to make

requests to the providers of those data to obtain what they need. This is important because even though data may be restricted to some, it may not hold the same usage restrictions for others.

The attributes carried over from the aggregation to organize data are the disaster event, spatiotemporal resolution, and data category. The event describes what disaster the data is relevant to, helping to narrow searches further during dissemination. Organizing the data by spatiotemporal resolution shows what resolution of research and DDDM it can support. For example, one cannot make a decision on statewide policy based off the data from a single town. Organizing by data categories provides a basic description based on what the data is outside of a community resilience framework. For example, data may be considered a built capital within WISC, but it could be described as road closure statistics related to the broader category of transportation. To increase the data utility it needs to be further organized using a robust community resilience framework. Such an organization establishes a link between current technical ways of understanding the data and ways of understanding the data through the lens of community resilience.

To date, there are no databases organizing data in such a way that would remove the work of deciding what data is relevant to a particular event, and how it can inform management and research towards community resilience. Organizing by what attributes of community resilience are associated with the data is very important. In reviewing the community resilience literature, it was decided the WISC framework is the most comprehensive. WISC offers a wide variety of attributes hierarchically structured as objects, constructs and variables. These components of WISC are integrated into the organization schema. Below is an overview of WISC and how it was integrated into the organization. This overview is not meant to be exhaustive, but rather provides understanding of how data is further organized by resilient attributes.

WISC is broken down into two models one, of which is static and the other being dynamic. The static model provides understanding of resilience as a state at a point in time and space, while the dynamic model helps to understand how states change across time and space. Figure 7 is a graphic representation for the organization of the static model of WISC. These objects, variables, and constructs grounded in the community resilience literature provide theoretical understanding to data gathered by broad categories from a variety of literature. This links the various ways of understanding the data into one comprehensive schema.

A piece of data gathered in the aggregation process was per capita income at the county level for the 2007 Lewis County Flood. This piece of data is organized based on the metadata, which are that it is associated with that flood, the spatial resolution is county level, the temporal resolution is by year, its descriptive category is economic, and its descriptive attribute is per capita income. Furthermore, the data is given attributes of WISC, such as that it belongs to the object of infrastructure, the construct of capital, and the variable of economics. Using the definitions of each object, construct, and variable, the framework of WISC can then be referenced to gain understanding of how this data can be used to drive decision making. Notice in this example that the descriptive attribute and variable of WISC are the same. This is not uncommon, but not a requirement of the organizational schema. Transportation, a descriptive attribute mentioned earlier, is not present as an object, construct, or variable in WISC. It is fairly apparent that transportation represents built capital and should therefore have those attributes of resilience in its organization. Furthermore, transportation data could be relevant to both services and capital. Due to this multiplicity, more than one attribute of WISC needs to be represented by the metadata.

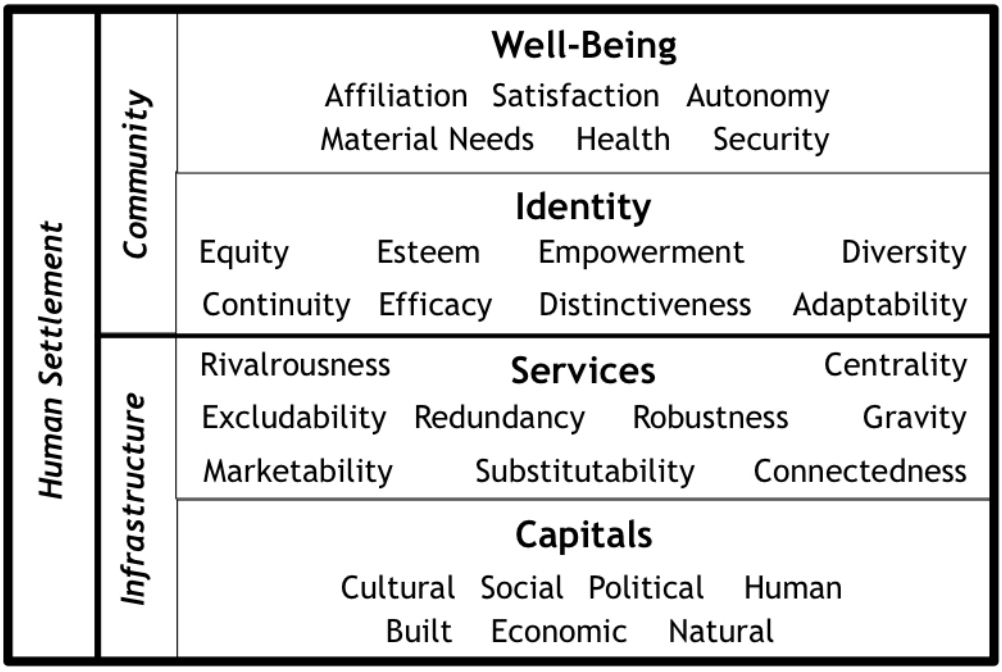

Figure 7: Structure of the WISC proken into Objects (community, infrastructure), constructs (capital, services, identity, well-being), and twenty-nine variables of human settlements (S. B. Miles 2015)

Because this schema is heavily organized around the theoretical understanding of community resilience provided by WISC, the definitions of each component need to be associated with the data in the schema. Doing so by embedding the definitions into each piece of data is far too redundant, especially if multiple WISC components are relevant. Instead, the organizational schema provides a way to organize a single list for the components of WISC, which are linked to each piece of disaster resilience data. This can be thought of as a one-to-many relationship represented in Figure 8. In this example there are three metadata documents for resilience data. One of the metadata documents has an item stored with it, one has no items, and one has multiple items. Each piece of metadata in this example has the same components of WISC associated with it. Rather than embedding these components into each piece of metadata, there is a key linking each metadata document to the WISC document. This way if the component of WISC needs to be modified it can be

done in one location rather than three. It also will take a third of the space on disc. This is done for all 35 components of WISC. . This organizational technique of WISCkey is in anticipation of saving space and time during the storage and management of data.

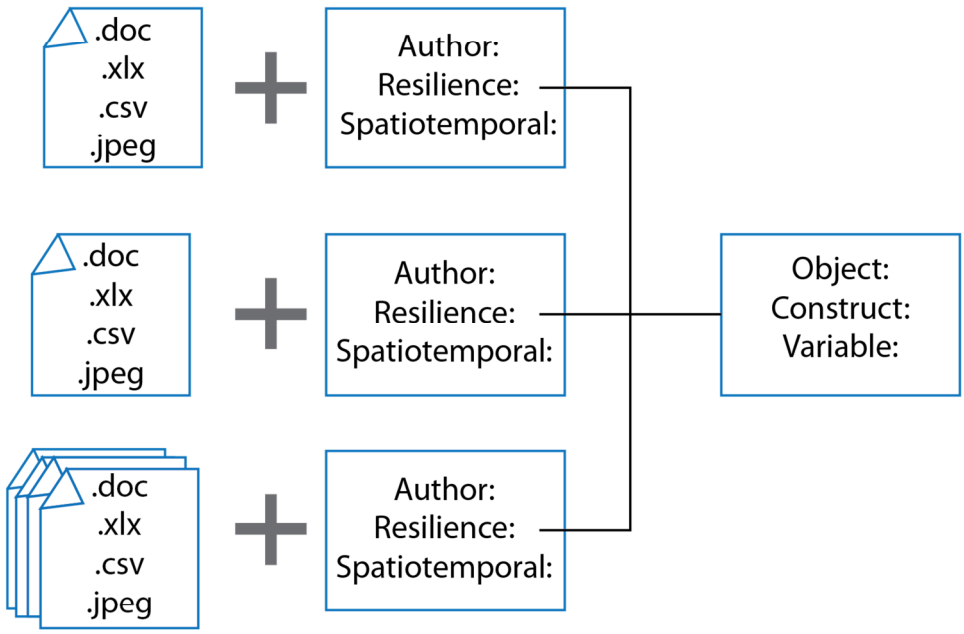

Figure 8 - Possible one-to-many relationship for storing resilience metadata

3.4 Application for Storing, Managing and Disseminating

The third objective was to develop an application that aids in storing, managing and disseminating data gathered in the first objective and organized in the second. While storage, management, and dissemination are all discrete stages in the data lifespan they are all integrated into the third objective of this research. In section 2.6 and 2.7 a synthesis of the literature was performed to evaluate the technical infrastructure that could be used for such an application. The specific technology chosen for storage and management is a non-relational DBMS, specifically

MongoDB. Once storage and management was in place a user friendly way to disseminate those data was needed. Disseminating data was achieved through a Python scripted Bottle web framework. Python was chosen because it is a well-established scripting language, is considered agile, and as a result is successful as a programming tool (Daly 2007). These technologies come together as the web application called WISCkey.

Figure 9 breaks WISCkey into a server-client model of computing. Within these two providers there exist three main components. On the server side there is a Mongo database (#1 in Figure 9) and a Bottle web framework (#2 in Figure 9) and on the client side there is a web browser (#3 in Figure 9). The Mongo database serves the purpose of storing and managing data and the Bottle web framework serves the purpose of aiding in disseminating that data. The following section of the research approach is broken down by these three main components of WISCkey. Within this model there must be a web server such as Apache or NIGNX, but discussing these configurations are out of scope for this research.

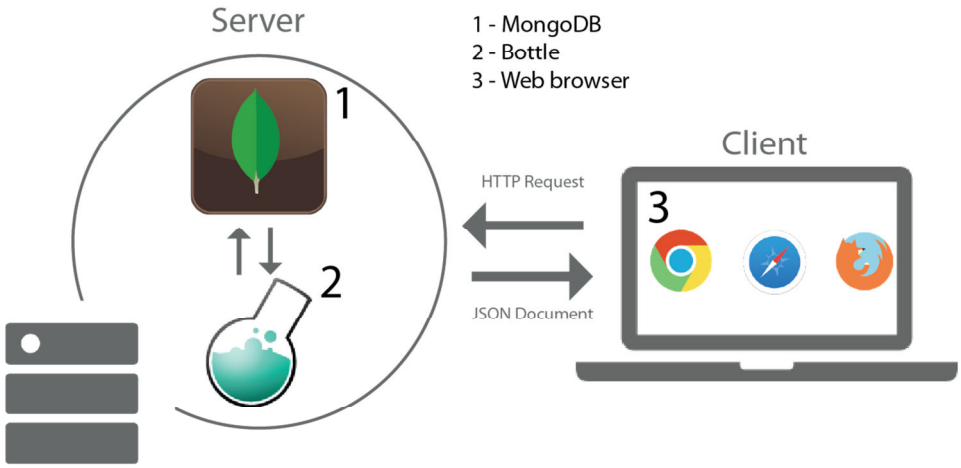

Figure 9 - WISCkey Application Architecture

MongoDB is a scalable and agile DBMS, which are desirable traits for this research. MongoDB was chosen as the DBMS for several reasons. First, MongoDB was readily available and useable in a wide variety of applications as was apparent in the usage examples. Secondly, the feature list is more robust than most other document DBMSs, not completely necessary for the current application, but allows for more advanced functionality to be added as needed in the future. Third, MongoDB is a very popular DBMS, meaning that documentation and help files are very good. Fourth, MongoDB offers a driver for the scripting language Python, whereas CouchDB for example does not.

MongoDB is a document style non-relational DBMS. MongoDB structures data into nested compartments of databases, collections, and documents. Within a MongoDB instance multiple databases are possible. Within each database there can be one or many collections and within each collection there can be one or many documents. A document database is essentially a key-value store, but the way values are stored is unique enough to warrant its own category. The difference is that instead of a single value, a document database will store an HTML, JSON, or BSON document as its value (Fowler and Sadalage 2013). These are three different standards used within programming and stand for Hyper Text Markup Language, Java Script Object Notation, and Binary JSON, respectively. MongoDB itself stores items using JSON and BSON formats. Both are easily read by humans and machines alike which makes it a convenient storage mechanism. Figure 10 shows an example of what a JSON document stored in MongoDB looks like. In reality these JSON documents are stored as one line of text without returns, but it has been formatted for human readability. This particular example is a description of a stress event. Some keys have a single value to them, such as disaster or type. The characteristics key is a collection of two key-value pairs. A key can have a single value as described before, or a list of values such as characteristic and states_with_deaths. This

JSON storage is limitless in its nesting capabilities and this flexibility is one reason a document database was chosen versus a simpler non-relational data model.

```
{
      “disaster” :  “Hurricane Rita”
      “type” :  “Hurricane”
      “characteristics” : [{
            “deaths” : “120”,
            “cost”: “12 billion”,
            “states_with_deaths” : [
                  “Florida”, “Louisiana”, “Mississippi”,”Texas”
                  ]
            }]
      “conclusion” : “While costs were extremely high, death
                        rates were relatively low”
}
```

Figure 10 - Basic example of a JSON document

MongoDB is meant to handle “big data,” which means it was designed to address the four Vs of data, which are volume, velocity, variety, and veracity (IBM 2013). MongoDB handles volume by scalability in the form of sharding collections across multiple machines. As data volume grows, it is easy to add another node into the system to meet the new storage needs. In mongo a single database is designed to spread itself across multiple machines rather than a single server. Rather than buying a single expensive high performance machine, a MongoDB can be spread across multiple low-end machines when high processing power is needed. This means the system remains fast when velocity and volume increases.

The variety of the data itself is handled by the fact that MongoDB has no strict schema (not requiring certain attributes to be populated or keeping new ones to be added without updating all data currently being stored). This schemaless trait makes it so as the variety of the data increases, the documents can adaptively change without having to go back a reformat all previous data.

MongoDB as a DBMS offers an aggregation pipeline and map-reduce function. The aggregation pipeline is specific to MongoDB and refers to its ability to have multiple layers of search parameters. This pipeline ensures that it only scans as much of the document as needed to satisfy the query. Map-reduce is a standard function across DBMS software. Map-reduce filters common queries and provides a summary operation on those queries which can be accessed quickly in the future. The veracity of the data is handled in the schema design by incorporating WISC, the most comprehensive community resilience framework, into its organization.

Using MongoDB as a DBMS allows a non-relational data model to be applied to data aggregated in the first objective. Using the key-value type storage allows components of community resilience, adopted from the WISC framework, to be applied to storage and management. To accomplish this one-to-many relationship described earlier are employed using key-value pairs. Within each piece of resilience data, a key (the name of a component of WISC) provides a link to the value (the definition of that WISC component). The linking of these two documents is handled in the code for the application and will be discussed in detail in section 4.4 of the results chapter. While MongoDB offers a command line interface to manage the storage of data, it is not a convenient or effective way to allow users to access/upload data. For dissemination purposes, WISCkey needed a user-friendly way of accessing data. This user interface is achieved using Bottle as a web framework and represents the second component of WISCkey within the server in figure 9.

Bottle is a basic python framework that is fast, simple, has URL parameter support and is contained in a single file with no dependencies (Piercy 2014). In other words, the application is a single .py file and does not require python libraries to be installed other than the standard library. This keeps the application architecture simple to move to a new server if necessary. These were all desirable traits for disseminating resilience data because it is not only simple, but effective as well.

In order to better disseminate data, WISCkey needed to accomplish several things. First it needed to be searchable so users can pull items and metadata out of the database. Second, it needed to provide an upload function so the database could continue to grow organically. Third, it needed to provide adequate documentation for theoretical understanding of the WISC framework.

Bottle was used to design a Representational state transfer (REST) interface for interacting with the MongoDB database to disseminate the data. A simple way to think of a RESTful interface is one which exposes certain resources. In this use case JSON documents inside MongoDB are exposed with a limited number of operations. The user interface designed for WISCkey uses the standard HTTP (foundation of data exchange via the World Wide Web) requests of GET and POST in order to communicate information. GET is used to request information, and POST is used to communicate that information. For example, GET is used to ask a user for a search parameter and POST is used to send that query to the MongoDB. GET is then used to retrieve the result of that query and POST used again to deliver it to the user. These communications occur between a client side (user's computer) web browser, the third component of WISCkey's architecture, and the MongoDB on the server.

Chapter 4: Results

4.1 Results Overview

The technical details of how the three objectives were achieved constitute the results of this thesis. The results for objective one are a set of resilience data, aggregated based on the capitals of the theoretical framework WISC, for two disaster case studies, at different spatiotemporal scales. An organization of data based on WISC is applied for the storage and management of those data. This organizational schema is the result for the second objective. This schema helps to organize data

based on elements of community resilience, as well as more conventional metadata. The result of the third objective is a web-based application called WISCkey. WISCkey was built using MongoDB, Python, and Bottle. It was developed to store, mange, disseminate, and provide the means to centralize a variety of resilience data. Ultimately these completed objectives will permit applying community resilience theory to facilitate data-driven decision-making, and research, in a user friendly way.

4.2 Aggregated

The first objective of aggregating decentralized resilience data at multiple spatial and temporal scales yielded a sample dataset for Hurricane Katrina and the 2007 Lewis County floods. Using the research framework, an aggregation protocol was used to gather 52 datasets from 22 sources of decentralized resilience data. Of these 52 datasets, 29 were for Hurricane Katrina and 23 were for Lewis County. Of the 52 datasets, 26 were for Hurricane Katrina and 23 for Lewis county floods contained items alongside them. Of the 52 datasets, entries had 0 cultural, 5 social, 1 political, 6 human, 8 built, 33 economic, and 0 natural tags of relevant elements of resilience. These add up to more than the total of 52 because each dataset can have multiple attributes of resilience. While expansive, these data are not comprehensive. Rather, they serve as an example of what spatiotemporal resolutions of data are available for large and small scale events. These data also help test the feasibility of using WISC in the next stage of organization.

The similar number of data for each dataset between the two case studies indicates access does not change based on spatiotemporal scale of event. However, these values do not mean that the amount of data relevant to each event was the same. Given the limited scope of this thesis, case studies were chosen using the spatial and temporal hierarchies designed for aggregation. If one were to repeat the same aggregation for a town in Oregon looking for effects from the Lewis County

flood data, very few datasets would show a recovery signal to an event of this scale. Because hurricane Katrina affected a large geographic area, data for a town in Mississippi would still show a recovery signal associated with hurricane Katrina.

Given the idea of Katrina affecting multiple spatial hierarchies, it can be concluded that events that span a large geographic region generate relevant data more so than intensity of the event. As the amount of data increases the variability of access will fluctuate. For example, if an event is spatially small it might affect a single city whose public data access is poor. If an event is spatially large it might affect multiple towns whose public data access is poor, but it might include one or two with high quality data access.

To ensure veracity of research and DDDM, a useful resolution of data is needed both temporally and spatially. A conclusion to any problem will be insufficient if it is not timely or spatially relevant to the community for which it was made. The intensity and a real extent of the disruption event will affect what temporal and spatial scales are relevant to understanding recovery. For example, if the intensity of the disruption event is such that a community fully recovers within 6 months, annual level data is not going to necessarily show a signal for the event and subsequent recovery. The same applies for geographic extent, if a single small town experiences a disturbance event, the signal showing a decline in services and subsequent recovery may not show up in state level data. This intensity and geographic extent are what define the overall scale of an event in the context of this research.

Scale of a disturbance event, in this context, is based largely on the intensity of the event, while the geographic extent becomes a multiplier adding to that scale. The smaller in scale the disturbance event, the more granular the data must be to see a disturbance signal, the larger the scale the more coarse it can be while still seeing a disturbance signal. For the most part, the

temporal resolution needed for recovery research and DDDM is primarily determined by the intensity of the stress event, while spatial resolution needed for research is determined by the geographic extent the stress event directly affected. The exception to this rule is if intensity is great enough to completely overwhelm the larger system. For example, 9/11 was a very intense event in very centralized locations, so much so that it had effects across the nation and even the world for many years. The intensity of this event was such that it overwhelmed the resiliency in place, which created a large scale disturbance event, even though it was small geographically. Another complicating factor in this exception is that the systems that the event effected, while localized geographically, were global in a political and economic nature.

4.3 Organized

With sample datasets aggregated, a schema was created to organize data with respect to components of resilience. The organizational model is referred to as a schema and will act as the data structure later when it is deployed within the application WISCkey. This was completed first at a higher conceptual level of organizing based on theoretical ideas of community resilience, source metadata, and descriptive attributes. This high level schema was largely driven by the research approach. Later, the schema was refined at a technical level to help operationalize storage and management of data.

There are two main portions of the schema as shown in Figure 11, a minimal portion and an optional portion. Each of these portions contains several fields which organize user-inputted metadata. The minimally required metadata fields are represented by red text in Figure 11 and are as follows: title, event type, event alias, summary, elements of WISC, descriptive category, spatiotemporal resolution, and source agent of the data. The schema will not allow an upload if this minimally viable metadata is not present. Optional metadata fields, ones which are not necessarily

needed but found to be useful if they exist, are blue in Figure 11 and are as follows: specific author, contact information of author, URL where data was found, publisher, primary source of the data if relevant, and the date of the data. These 22 metadata fields are the high level conceptual schema for organizing resilience data.

The second result for creating a schema was the technical details that operationalize the high level concepts. These details are the resulting decisions of where the schema is open to an infinite possibility of inputs and where inputs are restricted. The ultimate result of this decision are 6 fields being restricted, 3 being machine generated, and 13 being open. The black values in figure 11 are representations of the metadata input allowed for a given field. In the cases where inputs are restricted, the options for inputs are found in the boxes in the right side of the figure. These designations were chosen based on patterns across all datasets aggregated, methods used during the research approach, and designations established by other agents.

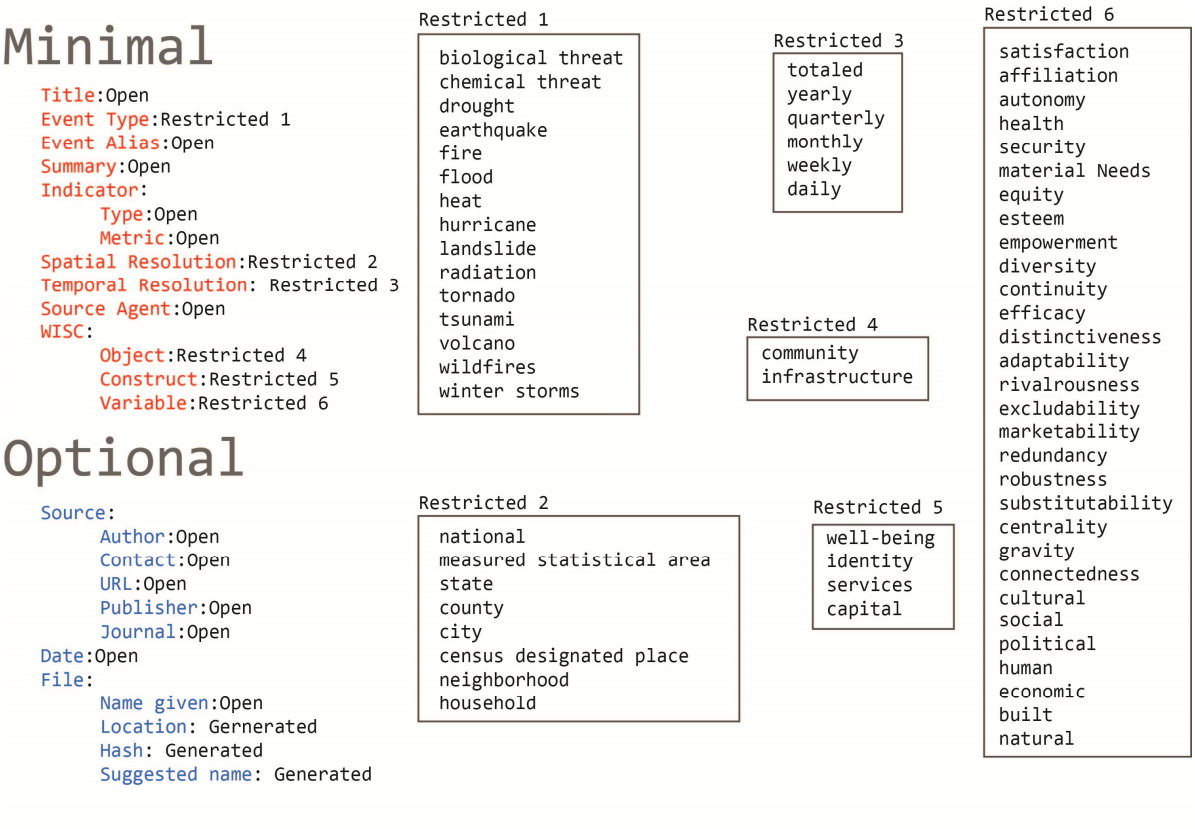

Figure 11 - Organizational Schema for Disaster Resilience data. Fields that are required are in red and fields that are optional are in blue. In some cases field inputs are restricted to a list of values which are on the right side of the figure.

Organization needed to be accomplished for two data scenarios. Recall that scenario one is when only metadata exists, and that scenario two is when metadata exists with an actual item of data as well. These dual scenarios drove the overall structure of the schema to take a metadata style design. The reason for organizing anything is to be able to easily find it when it is needed. To accomplish this basic goal of organization the where, when, who, and what of the data is important. Under both scenarios, there is a minimal amount of metadata needed in order to organize it. Once these minimal fields of metadata are satisfied there are optional fields that can, and should be, satisfied. Below is a discussion of the metadata fields used to organize disaster resilience data, and

how each group of fields relates to the larger concept of community resilience. Together these metadata fields constitute the organizational schema and provide answers to where, when, who, and what.

Each of the 22 fields of metadata falls within three major organizational elements; metadata on community resilience, metadata on the source, and metadata on descriptive attributes. The metadata for the descriptive attributes of the data provides a foundation to organize data based on its spatiotemporal resolution, location, and description. This allows data to be organized on a high level of how the data is currently labeled, what resolution it was captured at, and when it was captured. Descriptive attributes also make the schema searchable with a minimal working knowledge of WISC, or community resilience in general. By enabling users unfamiliar with resilience and WISC to easily find data it will allow for the progression of their understanding from disaster recovery toward disaster resilience.

The metadata for community resilience creates a key for exploring how the data is related to the theoretical framework of WISC. By associating data with elements of WISC, the schema organizes what the data represents in terms of resilience. For example, if data is tagged in the metadata as an economic capital and a rivalrous service, it can be linked to the relevant metadata documents of WISC and therefore community resilience.

Having metadata associated with WISC allows organization and searching to occur using an advanced knowledge of what resilience means. Having metadata for the source of the data allows a user to quickly check the legitimacy of the data, where to find related data, and who to contact with additional questions. Source metadata allows for further organization of data based on who captured, stores, manages, and disseminates those data.

By restricting some fields and leaving some fields open, this schema is pre-defined enough to organize data, but open enough to allow the user to capture any uniqueness those data provide. In order to organize data in meaningful ways, commonalities need to exist between the storage categories. These commonalities were partially identified and defined during the aggregation process, and partially already defined by frameworks such as WISC, or definitions recognized by FEMA. These commonalities are restricted fields controlled in the technical part of the schema and are as follows:

1. Event type
2. Spatiotemporal resolution
3. Elements of resilience.

Event type was limited based on FEMA recognized disaster event types. The spatiotemporal resolutions were limited based on the hierarchies recognized and developed during aggregation. Elements of resilience were limited based on the elements from the theoretical resilience framework WISC. These restricted inputs create predictable values which are recognized and searchable by WISCkey. These restricted inputs can easily be changed, modified, or created as the data evolves. The restricted fields falling under the three organizational elements in the schema provide an outline defining the what, when, where, and who of resilience data. By adding these pre-defined fields to the schema, structure is added to the organization that allows for storage and management through predictable values.

4.4 Storage and Management

The technical way the schema became operationalized affected the way in which data was stored and managed. The approach to storage and management is discussed briefly at a high level in section 3.4, but the details of how this occurs are the result of that approach. The schema

developed for Objective 2 required a minimal upload of metadata, and requested optional metadata when available. Once uploaded using WISCkey, the metadata for resilience data will have fields dictated by the schema and will be stored and managed within the DBMS MongoDB. How and why this occurs is addressed here.

Within MongoDB each upload is stored as a single JSON document, with an optional related item accompanying it stored outside of MongoDB. This item that accompanies the document is stored in its native format (such as jpg, png, xlsx, etc.) within the same directory on the server which contains the entire application of WISCkey. Within MongoDB there exist two collections for the storage and management of resilience data. One collection holds all user metadata and is titled "data". Figure 12 is an example of a metadata document as it exists within the data collection. Notice that within Figure 12 the field keys have been highlighted with their corresponding minimal and optional metadata categories. There are 3 fields not highlighted in Figure 12. These consist of an _id field created by MongoDB as a unique identifier for the JSON document, and two filename paths, which are created in the WISCkey code. The directory filename is the one which is used to store the item on the server in its native format.

```
{
        "_id" : "ObjectID("532ac942e138230e44fb7158"),
        "event" : "Hurricane Katrina",
        "indicator" : {
                "metric" : "Road closure",
                "type": "Transportation"
        },
        "title" : "Yearly road closure data Orleans Parish",
        "notes":  "Good yearly data from 2005 to 2014, it is missing 2011 though for some reason"
        "WISC": {
                "variable" : "built",
                "construct": "capital",
                "object" : "infrastructure"
        },
        "hazard" : "hurricane",
        "fileNames" : {
                "directory" : "/etc/Prototype/files/af436c8526e1596c1df45938990c2c72.jpg",
                "new" : "county_yearley_DOTD_Road-closure.jpg",
                "original" : "All_Mo_ASCE_RoadCLosure05.jpg"
        },
        "source" : {
                "date" : "2008-09-01"
                "contact" : "John Doe, 360-111-111",
                "author" :  "DOTD"
        },
        "spatial" : "county",
        "summary" : "This road closure data acquired from the DOT for the Orleans Parish",
        "temporal": "yearly"
}
```


Figure 12 - Disaster resilience metadata document as it exists within MongoDB. Per the conventions in figure 11, required fields are highlighted in red and optional fields in blue.

The second collection, "wisc", holds a JSON document for each element of WISC. An example of an object, variable, and construct document stored in this wisc collection can be found in Figure 13.This wisc collection is then linked to the data collection to connect the metadata of wisc, which are concepts of resilience, to the metadata of disaster recovery data. This relationship was shown in Figure 8. This is done on the server side and is handled through the HTML application code in WISCkey. When a user submits a query results are displayed in HTML, each result has an HTML hyperlink associated with it. When this hyperlink is clicked, it sends another query to MongoDB, via Bottle, and retrieves the corresponding WISC document and displays it as a popup for the user.

While possible to keep all documents in one collection within the MongoDB database, it was chosen to split the resilience metadata and WISC descriptive metadata into separate collections. This was done to create a layer of separation between the two to clarify that one was derived from

aggregation, and one was derived from the WISC framework. This separation also allows for another framework to be added in a separate collection to expand ways of understanding the data. Figure 13 is an example of the three layers of documents that exists for elements of WISC. The important connection between the data collection and the WISC collection is the WISC field inside of data. The value of this field acts as a tag that can be used to link the documents from the two separate collections. Notice that in Figure 12 the data document has a built capital value for the key of WISC. Using the value of this field in the data document one can look up the definition of the relevant variable, construct, and/or object of WISC.

```
{"object":"Infrastructure",
       "construct":"Capitals",
       "variable":"Social",
       "desc":"Social capital refers to the bonds that hold people and groups together
       as well as the traditions and languages within a community, and acts as a cohesive force."
},
{"object":"Infrastructure",
       "construct":"Services",
       "desc":"A service is the benefits people derive from capital, where the end or goal is sustainable
       human well-being. Services are how individuals and communities access and utilize capital towards
       promotion of their well-being."
},
{"object":"Community",
       "desc": "A community is any collection of agents that refer to themselves as members or us based
       on perceptions and variableions of identity. Agents are defined as any individual person or group
       of persons who have the ability, if not opportunity, to make decisions and exert power towards
       their and others well being"
}
```

Figure 13 - Metadata documents for 3 objects of WISC within MongoDB

Separate from these two collections are static items stored locally on the server alongside MongoDB. MongoDB offers a feature called GridFS which would allow for non JSON documents to be stored inside the database, but this added an unnecessary level of application code to GET and POST items uploaded by a user. The simplest solution is often the most elegant. For this reason, the simpler method of keeping the files in their native format, and store them alongside MongoDB was chosen. The metadata document is simply tagged with the item's directory location on the server.

This directory location information can be used to POST the item when a user requests it. Upon upload, the item is renamed for storage according to a unique hash to ensure each item is unique and not conflicting on the server. In order to manage these items once they are uploaded, the legacy name upon upload is saved to the metadata. Additionally, a new descriptive name is generated based on the field values from the metadata name, author, and spatiotemporal scale.

To this point, the most significant management function performed has been querying the database to pull data out of MongoDB and disseminating it through WISCkey. To accomplish this, a query was designed using MongoDB search syntax. Figure 14 is an example of a query performed within MongoDB, and uses inputs from WISCkey, which are highlighted in yellow. The first line is specifying to search inside the collection data. The second line (first line in the brackets) specifies to search for every document that has an event field with some piece ($regex) of the same value of event (as specified by the user), and apply the option i (non-case sensitive). This is done in turn for each field represented in Figure 14 by a new line of code. Since the user is only entering one search perimeter for the resilience component of WISC, the $or modifier will modify the command to first look at the WISC objects. If there is no WISC object that is equal to the input, then it will then look for constructs, and lastly variables.

```
db.data.find (
{
          'event':{'$regex':event, '$options':'i'},
          'indicator.type':{'$regex':indicator, '$options':'i'},
          '$or':[{'WISC.object':{'$regex':wisc, '$options':'i'}},
               {'WISC.construct':{'$regex':wisc, '$options':'i'}},
               {'WISC.variable':{'$regex':wisc, '$options':'i'}}],
          'hazard':{'$regex':hazard, '$options':'i'}
     }
)
```

Figure 14 - Example of the management code used to query the organization schema. The items highlighted in yellow are user inputs and change depending on the search criteria

The other significant management function that has been used so far is a remove command to delete test data earlier in the development process. MongoDB offers a full suite of commands for finding, and modifying documents as well. An upload can occur exclusively inside MongoDB but is more difficult, time consuming, and prone to error. Rather than using the mongo command line, the graphic user interface of WISCkey can be used to upload data.

The choice of the DBMS MongoDB, and the sequential storage and management based on its capabilities, was a large decision. The overall choice of MongoDB was derived by the need to have an extremely agile DBMS that could adapt as quickly as the schema changed, which was still in development at the time of DBMS selection.

4.5 Dissemination (WISCkey)

The results of this particular section are application code. This application code is at the heart of WISCkey and connects the HTML coded pages to search, gather, and disseminate data within the database. This application code can be found in appendix A. The HTML code associated with searching can be found in Appendix B, uploads can be found in Appendix C, and query results in Appendix D. Additional help pages are also available from within WISCkey which outline how to use

the application(Appendix E) and information on WISC itself (Appendix F). Additional CSS style sheets accompany some of the HTML pages and can be found in the appendices denoted with –b. The code itself represents the technical details of how these results were achieved and are components that make up WISCkey. The most important part which ties everything together is the application code in Appendix A.

The top of the application code starts by loading in the necessary python libraries such as pymongo, bottle, urllib2, hashlib. Next, global MongoDB variables are established using the MongoClient functionality and a connection is set based on which database to interact with. At the end of the API, is a section that establishes the host machine and its IP address, associated port numbers, and additional parameters. An example would be run(host='111.111.111.111', port=1010, reloader=True)**.** In this example, reloader=true indicates that the application should be reloaded every time there is a save made to the code. In between the library imports and the ending run function are a series of bottle containers. These containers run associated sections of python code on the server when a designated URL is visited. An example of a bottle container might be @route('/upload', method='POST'), which will run if the host webpage is visited. In the working example the host page would be 111.111.111.111:1010/upload, and would run any python code in that container. Essentially WISCkey is made up of two main containers: an upload and a search container.

Uploading starts on the client end. A user visits the upload page, seen in Figure 15, which is comprised of an HTML web form, CSS, and JavaScript in Appendices C and C-b. The HTML form is a series of mostly text inputs with dropdowns for the components of WISC, temporal and spatial resolution, and event type, all which are specified by the schema. Each input has an associated HTML name, which uses dot notation to specify a category and a subcategory. For example, there

are several inputs associated with the source category. In this example the dot notation for source category metadata would be source.author, source.url, source.contact, etc. When a user fills in the fields and clicks the submit button, the application performs the action of POST to a URL pointing to a specified bottle container holding a python script.

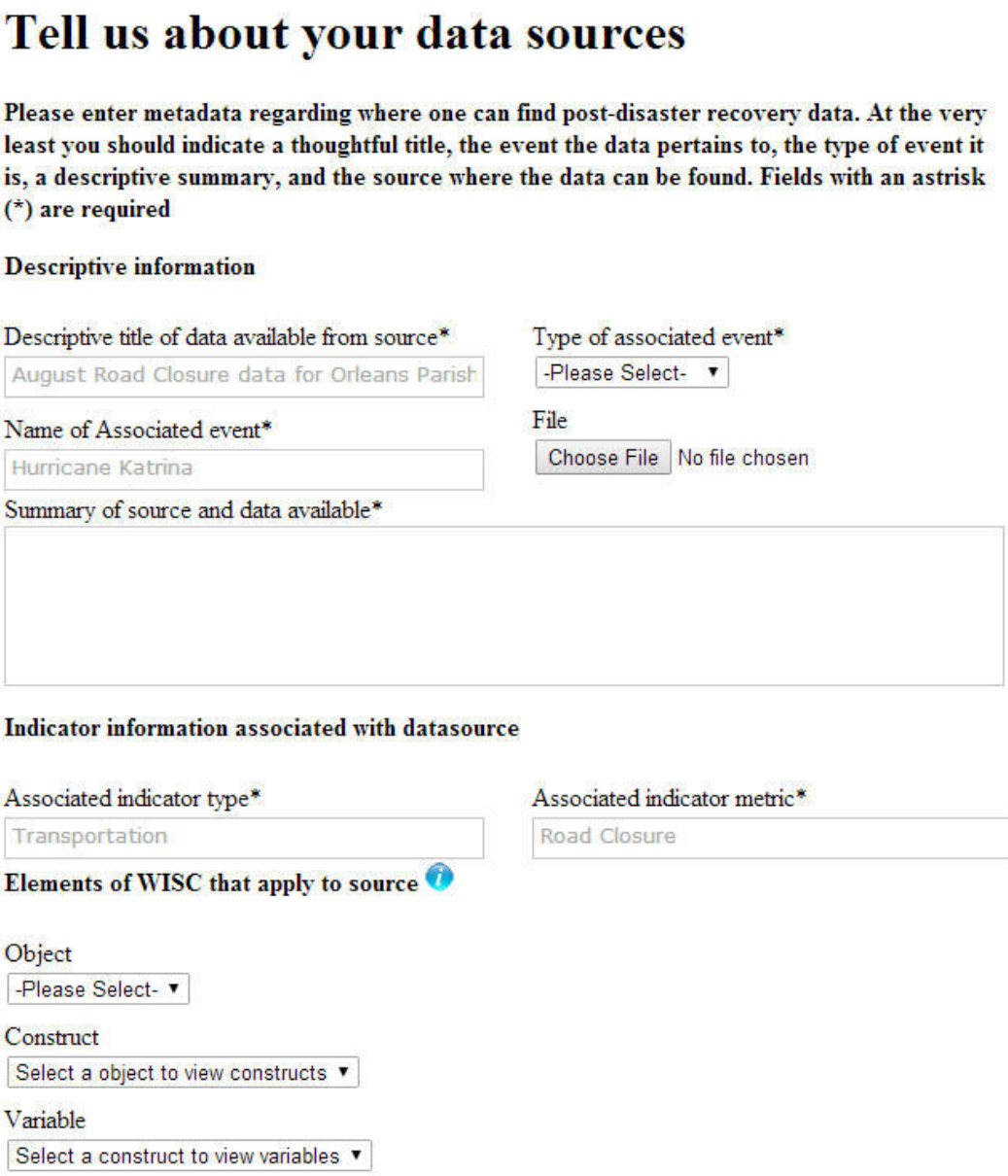

Figure 15 - Screen capture from the data upload form page

Once received by the bottle application, the python function that runs inside the Bottle container does five main things. The function pulls the upload form input, renames any attached files, saves it to the server, parses the form input, creates a JSON document, and finally uploads that JSON document to the data collection in MongoDB. The code uses the form input, which is in dot notation, and splits the contents up. Anything in the position to the left of the dot becomes a

category; anything to the right becomes a subcategory. In the source example mentioned above, source becomes the category and within that there are the subcategories of author, url, and contact. Each of these subcategories will then store the values that were uploaded via HTML form.

Using the upload function, WISCkey converts the form data to a JSON document that fits the schema, and allows for the insertion directly into MongoDB. Any associated file attached to the metadata has a unique hash key generated; the file is then renamed according to that hash and saved to the server. When this is done the data's original name, file path, recommended name based on spatial, temporal, source, descriptive information, and unique hash are all saved to the JSON, and uploaded to the mongo server. This will later allow access to that data when one queries its associated metadata through WISCkey.

The search functionality was implemented to make querying the database as simple as possible for the user. In order to do this a search page was created to serve as the landing page of WISCkey seen in Figure 16. From the landing page a user has the option to filter their search based on event name, event type, element of WISC, or what file format they want. The user can fill in all boxes to combine search criteria or search by a single field if a broader search is desired. Upon submitting the search, WISCkey uses a combination of jQuery, Ajax, JavaScript, and HTML to complete the query by accessing MongoDB.

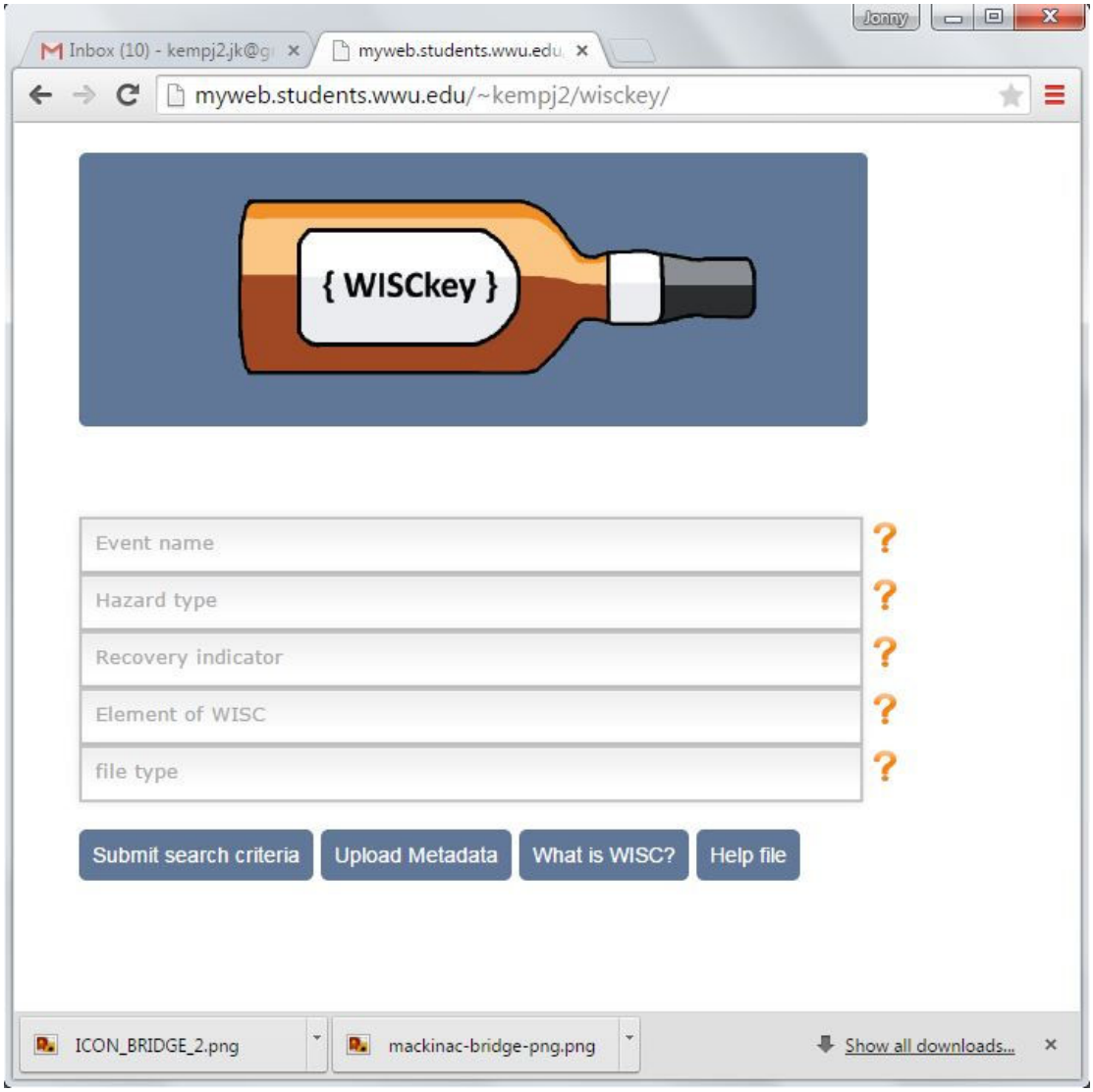

Figure 16 - Screen caputre of the home/search page

When the search button is hit, WISCkey sends the search parameters, which are stored in the URL (via Ajax), to an Ajax function located on a results HTML page (Appendix D). This Ajax function asynchronously runs the search container in the bottle framework (Appendix A). The search container within the bottle framework runs a regular expression query on the data collection within MongoDB (Figure 14). A regular expression query eliminates text case sensitivity, as well as returns partial matches. Once the query is run, the results of that query are returned as a JSON using the

JSON dumps command. Again, Ajax allows that search function within the bottle framework to run asynchronously on the results HTML page, while also receiving the JSON return from the search container.

To present the JSON document in a useful manner, the parse results function iterates through the returned JSON document. For this standard three-statement JavaScript loop was used to display contents in a structured, and formatted, way using HTML. Tertiary statements were used to return the optional fields, because they may not exist in the MongoDB. Within the HTML result are the hyperlinks discussed in section 4.4 that connect to the WISC collection. When active, these bottle containers run a separate query to find the desired WISC element, returning it via HTML similar to the data result as seen in Figure 17. These resulting documents are an exploration of effective methods of disseminating resilience data. Next, additional conclusions that can be drawn from these methods are explored.

non-farm employment
Hazard Type: hurricane
Associated event: Hurricane Katrina
Source: Bureau of Labor Statistics
Summary: quarterly data from the Bureau of Labor Statistics QCEW database. There was no NAIC code for non-farm so I downloaded an excel for total industry , and one for agriculture (NAIC 11) and will subtract the farming from the total. I downloaded the data specifically for Orleans Parish, Lewis County, Louisiana, and Washington. Timeframe is from 2002 until 2012
Recovery Indicator: Economic
Metric within indicator: employment
Spatial Resolution: county
Temporal Resolution: quarterly
Author: Bureau of Labor Statistics
URL: http://data.bls.gov/pdq/querytool.jsp?survey=en
WISC Objects:
infrastructure
community
WISC Constructs:
Capital
Well-being
WISC Variables:
Economic
Affiliation
Sample: county_quarterly_Bureau-of-Labor-Statistics-_employment.xls

Figure 17 - Sample return on the data results page

Chapter 5: Conclusion

5.1 Overview

The results of this research show that by completing the three research objectives the beginning stages of centralizing data about community disaster resilience have been set in place. The first research objective was to aggregate resilience data as it currently exists in a decentralized fashion. This aggregation was completed at multiple spatial and temporal scales for two case study events. The case study events chosen were Hurricane Katrina and the 2007 Lewis County floods. The reason these events were chosen was because they offered two very different spatiotemporal

scales. In both cases the aggregation was performed in a structured way using the developed spatiotemporal hierarchies.

After a sample set of data was aggregated for the two case studies, an organizational schema was created with respect to components of resilience. For the schema, components of resilience were drawn from a comprehensive community resilience framework known as WISC. The schema uses the pre-defined components of community resilience, the spatiotemporal hierarchies developed during the aggregation protocol, source information, and descriptive characteristics to organize resilience data items and metadata. With the ability to include items of data plus the associated metadata, or just the metadata, this organizational schema provides a way to organize useful datasets themselves, or the knowledge to track them down when needed.

By applying the organizational schema to a stack of data management software, an application was built to store, manage, and disseminate those data. The application was called WISCkey because it is the key to exploring data based upon its resilient components as defined by the framework WISC. The application was built on top of a CentOS operating system using python, MongoDB, and bottle. WISCkey facilitates a way to explore geographic data as it relates to community resilience and the sources that created them.

So then how can decentralized geographic data be leveraged to facilitate data-driven decision-making about community disaster resilience? Data can be represented by linking decentralized sources of data via metadata. These metadata, being grounded with a theoretically-based schema, help disaster professionals discover general data. Once discovered, these disaster professionals can then link and interpret the varieties of disaster and non-disaster specific data through the understanding of community resilience. For this research an appropriate stack of technologies was chosen to manage these datasets. These technologies include MongoDB and

Bottle. Application code including python, html, css, and Java Script was then used to communicate between client and server. All together this work operationalizes the theoretical knowledge of community resilience using an effective stack of technology to facilitate DDDM and research. It does this by cataloging the decentralized sources as they exist now, and offers additional storage mechanisms to begin centralizing the data.

5.2 Conclusion

This research provides insight into how to aggregate, organize, store, manage and disseminate data and metadata for understanding community disaster resilience. This insight provides researchers and decision makers with a theoretically-grounded system for facilitating data-driven decision-making in management and research in regards to community resilience. Such a system acts as a roadmap for gathering data, but can also act as a repository. A centralized resilience database prototype called WISCkey was developed and loaded with sample data. WISCkey operationalizes the theoretical community resilience framework WISC using a simple design and effective stack of data management technology. WISCkey serves as an exemplar for how to better organize, store, manage, and disseminate community resilience data. Furthermore, this system is highly generalizable because it can easily be restructured with new frameworks. These frameworks could pertain to any type of decentralized data. This means that the guiding principles behind WISCkey could be used to build applications housing geographic data to facilitate understanding of other disciplines.

WISCkey is the only application specifically designed for aiding in the research of community resilience. It is also the only application organizing data based on the disaster event it is associated with. By leveraging this organizational schema, WISCkey ties in concepts of community resilience directly to datasets relevant to specific disasters. These linked concepts can be refined even more by

providing the spatiotemporal scales the data is relevant towards. It was determined during the aggregation protocol that the spatial resolutions the data can be organized by are national, state, metropolitan statistical area, county, city, neighborhood, census designated area, and individual. The temporal resolution that data can organized by is totaled, yearly, quarterly, monthly, weekly, daily, hourly, or more than once per hour.

Ideally, as new data becomes available, disaster researchers and managers will load the items and metadata into WISCkey using the concepts described above. This collaborative environment acts as a centralized location to kick off any number of disaster research projects. As it grows and becomes more sophisticated by future work, WISCkey could easily become the equivalent of a google search engine, but specifically for disaster resilience data. Furthermore, it reaches beyond a simple search engine by facilitating understanding of the datasets through the WISC community resilience framework.

By facilitating the understanding of datasets in this way, this research provides a way of representing data in such a way that it is relevant toward community resilience research. It does so by using metadata to embed knowledge of a topic into the data and its source, based on a theoretical framework. In this research WISC (S. B. Miles 2015) was used to embed knowledge of community resilience, but this could be extruded to other disciplines and their theoretical frameworks as well. WISCkey exposes a framework to a wide audience, and allows them to operationalize it. By operationalizing them, new insight can be gained by examining how different people, with different biases, choose to embed knowledge into their data.

While highly technical in nature, this research is grounded in the discipline of Geography. It carefully ties the information of space, time, and event to data. It also borrows heavily on the Geographic Approach for understanding how data moves from creation to decision support in the

data lifespan. Looking at how data was aggregated outside of a WISCkey type application, it is clear that spatiotemporal elements have an effect on readily available data. It was concluded that the temporal resolution needed for recovery research and DDDM is primarily determined by the intensity of the stress event, while spatial resolution needed for research is determined by the geographic extent the stress event directly affected. Again, the exception to this rule is if intensity is enough to completely overwhelm the system.

As providers like Amazon Web Services push projects such as their Open Data Project forward, more and more resources are going to be stored efficiently in the cloud and will be freely available. The data model Amazon suggests with this project is to provide one location that will serve the most up to date datasets. However, this does not solve the problem of navigating the datasets for what is relevant toward research. There will inevitably be multiple agents providing such services, and none are going to be specific to resilience data. WISCkey and the schema that organizes it is the compass to navigate this composite of data. By providing metadata and examples, WISCkey takes the theoretical understanding of community resilience and operationalizes it by providing insightful methods of aggregating, organizing, storing, managing, and disseminating community resilience data.

5.3 Limitations and Future Research

The application WISCkey developed and supported through the research performed in this thesis serves as a proof of concept, but there is an extensive list of features to be added and things to improve upon related to style, functionality, and data collection.

One of the many strengths of WISCkey is that anyone can contribute data and metadata to grow the knowledge base. Unfortunately this open contribution is a limitation of WISCkey as well. Because anyone can contribute, there is no guarantee that the entry will be appropriate or relevant.

The required fields upon data entry help to ensure all relevant data is included, but can only go so far and will certainly not protect against malicious entries. This means that WISCkey will need to be monitored to assure data quality is maintained.

With respect to style, cascading style sheets (CSS) were used to present Hyper Text Markup Language (HTML) more attractively in the browser. For example the data upload form uses CSS to outline input boxes once the mouse hovers over them. Multiple styles could be developed and tested with consumers of WISCkey to find which design is most appealing and functional to the user. At the same time usability surveys could be done that would be used to gauge the actual effectiveness of WISCkey. When WISCkey was being designed the HTML and CSS used was under the working assumption that the application would mainly be used via desktops. This means the design is non-responsive to mobile devices. This acts as a limitation and should be addressed in future work

More functionality could be added to WISCkey so it could be integrated into automated analytics and data visualization. Within MongoDB, indexes could be established on datasets. A normal query that is sent to MongoDB has to scan every document in a collection to select those documents that match the query statement. These types of scans are not particularly efficient because they require the database to process a large volume of data. Indexes are special data structures that store a small portion of a collection that is easy to transverse. By predefining queries that will be performed often, the MongoDB can use an index to limit the number of documents it must inspect. Establishing these indexes in future work will improve the functionality of WISCkey.

As it functions now, the application uses regular expressions to query the data which cannot be indexed and therefore are less performant. This is fine for the current scale of data within WISCkey, but as data grows, these queries should be changed to use non-regular expressions and indexes. Perhaps another option is to explore leveraging a supplemental technology such as

Elasticsearch ELK stack alongside MongoDB. Elasticsearch provides the ability to move easily beyond simple full-text searches, and could improve the results from queries that were previously hampered by misspelling and other complicating factors.

The investigation of data aggregation provides a set of useful examples but it should not be considered comprehensive. Future work should be performed to understand the data workflow characteristics of more, and different, types of disasters. In the near term, sample data from a third event should be aggregated with a spatiotemporal scale somewhere in the middle of Lewis County Floods and Hurricane Katrina. This could in fact generate new information and cause changes to the schema designed for this application. This data aggregation could also be completed for multiple spatial hierarchies from the same large event. Once this has been completed for several towns, counties, states, etc. a comparison could be made to see how data availability varies within an event and across geographies. This analysis could identify additional institutional variables that could limit data access. This access availability could establish a baseline for which agencies are working toward to provide meaningful data. Once a baseline for data management practices is established this could be used as a guiding force in how other agencies should operate.

The technology used in this application works well, but it's hard to tell if it is the best that could be in place. The use of a NoSQL system could be considered somewhat of a controversial decision. MongoDB, and NoSQL, in general is still a very new technology and it has yet to be seen whether it is a passing fad in file management or a long-term solution. The use of MongoDB was convenient because it allowed for development of the schema and the database at the same time. Now that the schema is developed it could easily be adapted to a traditional RDBMS. More people are familiar with SQL so porting to an RDBMS could make it easier to complete future development of WISCkey. Alternatively, a document store could be developed in multiple database technologies

and then benchmark tests could be run on upload and download speeds. This not only would add to the literature surrounding the choices between traditional and schemaless document stores, but would also ensure the fastest application moving forward.

Works Cited

Artz, Matt. 2008. “What Is The Geographic Approach?” *GIS and Science*. December 21. http://gisandscience.com/2008/12/21/what-is-the-geographic-approach/.

Bartholomew, Daniel. 2010. “SQL vs. NoSQL.” *Linux Journal* 195: 54–59.

Beran, Bora, David Valentine, Catharine Ingen, Ilya Zaslavsky, and Tom Whitenack. 2008. *A Data Model for Environmental Observations*. Technical Report MSR-TR-2008-92. Microsoft. http://research.microsoft.com/pubs/70602/tr-2008-92.pdf.

Bharosa, Nitesh, JinKyu Lee, and Marijn Janssen. 2010. “Challenges and Obstacles in Sharing and Coordinating Information during Multi-Agency Disaster Response: Propositions from Field Exercises.” *Information Systems Frontiers* 12 (1): 49–65. doi:10.1007/s10796-009-9174-z.

Bizer, Christian, Tom Heath, and Tim Berners-Lee. 2009. “Linked Data - The Story So Far.” *Issue on Linked Data, International Journal on Semantic Web and Information Systems* Special Issue.

Brewer, Eric. 2000. “Towards Robust Distributed Systems”. Keynote, PODC.

Cattell, Rick. 2011. “Scalable SQL and NoSQL Data Stores.” *ACM SIGMOD Record* 39 (4): 12. doi:10.1145/1978915.1978919.

Cerami, Ethan. 2002. *Web Services Essentials*. 1st ed. O’Reailly and Associates.

Chang, Stephanie E. 2009. “Urban Disaster Recovery: A Measurement Framework and Its Application to the 1995 Kobe Earthquake.” *Disasters* 34 (2): 303–27. doi:10.1111/j.1467-7717.2009.01130.x.

Cutter, Susan. 2003. “GI Science, Disasters, and Emergency Management.” *Transactions in GIS* 7 (4): 439–45.

Dangermond, Jack. 2007. “GIS--The Geographic Approach”. Industry. *ESRI Understanding Our World*. http://www.esri.com/news/arcnews/fall07articles/gis-the-geographic-approach.html.

Datnow, Amanda, and Vicki Park. 2014. *Data-Driven Leadership*. http://catalogimages.wiley.com/images/db/jimages/9780470594797.jpg.

Deutsch, Donald R. 2013. “The SQL StandarD: How It Happened.” *IEEE Annals of the History of Computing* 35 (2): 72–75. doi:10.1109/MAHC.2013.30.

Fowler, Martin, and Pramod Sadalage. 2013. *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*. Tech books.

Frawley, William, Gregory Piatetsky_Shapiro, and Christopher Matheus. 1992. “Knowledge Discovery in Databases an Overview.” *AI Magazine* 13 (3).

Gilbert, Seth, and Nacy Lynch. 2000. “Brewer’s Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services.”

Haerder, Theo, and Anres Reuter. 1983. “Principles of Transaction-Oriented Database Recovery.” *Computing Surveys* 15 (4).

Huggins, Leonard. 2007. “Comprehensive Disaster Management and Development: The Role of Geoinformatics and Geo-Collaboration in Linking Mitigation and Disaster Recovery in the Eastern Caribbean.” http://d-scholarship.pitt.edu/6252/.

IBM. 2013. “What Is Big Data?” *Big Data at the Speed of Business*. http://www-01.ibm.com/software/data/bigdata/.

Kates, R. W., C. E. Colten, S. Laska, and S. P. Leatherman. 2006. “Reconstruction of New Orleans after Hurricane Katrina: A Research Perspective.” *Proceedings of the National Academy of Sciences* 103 (40): 14653–60. doi:10.1073/pnas.0605726103.

Kumar, Girish, and Rahul Checker. 2013. “Exploring the Different Types of NoSQL Databases.” October. http://www.3pillarglobal.com/blog/exploring-different-types-nosql-databases.

Leavitt, Neal. 2010. “Computer: Will NoSQL Databases Live Up to Their Promise?” *Computer* 43 (2): 12–14.
Lewis County. 2014. “Community Outreach Flood Information - Lewis County Washington.” http://lewiscountywa.gov/communitydevelopment/community-outreach-flood-information.
Maceachren, Alan, Monica Wachowicz, Robert Edsall, Daniel Haug, and Raymon Masters. 1999. “Constructing Knowledge from Multivariate Spatiotemporal Data: Integrating Geographical Visualization with Knowledge Discovery in Database Methods.” *International Journal of Geographical Information Science* 13 (4): 311–34.
Miles, Scott B. 2015. “Foundations of Community Disaster Resilience: Well-Being, Identity, Services, and Capitals.” *Environmental Hazards*, January, 1–19. doi:10.1080/17477891.2014.999018.
Miles, Scott, and Stephanie E. Chang. 2011. “ResilUS: A Community Based Disaster Resilience Model.” *Cartography and Geographic Information Science* 38 (1): 5–20.
Mulligan, Gavin, and Denis Gracanin. 2009. *Proceedings of the 2009 Winter Simulation Conference*. Piscataway: IEEE. http://dl.acm.org/citation.cfm?id=1995456.
ORACLE. 1995. “A Relational Database Overview (The Java™ Tutorials > JDBC(TM) Database Access > JDBC Introduction).” http://docs.oracle.com/javase/tutorial/jdbc/overview/database.html.
Pautasso, Cesare, Olaf Zimmermann, and Frank Leymann. 2008. “RESTful Web Services vs. ‘Big’ Web Services: Making the Right Architectural Decision.” In *Web Service Deployment*. Beijing.
Piercy, Steve. 2014. “WebFrameworks - Python Wiki.” *Web Frameworks for Python*. https://wiki.python.org/moin/WebFrameworks.
Pokorny, Jaroslav. 2011. “NoSQL Databases: A Step to Database Scalability in Web Environment.” In , 278. ACM Press. doi:10.1145/2095536.2095583.
Pradhan, Anu. 2007. “Infrastructure Management Information System Framework Requirements for Disasters.” http://hdl.handle.net/10197/2377.
Provost, Foster, and Tom Fawcett. 2013. “Data Science and Its Relationship to Big Data and Data-Driven Decision Making.” *Big Data* 1 (1): 51–59. doi:10.1089/big.2013.1508.
Rebelic. 2013. “The Four Categories of NoSQL Databases”. Blog. May 28. http://rebelic.nl/2011/05/28/the-four-categories-of-nosql-databases/.
Rodrigue, Chrintine. 2011. “CSU, Long Beach -- Geography 558 -- Hazards and Risk Management.” http://www.csulb.edu/~rodrigue/geog558/lectures/disastercycle.html.
State of Washington. 2011. “2011 Washington State Data Book.”
Stonebraker, Michael. 2010. “SQL Databases v. NoSQL Databases.” *Communications of the ACM* 53 (4): 10. doi:10.1145/1721654.1721659.
Techopedia.com. 2014. “What Is a Relational Database Management System (RDBMS)? - Definition from Techopedia.” http://www.techopedia.com/definition/1235/relational-database-management-system-rdbms.
Tekstenuitleg.net. 2013. “Database Design Tutorial.” May 30. http://en.tekstenuitleg.net/articles/software/database-design-tutorial/intro.html.
U.S Department of Homeland Security. 2013. “UICDS.us - Unified Incident Command and Decision Support.” http://www.uicds.us/.
US Census. 2013. “New Orleans Quickfacts.” http://quickfacts.census.gov/qfd/states/22/2255000.html.
Voigt, Kristina. 1998. “Environmental Information Databases.” http://www.wiley.com/legacy/wileychi/ecc/samples/sample06.pdf.
Wohlstetter, Priscilla, Amanda Datnow, and Vicki Park. 2008. “Creating a System for Data-Driven Decision-Making: Applying the Principal-Agent Framework.” *School Effectiveness and School Improvement* 19 (3): 239–59. doi:10.1080/09243450802246376.

Ye, Xinyue, and Sergio Rey. 2013a. “A Framework for Exploratory Space-Time Analysis of Economic Data.” *The Annals of Regional Science* 50 (1): 315–39. doi:10.1007/s00168-011-0470-4.

———. 2013b. “A Framework for Exploratory Space-Time Analysis of Economic Data.” *The Annals of Regional Science* 50 (1): 315–39. doi:10.1007/s00168-011-0470-4.

Zhang, Dongsong, Lina Zhou, and Jay Nunamaker. 2002. “A Knowledge Management Framework for the Support of Decision Making in Humanitarian Assistance/Disaster Relief.” *Knowledge and Information Systems* 4 (3): 370–85.

Zheng, Li, Chao Shen, and Liang Tang. 2012. “Disaster SitRep - A Vertical Search Engine and Information Analysis Tool in Disaster Management Domain.” *Information Reuse and Integration (IRI), 2012 IEEE 13th International Conference on*, 457–65.

Zur Muehlen, Michael, Jeffrey V. Nickerson, and Keith D. Swenson. 2005. “Developing Web Services Choreography Standards—the Case of REST vs. SOAP.” *Decision Support Systems* 40 (1): 9–29. doi:10.1016/j.dss.2004.04.008.

Appendix A: Application code

```
1.  import pymongo
2.  import gridfs
3.  import json
4.  import hashlib
5.  import urllib2
6.  import logging
7.  import os
8.  import bottle
9.  from types import *
10. from pymongo import MongoClient
11. from bson.objectid import ObjectId
12. from bson import json_util
13.
14. version = "0.0.1"
15. app = application = bottle.Bottle()
16. ############mongoDB  Global variables#################
17. client = MongoClient()
18. db = client.test
19.
20. ############Cross Origin Resource Sharing (CORS)#################
21. @app.hook('after_request')
22. #the server is allowed to deliver the results to any domain
23. def enable_cors():
24.     #bottle.response.headers['Access-Control-Allow-Origin'] = '*'
25.     bottle.response.headers['Access-Control-Allow-
    Methods'] = 'GET, POST, OPTIONS'
26.     #bottle.response.headers['Access-Control-Allow-
    Headers'] = 'Origin, Accept, Content-Type, X-Requested-With, X-CSRF-Token'
27.
28. @app.route('/', method=['OPTIONS', 'POST', 'GET'])
29. def home():
30.     return "You are running version " + version + " of the WISCkey API Server"
31.
32.
33. #############This POSTS information that is uploaded using the DATAForm.html docume
    nt on Udrive########
34. @app.route('/upload', method='POST')
35. def upload():
36.     #FileUpload
37.     """Loads filefield (which is the data from form into the file variable"""
38.     file = bottle.request.files.get('fileField')
39.     """If there is actualll a  file uploaded, this creates an object ext
40.     which is the raw_filename popped after the. which essentially is just the file
    extention"""
41.     if type(file) is NoneType:
42.         pass
43.     else:
44.         #Naming Conventions
45.         ext = file.filename.rsplit('.',1)[1]
46.         """Creates a uniqe hash of the content from file"""
47.         hash = hashlib.md5(file.file.read()).hexdigest()
48.         hashname = hash+"."+ext
49.         file.file.seek(0)
50.
51.         """This block pulls from data used as the standard naming convention for fi
    les
52.             This should be used to when returning data to the user, not to post"""
```

```
53.         spatial = bottle.request.forms.get('spatial')
54.         temporal = bottle.request.forms.get('temporal')
55.         author = bottle.request.forms.get('source.author')
56.         metric = bottle.request.forms.get('indicator.metric')
57.         name = spatial +"_"+temporal+"_"+author+"_"+metric+"."+ext
58.         """This strips the name field of any spaces before or after"""
59.         name = name.strip()
60.         """This replaces spaces with dashes"""
61.         name = name.replace(" ", "-")
62.
63.         #save the file
64.         path = '/var/www/wisc/static/doc/'+hashname
65.         bottle.request.files.get('fileField').save(path)
66.
67.     #Mongo Side upload
68.     """Create an empty dict for data, then create empty lists for each nested objec
    t"""
69.     data = {}
70.     data["source"] = {}
71.     data["indicator"] = {}
72.     data["WISC"] = {}
73.     data["spatial"] = {}
74.     """If a file was uploaded add the filenames, if not pass"""
75.     if type(file) is NoneType:
76.         pass
77.     else:
78.         """All the naming conventions created in the above section"""
79.         data["fileNames"] = {"directory":path,"original":file.raw_filename,"hash":h
    ashname,"new":name,"ext":ext}
80.     """request the form data"""
81.     input = bottle.request.forms.dict
82.     #get rid of keys with empty values
83.     for key,value in input.items():
84.         if value ==['']:
85.             input.pop(key)
86.         elif value ==['-Please Select-']:
87.             input.pop(key)
88.         elif value ==['Select a object to view constructs']:
89.             input.pop(key)
90.         elif value ==['Select a construct to view variables']:
91.             input.pop(key)
92.         else:
93.             pass
94.     for key in input:
95.         """splits the names based on."""
96.         loc = key.split(".")
97.         """if there is only one value aka there was no . use that key"""
98.         if len(loc) == 1:
99.             data[loc[0]] = input[key][0]
100.                #else go through and create two dics with the split values
101.                elif len(loc) == 2:
102.                    data[loc[0]][loc[1]]= input[key][0]
103.            data = json.dumps(data)
104.            if not data:
105.                bottle.abort(400, 'No data received')
106.            entity = json.loads(data)
107.            db['data'].save(entity)
108.            return """
109.            data entry successful<br>
110.            <button onclick="location.href='http://myweb.students.wwu.edu/~kempj2/wi
    sckey/dataform.html'">Return to upload</button><br>
```

```
111.           <button onclick="location.href='http://myweb.students.wwu.edu/~kempj2/wi
    sckey'">home</button>
112.           """
113.
114.       @app.route('/search', method=['OPTIONS', 'POST', 'GET'])
115.       def search():
116.           if bottle.request.method == 'OPTIONS':
117.               return {}
118.           #use headers to communicate to the server for what your doing
119.           bottle.response.headers['Content-Type'] = 'application/json'
120.           #pulls the data from ajax on results.html and assigns them to a variable
     based on what they are
121.           event = bottle.request.forms.get('event')
122.           indicator = bottle.request.forms.get('indicator')
123.           metric = bottle.request.forms.get('metric')
124.           hazard = bottle.request.forms.get('hazard')
125.           ext = bottle.request.forms.get('ext')
126.
127.           if ext =="":
128.               search_parameters = {
129.                   'event':{'$regex':event, '$options':'i'},
130.                   'indicator.type':{'$regex':indicator, '$options':'i'},
131.                   '$or':[
132.                       {'WISC.object':{'$regex':metric, '$options':'i'}},
133.                       {'WISC.object1':{'$regex':metric, '$options':'i'}},
134.                       {'WISC.object2':{'$regex':metric, '$options':'i'}},
135.                       {'WISC.construct':{'$regex':metric, '$options':'i'}},
136.                       {'WISC.construct1':{'$regex':metric, '$options':'i'}},
137.                       {'WISC.construct2':{'$regex':metric, '$options':'i'}},
138.                       {'WISC.variable':{'$regex':metric, '$options':'i'}},
139.                       {'WISC.variable1':{'$regex':metric, '$options':'i'}},
140.                       {'WISC.variable2':{'$regex':metric, '$options':'i'}},
141.                       ],
142.                   'hazard':{'$regex':hazard, '$options':'i'}
143.               }
144.           else:
145.               search_parameters = {
146.                   'event':{'$regex':event, '$options':'i'},
147.                   'indicator.type':{'$regex':indicator, '$options':'i'},
148.                   '$or':[
149.                       {'WISC.object':{'$regex':metric, '$options':'i'}},
150.                       {'WISC.object1':{'$regex':metric, '$options':'i'}},
151.                       {'WISC.object2':{'$regex':metric, '$options':'i'}},
152.                       {'WISC.construct':{'$regex':metric, '$options':'i'}},
153.                       {'WISC.construct1':{'$regex':metric, '$options':'i'}},
154.                       {'WISC.construct2':{'$regex':metric, '$options':'i'}},
155.                       {'WISC.variable':{'$regex':metric, '$options':'i'}},
156.                       {'WISC.variable1':{'$regex':metric, '$options':'i'}},
157.                       {'WISC.variable2':{'$regex':metric, '$options':'i'}},
158.                       ],
159.                   'hazard':{'$regex':hazard, '$options':'i'},
160.                   'fileNames.ext':{'$regex':ext, '$options':'i'}
161.                   }
162.
163.           cur = list(db['data'].find(search_parameters))
164.
165.
166.           # dropped   "for item in cur:item.pop("_id")" and use the default instea
    d which preserves the bson
167.           if cur ==[]:
```

```
168.                cur =[{"indicator" : { "metric" : "", "type" : "" }, "title" : "No M
    etadata Available", "temporal" : "", "WISC" : {  }, "hazard" : "", "summary" : "",
    "source" : { "author" : "" }, "spatial" : "", "event" : "" }]
169.                return json.dumps(cur, default=json_util.default)
170.            else:
171.                return json.dumps(cur, default=json_util.default)
172.
173.        @app.route('/download/<hash>', method=['OPTIONS', 'POST', 'GET'])
174.        def server_static(hash):
175.            print hash
176.            return bottle.static_file(hash, root='/var/www/wisc/static/doc/', downlo
    ad=hash)
177.
178.
179.        @app.route('/wiscinfo/<wisc>', method=['OPTIONS', 'POST', 'GET'])
180.        def get_services(wisc):
181.            search_parameters= {'key':{'$regex':wisc, '$options':'i'}}
182.
183.            cur = list(db['wisc'].find(search_parameters))
184.
185.            return json.dumps(cur, default=json_util.default)
186.
187.
188.        """Run the function on this host using this port reload every time it is edi
    ted"""
189.        if __name__ == '__main__':
190.            bottle.run(app, host='140.160.114.193', port=8080, reloader=True)
```

Appendix B: Landing Page HTML

```
1.  <!DOCTYPE html>
2.  <html>
3.  <head>
4.      <script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
5.      <link rel="stylesheet" href="IndexStyle.css">
6.
7.  </head>
8.
9.  <body>
10.     <div id='page'>
11.         <div class='logo'>
12.             <div class='picture'>
13.                 <img src="http://myweb.students.wwu.edu/~kempj2/wisckey/Logo4.png"/
    >
14.             </div>
15.         </div>
16.         <form id="search-form" name="search-
    form" action="results.html" method="GET" enctype="multipart/form-data">
17.             <input type="text" placeholder="Event name" name="event" id="event" />
    <img src="Info-03.png" onclick="$('#info1').toggle()" /><br />
18.                 <div id="info1" style="display:none">What event are you interested
    in? This is what the news would call the event if they were reporting on the event.
     The popular name that you would use to search scholarly articles pertaining to the
     event. Example Hurricane Katrina</div>
19.             <input type="text" placeholder="Hazard type" name="hazard" id="hazard">
     <img src="Info-03.png" onclick="$('#info2').toggle()" /><br />
20.                 <div id="info2" style="display:none">What type of hazard are you in
    terested in? This is the initial hazard that caused a disaster when it collided wit
    h human development. The types could be biological threat, chemical threat, drought
```

```
    , earthquake,fire, flood, heat, hurricane, landslide, radiation, tornado, tsunami,
    volcano, wild fire, or winter storm.</div>
21.             <input type="text" placeholder="Recovery indicator" name="indicator" id
    ="indicator"> <img src="Info-03.png" onclick="$('#info3').toggle()" /><br />
22.                 <div id="info3" style="display:none">Recovery indicator is the broa
    d category of data you are looking for. What sector does your data below to? Some e
    xamples would be economic, transportation, demographics, etc. </div>
23.             <input type="text" placeholder="Element of WISC" name="metric" id="metr
    ic"> <img src="Info-03.png" onclick="$('#info4').toggle()" /><br />
24.                 <div id="info4" style="display:none">What componenet of WISC does t
    he data you are looking for pertain to? It can be an object, construct, or variable
    . There are 37 components of WISC you could search for. For more information on the
     community resiliece framework WISC please click on the information tab below</div>

25.             <input type="text" placeholder="file type" name="ext" id="ext"> <img sr
    c="Info-03.png" onclick="$('#info5').toggle()" /><br />
26.                 <div id="info5" style="display:none">If a sample dataset was upload
    ed a source, you can search the directory for the types of documents you are intere
    sted in. A few exampels could be pdf, doc, csv, jpg, etc.</div>
27.             <br><button class="submit" id="submit">Submit search criteria</button>

28.         </form>
29.         <button onclick="location.href='http://myweb.students.wwu.edu/~kempj2/wisck
    ey/dataform.html'">Upload Metadata</button>
30.         <button onclick="location.href='http://myweb.students.wwu.edu/~kempj2/wisck
    ey/wisc'">What is WISC?</button>
31.         <button onclick="location.href='http://myweb.students.wwu.edu/~kempj2/wisck
    ey/help.html'">Help file</button>
32.
33. </body>
34.
35. </html>
```

Appendix B-b: Landing page CSS

```
1.  #page{
2.      width: 624px;
3.      margin: 0 auto;
4.      margin-top:20px;
5.  }
6.  #search-form{
7.      padding-top:60px;
8.      padding-bottom:0px
9.      }
10. input, textarea {
11.     padding: 9px;
12.     border: solid 2px #C5C5C5;
13.     outline: 0;
14.     font: normal 13px/100% Verdana, Tahoma, sans-serif;
15.     width: 500px;
16.     background: #FFFFFF url('bg_form.png') left top repeat-x;
17.     background: -webkit-gradient(linear, left top, left 25, from(#FFFFFF), color-
    stop(4%, #EEEEEE), to(#FFFFFF));
18.     background: -moz-linear-gradient(top, #FFFFFF, #EEEEEE 1px, #FFFFFF 25px);
19.     box-shadow: rgba(0,0,0, 0.1) 0px 0px 8px;
20.     -moz-box-shadow: rgba(0,0,0, 0.1) 0px 0px 8px;
21.     -webkit-box-shadow: rgba(0,0,0, 0.1) 0px 0px 8px;
22.   }
23. #info1,#info2,#info3,#info4 ,#info5{
```

```
    width: 500px;
    text-align: left;
    display:block;
}

input:hover, textarea:hover,
input:focus, textarea:focus {
    border-color: #606060;
    -webkit-box-shadow: rgba(0, 0, 0, 0.15) 0px 0px 8px;
  }

button {
    width: auto;
    padding:9px;
    background: #617798;
    border: 0;
    font-size: 14px;
    color: #FFFFFF;
    -moz-border-radius: 5px;
    -webkit-border-radius: 5px;
    float:left;
    margin-right:5px;
    cursor:pointer;
    cursor:hand;
}
#submit{
    width: auto;
    padding:9px;
    background: #617798;
    border: 0;
    font-size: 14px;
    color: #FFFFFF;
    -moz-border-radius: 5px;
    -webkit-border-radius: 5px;
    float:left;
    margin-right:5px;
    cursor:pointer;
    cursor:hand;
    }
.logo {

    padding-bottom:30px;
    background: #617798;
    width:525px;
    border: 1;
    font-size: 14px;
    color: #FFFFFF;
    -moz-border-radius: 5px;
    -webkit-border-radius: 5px;
    }
.picture{
    padding-left:102px;
    padding-top:31px;
    }
```

Appendix C: Upload page HTML

```
<!DOCTYPE html>

<head>
  <meta charset='UTF-8'>
  <script src="http://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js"></script>
  <link rel="stylesheet" href="FormStyle.css">

</head>

<body>
    <div id='page'>
        <h1>Tell us about your data sources</h1>
        <p>
            Please enter metadata regarding where one can find post-disaster recovery
            data. At the very least you should indicate a thoughtful title, the event the data pertains to,
            the type of event it is, a descriptive summary, and the source where the data can be found.
            Fields with an astrisk (*) are required
        </p>

        <form name="input" action="http://140.160.114.190/proxy/proxy.ashx?http://140.160.114.193/upload" method="POST" enctype="multipart/form-data">
            <div class="row header">Descriptive information</div>
            <div class="row">
                <div class="col50">
                    <label for="event">Name of Associated event<a>*</a></label>
                    <input id="event" placeholder="Hurricane Katrina" type="text" name="event" class='field' required></input>

                </div>
                <div class="col50">
                    <label for="type">Type of associated event<a>*</a></label>
                    <select id="hazard" name="hazard" class='select' required><br>

                        <option value="">-Please Select-</option>
                        <option value="biological">Biological Threat</option>
                        <option value="chemical">Chemical Threat</option>
                        <option value="drought">Drought</option>
                        <option value="earthquake">Earthquake</option>
                        <option value="fire">Fire</option>
                        <option value="flood">Flood</option>
                        <option value="heat">Heat</option>
                        <option value="hurricane">Hurricane</option>
                        <option value="landslide">Landslide</option>
                        <option value="radiation">Radiation</option>
                        <option value="tornado">Tornado</option>
                        <option value="tsunami">Tusnami</option>
                        <option value="volcano">Volcano</option>
                        <option value="wildfires">Wildfires</option>
                        <option value="winter.storm">Winter Storm</option>
                    </select>
                </div>
            <div class="row">
                <div class="col50">
```

```
51.                     <label for="title">Descriptive title of data available from sou
    rce<a>*</a></label>
52.                     <input id='title' placeholder="August Road Closure data for Orl
    eans Parish" type="text" name="title" class='field' required></input>
53.                 </div>
54.                 <div class="col50">
55.                     <label for="file">File</label>
56.                     <input id="file" type="file" name="fileField" class="file"></in
    put>
57.
58.                 </div>
59.             <div class="row">
60.                 <div class="col100">
61.                     <label for="summary">Summary of source and data available<a>*</
    a></label>
62.                     <textarea id="summary" name="summary" class='notes' required>
63.                 </div>
64.             </div>
65.             <div class="row header">Descriptive information associated with datasou
    rce</div>
66.             <div class="row">
67.                 <div class="col50">
68.                     <label for="indicator type">Associated indicator type<a>*</a></
    label>
69.                     <input id="indicator type" placeholder="Transportation" type="t
    ext" name="indicator.type" class='field' required></input>
70.                 </div>
71.                 <div class="col50">
72.                     <label for="indicator metric">Associated indicator metric<a>*</
    a></label>
73.                     <input id="indicator metric" placeholder="Road Closure"  type="
    text" name="indicator.metric" class='field' required></input>
74.                 </div>
75.             <div class="row header">Elements of WISC that apply to source  <a href=
    "http://myweb.students.wwu.edu/~kempj2/wisckey/wisc" target="_blank"><img src="Info
    -03.png" alt="WISC"></a></div>
76.             <div class="row">
77.                 <div class="col33">
78.                     <div class="row header">Element 1<a>*</a></div>
79.                     <label for="object">Object</label><select id="object" name="WIS
    C.object" required>
80.                         <option value="" >-Please Select-</option>
81.                         <option value="infrastructure">Infrastructure</option>
82.                         <option value="community">Community</option>
83.                     </select>
84.                     <label for="construct">Construct</label><select id="construct"
    name="WISC.construct" required>
85.                         <option>Select a object to view constructs </option>
86.                     </select>
87.                     <label for="variable">Variable</label><select id="variable" nam
    e="WISC.variable" required>
88.                         <option>Select a construct to view variables</option>
89.                     </select>
90.                 </div>
91.                 <div class="col33">
92.                     <div class="row header">Element 2</div>
93.                     <label for="object">Object</label><select id="object1" name="WI
    SC.object1">
94.                         <option value="" >-Please Select-</option>
95.                         <option value="infrastructure">Infrastructure</option>
96.                         <option value="community">Community</option>
```

```
                    </select>
                    <label for="construct">Construct</label><select id="construct1" name="WISC.construct1">
                        <option>Select a object to view constructs </option>
                    </select>
                    <label for="variable">Variable</label><select id="variable1" name="WISC.variable1">
                        <option>Select a construct to view variables</option>
                    </select>
                </div>
                <div class="col33">
                    <div class="row header">Element 3</div>
                    <label for="object">Object</label><select id="object2" name="WISC.object2">
                        <option value="" >-Please Select-</option>
                        <option value="infrastructure">Infrastructure</option>
                        <option value="community">Community</option>
                    </select>
                    <label for="construct">Construct</label><select id="construct2" name="WISC.construct2">
                        <option>Select a object to view constructs </option>

                    </select>
                    <label for="variable">Variable</label><select id="variable2" name="WISC.variable2">
                        <option>Select a construct to view variables</option>
                    </select>
                </div>
            <div class="row header">Finest resolution(accuracy) of data offered from source</div>
            <div class="row">
                <div class="col50">
                <label for="spatial">Spatial Resolution<a>*</a></label><select required id="spatial" name="spatial" class='select'><br>
                        <option value="" >-Please Select-</option>
                        <option value="national">National</option>
                        <option value="state">State</option>
                        <option value="msa">Metropolitan Statistical Area</option>
                        <option value="county">County</option>
                        <option value="city">City</option>
                        <option value="neighborhood">Neighborhood</option>
                        <option value="csa">Census Designated Area</option>

                        <option value="household">Household</option>
                        </select>
                </div>
                <div class="col50">
                <label for="temporal">Temporal Resolution<a>*</a></label><select id="temporal" name="temporal" class='select' required><br>
                        <option value="" >-Please Select-</option>
                        <option value="totaled">Totaled</option>
                        <option value="yearly">Yearly</option>
                        <option value="quarterly">Quarterly</option>
                        <option value="monthly">Monthly</option>
                        <option value="weekly">Weekly</option>
                        <option value="daily">Daily</option>
                        </select>
```

```
                        </div>
                    <div class="row header">Author of the dataset</div>
                    <div class="row">
                        <div class="col50">
                            <label for="source author">Authoring person, agency, or group<a>*</a></label>
                            <input id="source author" placeholder="Louisiana DOTD" type="text" name="source.author" class='field' required></input>
                        </div>
                        <div class="col50">
                            <label for="source contact">Author Contact information</label>
                            <input id="source contact" placeholder="John Doe, (111)111-
1111, Jdoe@example.com" type="text" name="source.contact"class='field'></input>
                        </div>
                    </div>
                    <div class="row">
                        <div class="col50">
                            <label for="source url">Source URL</label>
                            <input id="source url"placeholder="www.exmple.com/dotd/road/closure" type="text" name="source.url"class='field'></input>
                        </div>
                        <div class="col50">
                            <label for="source publisher">Source Publisher</label>
                            <input id="source publisher"placeholder="BigPublishing Unlimited" type="text" name="source.publisher"class='field'></input>
                        </div>
                    </div>
                    <div class="row">
                        <div class="col50">
                            <label for="source journal">Primary Source</label>
                            <input id="source journal" placeholder="Journal of Science" type="text" name="source.journal"class='field'></input>
                        </div>
                        <div class="col50">
                            <label for="source date">Source Date</label>
                            <input id="source date" type="date" name="source.date"class='field'></input>
                        </div>
                    </div>
                    <div class="row">
                        <label for="submit"></label>
                        <button class="submit" value="Submit">Submit</button>
                    </div>

                </form>
            </div>
                  <script>
                    //This sets the list of all the difrerent elements of WISC
                        var selects = {
                            "infrastructure": ["-Please Select-","Capital","Services"],
                            "community": ["-Please Select-","Identity","Well-being"],
                            "dynamic":["-Please Select-","Metabolism","Geography","Sufficency"],
                            "capital":["-Please Select-","Social","Political","Human","Economic","Built","natural"],
```

```
                            "services":["-Please Select-
","Rivalrousness","Excludability","Marketability","Redundancy","Substitutability","
Robustness","Gravity","Centrality","Connectedness"],
                            "identity":["-Please Select-
","Equity","Continuity","Esteem","Efficacy","Empowerment","Distinctiveness","Divers
ity","Adaptability"],
                            "well-being":["-Please Select-
","Satisfaction","Affiliation","Autonomy","Health","Security","Material Needs"],
                            "metabolism":["-Please Select-
","Activities of Governence","Intervention","Disruption","Provision","Consumption"]
,
                            "geography":["-Please Select-
","Space","Time","Scale","Topology"],
                            "sufficency":["-Please Select-
","Supply","Demand","Speed","Adequacy","Externality"]


                        }
                        //Set an event listener on the objects dropdown
                        $('#object').on('change', function(){
                            //set the construct to empty
                            $('#construct').empty();
                            //select the value from the object dropdown
                            $.each(selects[$('#object').val()], function(i,value) {

                                //append the value from object drop down and get the
 value from selects
                                $('#construct').append("<option>" + value + "</optio
n>")
                            });

                        })
                        //this second function is a listener for the second dropdown
. Check for a change in the construct dropdown
                        $('#construct').on('change', function(){
                            //set third (variable) drop down to empty
                            $('#variable').empty();
                            //for the value you chose in construct convert to lowerc
ase and run function
                            $.each(selects[$('#construct').val().toLowerCase()], fun
ction(i,value) {
                                //append the value from the construct to the variabl
e.
                                $('#variable').append("<option>" + value + "</option
>")
                            });
                        })
                        //Set an event listener on the objects dropdown(1)
                        $('#object1').on('change', function(){
                            //set the construct to empty
                            $('#construct1').empty();
                            //select the value from the object dropdown
                            $.each(selects[$('#object1').val()], function(i,value) {

                                //append the value from object drop down and get the
 value from selects
                                $('#construct1').append("<option>" + value + "</opti
on>")
                            });

                        })
```

```
                        //this second function is a listener for the second dropdown
    . Check for a change in the construct dropdown(1)
                        $('#construct1').on('change', function(){
                            //set third (variable) drop down to empty
                            $('#variable1').empty();
                            //for the value you chose in construct convert to lowerc
    ase and run function
                            $.each(selects[$('#construct1').val().toLowerCase()], fu
    nction(i,value) {
                                //append the value from the construct to the variabl
    e.
                                $('#variable1').append("<option>" + value + "</optio
    n>")
                            });
                        })
                        //Set an event listener on the objects dropdown(2)
                        $('#object2').on('change', function(){
                            //set the construct to empty
                            $('#construct2').empty();
                            //select the value from the object dropdown
                            $.each(selects[$('#object2').val()], function(i,value) {

                                //append the value from object drop down and get the
     value from selects
                                $('#construct2').append("<option>" + value + "</opti
    on>")
                            });

                        })
                        //this second function is a listener for the second dropdown
    . Check for a change in the construct dropdown(2)
                        $('#construct2').on('change', function(){
                            //set third (variable) drop down to empty
                            $('#variable2').empty();
                            //for the value you chose in construct convert to lowerc
    ase and run function
                            $.each(selects[$('#construct2').val().toLowerCase()], fu
    nction(i,value) {
                                //append the value from the construct to the variabl
    e.
                                $('#variable2').append("<option>" + value + "</optio
    n>")
                            });
                        })

                    </script>

        </body>
```

Appendix C-b: Upload page CSS

```
body {
  text-align:left;
}
h1{
    text-align:left;
}
```

```
p{
    text-align:left;
    font-weight:bold;
    padding-top:0px;
    margin-top:16px;

}
textarea{
    -webkit-box-sizing: border-box;
    -moz-box-sizing: border-box;
    box-sizing: border-box;
}

#page{
    width: 950px;
    margin: 0 auto;
}


.field {
  padding: 4px;
  border: solid 1px #C5C5C5;
  outline: 0;
  font: normal 13px/100% Verdana, Tahoma, sans-serif;
  width: 300px;
  }
.Rfield {
  padding: 4px;
  border: solid 1px #E87B7B;
  outline: 0;
  font: normal 13px/100% Verdana, Tahoma, sans-serif;
  width: 300px;
}
.notes {
  padding: 4px;
  border: solid 1px #C5C5C5;
  outline: 0;
  font: normal 13px/100% Verdana, Tahoma, sans-serif;
  width: 625px;
  height:100px;
}
select {
  border: solid 1px #C5C5C5;
  padding:2.5px;
  outline: 0;
  font: normal 13px/100% Verdana, Tahoma, sans-serif;
}
input:hover, textarea:hover,
input:focus, textarea:focus {
  border-color: #606060;
  -webkit-box-shadow: rgba(0, 0, 0, 0.15) 0px 0px 8px;
}

button.submit {

    width: 150px;
    padding:9px;
    background: #617798;
    border: 0;
    font-size: 14px;
    color: #FFFFFF;
```

```
    -moz-border-radius: 5px;
    -webkit-border-radius: 5px;
    float:left;
    margin-right:5px;
    cursor:pointer;
    cursor:hand;

}

.radio {
  padding:4px;
}
label {
  display: block;
  padding-top:8px;
  width: 300px;
  text-align:left;
  margin-right: 10px;
}

.header {
    font-weight: bold;
    padding-top:20px;
    padding-bottom:20px;
}
.file{
  border:20px;
}

.row {
    clear: both;
}

       .col33{
           width: 310px;
           float: left;
       }

       .col50{
           width: 350px;
           float: left;
       }

       .col100{
           width: 960px;
           float: left;
       }
       a{
           color:#FF0000;

       }
```

Appendix D: Results page

```
<!DOCTYPE html>
<html>
<head>
```

```
4.      <title>WiskKey Results</title>
5.      <script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
6.      <link rel="stylesheet" href="ResultStyle.css">
7.
8.
9.      <script>
10.         function getURLParameter(name) {
11.             return decodeURIComponent((new RegExp('[?|&]' + name + '=' + '([^&;]+?)
    (&|#|;|$)').exec(location.search)||[,""])[1].replace(/\+/g, '%20'))||null;
12.         }
13.
14.
15.         function parseResults(results) {
16.
17.             console.log(results);
18.             for (var i = 0; i < results.length; i++) {
19.                 var html = '';
20.                 html += "<div>";
21.                 html += "<b>" + results[i].title + "</b>";
22.                 html += "<br><i>Hazard Type:</i> " + results[i].hazard;
23.                 html += "<br><i>Associated event:</i> " + results[i].event;
24.                 html += "<br><i>Source:</i> " + results[i].source.author;
25.                 html += "<br><i>Summary:</i> " + results[i].summary;
26.                 html += "<br><i>Recovery Indicator:</i> " + results[i].indicator.ty
    pe;
27.                 html += "<br><i>Metric within indicator:</i> " + results[i].indicat
    or.metric;
28.                 html += "<br><i>Spatial Resolution:</i> " + results[i].spatial;
29.                 html += "<br><i>Temporal Resolution:</i> " + results[i].temporal +"
    <br>";
30.
31.                 /// TERNARY:
32.                 // Condition ? true : false
33.                 html += results[i].source.publisher ? "<i>Publisher:</i> " + result
    s[i].source.publisher   + "<br>" : "";
34.                 html += results[i].source.author ? "<i>Author:</i> " + results[i].s
    ource.author   + "<br>" : "";
35.                 html += results[i].source.url ? "URL: " + results[i].source.url   +
     "<br>" : "";
36.                 ///journal is now priamry author via scott's edits
37.                 html += results[i].source.journal ? "<i>Primary Source:</i> " + res
    ults[i].source.journal   + "<br>" : "";
38.                 html += results[i].source.contact ? "<i>Contact:</i> " + results[i]
    .source.contact   + "<br>" : "";
39.                 html += results[i].source.date ? "<i>Date:</i> " + results[i].sourc
    e.date   + "<br>" : "";
40.
41.                 html += results[i].notes ? "<i>Notes:</i> " + results[i].notes   +
    "<br>" : "";
42.                 html +="<i>WISC Objects:</i>"
43.                 html += results[i].WISC.object ? "<br>    " + "<a href='http://140.
    160.114.193/wiscinfo/" + results[i].WISC.object+ "'>"+results[i].WISC.object+"</a>"
    :"";
44.                 html += results[i].WISC.object1 ? "<br>    " + "<a href='http://140
    .160.114.193/wiscinfo/" + results[i].WISC.object1+ "'>"+results[i].WISC.object1+"</
    a>":"";
45.                 html += results[i].WISC.object2? "<br>    " + "<a href='http://140.
    160.114.193/wiscinfo/" + results[i].WISC.object2+ "'>"+results[i].WISC.object2+"</a
    >":"";
46.                 html +="<br><i>WISC Constructs:</i>"
```

```
                html += results[i].WISC.construct ? "<br>    " + "<a href='http://140.160.114.190/proxy/proxy.ashx?http://140.160.114.193/wiscinfo/" + results[i].WISC.construct+ "'>"+results[i].WISC.construct+"</a>":"";
                html += results[i].WISC.construct1 ? "<br>    " + "<a href='http://140.160.114.190/proxy/proxy.ashx?http://140.160.114.193/wiscinfo/" + results[i].WISC.construct1+ "'>"+results[i].WISC.construct1+"</a>":"";
                html += results[i].WISC.construct2 ? "<br>    " + "<a href='http://140.160.114.190/proxy/proxy.ashx?http://140.160.114.193/wiscinfo/" + results[i].WISC.construct2+ "'>"+results[i].WISC.construct2+"</a>":"";
                html += "<br><i>WISC Variables:</i>"
                html += results[i].WISC.variable ? "<br>    " + "<a href='http://140.160.114.190/proxy/proxy.ashx?http://140.160.114.193/wiscinfo/" + results[i].WISC.variable+ "'>"+results[i].WISC.variable+"</a>":"";
                html += results[i].WISC.variable1 ? "<br>    " + "<a href='http://140.160.114.190/proxy/proxy.ashx?http://140.160.114.193/wiscinfo/" + results[i].WISC.variable1+ "'>"+results[i].WISC.variable1+"</a>":"";
                html += results[i].WISC.variable2 ? "<br>    " + "<a href='http://140.160.114.190/proxy/proxy.ashx?http://140.160.114.193/wiscinfo/" + results[i].WISC.variable2+ "'>"+results[i].WISC.variable2+"</a>":"";

                //link to bottle that will serve up the document
                html += results[i].fileNames ? "<br>Sample: " + "<a href='http://140.160.114.190/proxy/proxy.ashx?http://140.160.114.193/download/" + results[i].fileNames.hash + "'>" + results[i].fileNames.new +"</a>" : "";

                html += "<br>_____________________________________________________";

                html += "</div><br>";

                $('#results').append(html);
            }
        }




        $.ajax({
            type: "POST",
            url: 'http://140.160.114.190/proxy/proxy.ashx?http://140.160.114.193/search',
            data: {
                'event' : getURLParameter('event'),
                'indicator' : getURLParameter('indicator'),
                'hazard' : getURLParameter('hazard'),
                'metric' : getURLParameter('metric'),
                'ext': getURLParameter('ext')
            },
            success: parseResults,
            dataType: 'json'

        });


    </script>


</head>

```

```
<body>

<div id="results">

</div>


      </body>
      </html>
```

Appendix E: Help documentation HTML

```
<!DOCTYPE html>
<html>
<head>
    <link rel="stylesheet" href="HelpStyle.css">
</head>
<body>

    <h1>Welcome to the help documentation!</h1>
    <div>
        This application is meant as a proof of concept for a better way of conduct
ing disaster recovery  by providing a metadata repository
        to aid in research. What is metadata? Great question! Metadata is data abou
t data. Essentially each document stored in this application is a
        metadata document which provides valuable information on where to find a pa
rticular dataset.
    </div>
    <div>
        More than providing a roadmap this application is about imbedding useful, s
earchable,
        information within the metadata. Information like at what scale both tempor
ally and spatially a
        particular source of information is useful at.
    </div>
    <div>
        The application goes further by incorporating elements of community resilie
nce within the data
        as well. This is useful because it promotes the use of these the WISC frame
work in helping
        emergency management and researchers apply the theory to their work. The go
al of applying such a
        framework in the overall well-being of a community
    </div>
    <h1> Searching:</h1>
    <div>
    Submitting a search to the application is very easy. Fill in one, or a combinat
ion, of  any of the
    5 search boxes on the WISCkey homepage. If you are unsure of what each box is a
sking for click
    the info button to the right of that search box.
    </div>
    <h1>Uploading:</h1>
    <div>
    Just as easy as searching uploading only requires a little bit of necessary inf
ormation. Each
    mandatory field is marked with a red asterisk. Additional fields while unnecess
ary are highly
```

```
    recommended. Remember that this is not an actual piece of data, this is metadat
a, so itâ€™s almost
    like you are writing a citation for data you have found.
    </div>

</body>
</html>
```

Appendix E-b: Help documentation CSS

```
body {
    margin: 0 auto;

    width:800px;

    }

div{
    margin-top:50px;
    margin-bottom:50px;
    text-align:justify;
    }

h1{
    text-align:center;
    padding:9px;
    background: #617798;
    border: 0;
    font-size: 20px;
    color: #FFFFFF;
    -moz-border-radius: 5px;
    -webkit-border-radius: 5px;
    margin-right:5px;

    }
```

Appendix F: WISC documentation HTML

```
<!DOCTYPE html>
<html>
<head>
    <link rel="stylesheet" href="WISC_index_style.css">
    <meta charset=utf-8 />
    <title>WISC</title>
</head>
<body>
  <div class="img-container">
    <div class="hot-spots">
      <div class="hot-spot settlment"> Human Settlement
        <div class="info">
        </div>
      </div>
      <div class="hot-spot infrastructure"> Infrastructure
```

```
        <div class="info">
        </div>
      </div>
      <div class="hot-spot community"> Community
        <div class="info">
        </div>
      </div>
      <div class="hot-spot well-being">Well-Being
        <div class="info">
          <h2>Well-Being</h2>
          <p>Well-being is a high and non-
disparate level of mental and behavioral health, role functioning, and quality of l
ife. The goal of community resilience should go beyond safety or functioning and ul
timately ensure the well-being of different communities and their members. Well-
being is more than just market based measures and is both an appropriate and necess
ary standard.  There is a distinct separation within the variables of well-
being between agents with the capabilities and those with achievements.</p>
        </div>
      </div>
      <div class="hot-spot identity">Identity
        <div class="info">
          <h2>Identity</h2>
          <p>Community identity has been empirically linked to indicators and, thus
, variables of well-
being. Identity provides a strong connection to root causes of social vulnerability
 because it constitutes a foundation for a variety of social effects, from humansâ€
™ ability to feel, act, and think as members of a social group to intergroup behavi
ors such as discrimination, confrontation, and cooperation. </p>
        </div>
      </div>
      <div class="hot-spot services">Services
        <div class="info">
          <h2>Services</h2>
          <p>A service is the benefits people derive from capital, where the end or
 goal is sustainable human well-
being. Services are how individuals and communities access and utilize capital towa
rds promotion of their well-being. </p>
        </div>
      </div>
      <div class="hot-spot capitals">Capitals
        <div class="info">
          <h2>Capitals</h2>
          <p>Capital refers to a stock of assets used to create or obtain additiona
l assets or derive services. In the community sense, it refers to any asset, whethe
r corporeal, material, or non-
material, that is utilized as part of the metabolic flows supporting human settleme
nts.</p>
        </div>
      </div>
      <div class="hot-spot satisfaction">Satisfaction
        <div class="info">
          <h2>Satisfaction</h2>
          <p>Satisfaction is hard to categorize other than to say it is a level whi
ch most agents have the capabilities of imagination, free emotions, reason, play, a
nd interaction with one another.</p>
        </div>
      </div>
      <div class="hot-spot affiliation">Affiliation
        <div class="info">
          <h2>Affiliation</h2>
```

```
          <p>The variable of affiliation, when achieved, means that an agent is a p
roductive member of the larger community that helps to promote variables that lead
to the well-being.  </p>
        </div>
      </div>
      <div class="hot-spot health">Health
        <div class="info">
          <h2>Health</h2>
          <p>Mental and physical health are both represented by the variable of hea
lth. The achievement of health is the cumulative use of all other variables possibl
e to alleviate illness and death.</p>
        </div>
      </div>
      <div class="hot-spot autonomy">Autonomy
        <div class="info">
          <h2>Autonomy</h2>
          <p>Autonomy is the level of independence an agent has in using capitals a
nd services to produce well-
being. Higher levels of autonomy are associated with an increase in well-
being. </p>
        </div>
      </div>
      <div class="hot-spot security">Security
        <div class="info">
          <h2>Security</h2>
          <p>The variable of security is an agentâ€™s ability to feel safe and secu
re from stress events. Predictability is also a component of security because it de
creases vulnerability. When agent achieves the variable of security metal anguish i
s avoided because their well-being is more secured.  </p>
        </div>
      </div>
      <div class="hot-spot material">Material Needs
        <div class="info">
          <h2>Material Needs </h2>
          <p>A variety of conditions that need to be satisfied in order to keep deg
radation from occurring to well-
being that come in physical forms such as food, housing, heat, energy. A poor level
 of the variable of material needs can result in poverty. </p>
        </div>
      </div>
      <div class="hot-spot equality">Equality
        <div class="info">
          <h2>Equality</h2>
          <p>Equity is the stock, investment, and ownership an agent has within a c
ommunity. In the case of resilience when something is inequitable it is unjust, whi
ch equates to poor access to community infrastructure. </p>
        </div>
      </div>
      <div class="hot-spot esteem">Esteem
        <div class="info">
          <h2>Esteem</h2>
          <p>Esteem goes hand in hand with Efficacy, and it is the respect or admir
ation an agent or community has for the organizations that respond during and after
 stress events, the larger the efficacy the higher the esteem.</p>
        </div>
      </div>
      <div class="hot-spot empowerment">Empowerment
        <div class="info">
          <h2>Empowerment</h2>
```

```
98.           <p>Empowerment refers to whether an agent or human settlement has the abi
    lity to access the infrastructure they desire. The ability to participate in the re
    construction and recovery of their collective identity is also key.</p>
99.         </div>
100.             </div>
101.             <div class="hot-spot diversity">Diversity
102.               <div class="info">
103.                 <h2>Diversity</h2>
104.                 <p>Diversity is the variety of variables that that exists within,
    or is connected to, community infrastructure. More important than the quantity of i
    nfrastructure is the diversity of access to infrastructure.</p>
105.               </div>
106.             </div>
107.             <div class="hot-spot continuity">Continuity
108.               <div class="info">
109.                 <h2>Continuity</h2>
110.                 <p>The overall maintenance of identity is the variable of continui
    ty. If threats or access to infrastructure the continuity can be disrupted which ca
    n leave agents or groups isolated which can have negative effects on the overall we
    ll-being of the system.</p>
111.               </div>
112.             </div>
113.             <div class="hot-spot efficacy">Efficacy
114.               <div class="info">
115.                 <h2>Efficacy</h2>
116.                 <p>Efficacy is the trust an agent or community has in the overall
    effectiveness of organizations to take action during, or after a stress event.</p>

117.               </div>
118.             </div>
119.             <div class="hot-spot distictiveness">Distictiveness
120.               <div class="info">
121.                 <h2>Distictiveness</h2>
122.                 <p>Distinctiveness is strongly linked to peopleâ€™s sense of place
     and attachment. This allows members of a community to differentiate themselves and
     compare relative monetary and non-monetary costs</p>
123.               </div>
124.             </div>
125.             <div class="hot-spot adaptability">Adaptability
126.               <div class="info">
127.                 <h2>Adaptability</h2>
128.                 <p>Adaptability can be thought of as the flexibility or absorption
     rate of a system. Adaptability often times needs to be a creative solution, but ca
    reful consideration needs to be given in order to ensure a community is not absorbi
    ng and then returning to pre-event vulnerabilities. </p>
129.               </div>
130.             </div>
131.              <div class="hot-spot rivalrousness">Rivalrousness
132.               <div class="info">
133.                 <h2>Rivalrousness</h2>
134.                 <p>If a service is rivalrous it is finite, so once the capital tha
    t it is derived from is used up, it cannot be continued</p>
135.               </div>
136.             </div>
137.             <div class="hot-spot excludability">Excludability
138.               <div class="info">
139.                 <h2>Excludability</h2>
140.                 <p>When excludable, a service can be realistically kept from certa
    in groups via explicit or institutionalized access.</p>
141.               </div>
142.             </div>
```

```
143.             <div class="hot-spot marketability">Marketability
144.               <div class="info">
145.                 <h2>Marketability</h2>
146.                 <p>Services can be either maintained as a public commons or manage
    d by private market which drives the marketability of that service.</p>
147.               </div>
148.             </div>
149.             <div class="hot-spot redundancy">Redundancy
150.               <div class="info">
151.                 <h2>Redundancy</h2>
152.                 <p>a redundant service means that there is a backup service of the
     same type if one were to fail.</p>
153.               </div>
154.             </div>
155.             <div class="hot-spot robustness">Robustness
156.               <div class="info">
157.                 <h2>Robustness</h2>
158.                 <p>A robust service means that it resists degradation or loss duri
    ng a stress event</p>
159.               </div>
160.             </div>
161.             <div class="hot-spot gravity">Gravity
162.               <div class="info">
163.                 <h2>Gravity</h2>
164.                 <p>High gravity services are important services with respect to di
    fferent agents or communities such as clean water, but gravity is also highly depen
    dent on the unit of </p>
165.               </div>
166.             </div>
167.             <div class="hot-spot centrality">Centrality
168.               <div class="info">
169.                 <h2>Centrality</h2>
170.                 <p>Centrality represents the degree of how centralized the service
    s of infrastructure are.</p>
171.               </div>
172.             </div>
173.             <div class="hot-spot substitutability">Substitutability
174.               <div class="info">
175.                 <h2>Substitutability</h2>
176.                 <p>Substitution of a service would be replacing a service with a d
    ifferent type but still receiving the same benefits.</p>
177.               </div>
178.             </div>
179.             <div class="hot-spot connectedness">Connectedness
180.               <div class="info">
181.                 <h2>Connectedness</h2>
182.                 <p>Connectedness of services shows us how related service benefits
     are within a network. The idea here is if the variable of connectedness is high, t
    he benefits from that service go up relative to how many units are active within th
    e network</p>
183.               </div>
184.             </div>
185.             <div class="hot-spot social">Social
186.               <div class="info">
187.                 <h2>Social</h2>
188.                 <p>Social capital refers to the bonds that hold people and groups
    together as well as the traditions and languages within a community and acts as a c
    ohesive force.</p>
189.               </div>
190.             </div>
191.              <div class="hot-spot political">Political
```

```
192.               <div class="info">
193.                 <h2>Political</h2>
194.                 <p> Political capital reflects on the access to power via politica
    l channels and the ability to contribute to the community.</p>
195.               </div>
196.             </div>
197.             <div class="hot-spot economic">Economic
198.               <div class="info">
199.                 <h2>Economic</h2>
200.                 <p>Economic capital is the resources that can be used to invest in
     the development of businesses and support civic and social entrepreneurship.</p>
201.               </div>
202.             </div>
203.             <div class="hot-spot human">Human
204.               <div class="info">
205.                 <h2>Human</h2>
206.                 <p>Human capital is the skills and abilities of the people inside
    a community, especially those in leadership positions.</p>
207.               </div>
208.             </div>
209.              <div class="hot-spot built">Built
210.               <div class="info">
211.                 <h2>Built</h2>
212.                 <p>Built Capital is the infrastructural interface between natural
    capital and all other capital variables, which is to say that built capital control
    s the access to natural capital from all other capitals</p>
213.               </div>
214.             </div>
215.               <div class="hot-spot natural">Natural
216.               <div class="info">
217.                 <h2>Natural</h2>
218.                 <p>Natural capital refers to the natural resources of such as weat
    her, amenities, natural beauty, timber, oil, etc.</p>
219.               </div>
220.               </div>
221.               <div class="hot-spot cultural">Cultural
222.               <div class="info">
223.                 <h2>Cultural</h2>
224.                 <p>Cultural capitals are the traditions and languages within a com
    munity, together with social capital they form a cohesive force within the communit
    y</p>
225.               </div>
226.             </div>
227.           </div>
228.         </div>
229.           <div class="desc">
230.               WISC is this really neat comprehensive framework for community resil
    ience. Go ahead and see what it's all about by holding down your left mouse bottom
    over any element of WISC for a description exaplining each component. Essentially i
    ts the idea that communities will use capitals to gerneate services, which in turn
    are consumed to create idenity and overall well-being.
231.           </div>
232.       </body>
233.       </html>
```

Appendix F-b: WISC documentation CSS

```
1.  .img-container {
2.
3.    background: url('http://myweb.students.wwu.edu/~kempj2/WiscKey/WISC3') center cen
    ter;
4.    margin: 0 auto;
5.    position: relative;
6.    width: 554px;
7.    max-width: 100%;
8.    height: 487px;
9.
10. }
11.
12. .hot-spots {
13.
14.   background: rgba(255,255,255,0.5);
15.   margin: 0 auto;
16.   position: relative;
17.   width: 0px;
18.
19. }
20.
21. .hot-spot {
22.   color: #000000;
23.   padding: 5px;
24.   position: absolute;
25.   text-align: center;
26.   width: 140px;
27.   height: 10px;
28.
29. }
30.
31. .hot-spot:active {
32.
33.   background-color: #666;
34.   cursor: pointer;
35.
36. }
37.
38. .hot-spot:active .info {
39.
40.   display: block;
41.
42. }
43.
44.
45. .info {
46.
47.   background-color: #ccc;
48.   border: 1px solid #000;
49.   color: #222;
50.   display: none;
51.   padding: 5px;
52.   position: absolute;
53.   top: 0;
54.   width: 250px;
55.   z-index: 10;
56.   text-align: left;
57.    transform:none;
```

```
  -ms-transform:none;
  -webkit-transform:none;

}
h2 {

  font-size: 20px;
}
p {

  font-size: 12px;
}
.settlment {

  top: 235px;
  left: -344px;
  font-weight:bold;
  font-size: 20px;
  color: #FFFFFF;
  transform:rotate(-90deg);
  -ms-transform:rotate(-90deg); /* IE 9 */
  -webkit-transform:rotate(-90deg); /* Opera, Chrome, and Safari */

}
.infrastructure {

  top: 356px;
  left: 170px;
  font-weight:bold;
  font-size: 20px;
  color: #FFFFFF;
  transform:rotate(-90deg);
  -ms-transform:rotate(-90deg); /* IE 9 */
  -webkit-transform:rotate(-90deg); /* Opera, Chrome, and Safari */

}
.community {

  top: 110px;
  left: 170px;
  font-weight:bold;
  font-size: 20px;
        color: #FFFFFF;
        transform:rotate(-90deg);
        -ms-transform:rotate(-90deg); /* IE 9 */
        -webkit-transform:rotate(-90deg); /* Opera, Chrome, and Safari */

      }

      .well-being {

        top: 1px;
        left: -64px;
        font-weight:bold;
        font-size: 20px

      }

      .identity {

        top: 148px;
```

```
        left: -65px;
        font-weight:bold;
        font-size: 20px

      }
      .services {

        top: 236px;
        left: -65px;
        font-weight:bold;
        font-size: 20px

      }
      .capitals {

        top: 328px;
        left: -65px;
        font-weight:bold;
        font-size: 20px

      }
      .satisfaction {

        top: 30px;
        left: -65px;
        font-size: 20px
      }
      .affiliation {

        top: 60px;
        left: -120px;
        font-size: 20px
      }
      .health {

        top: 90px;
        left: -120px;
        font-size: 20px
      }
      .autonomy {

        top: 60px;
        left: -20px;
        font-size: 20px
      }
      .security {

        top: 90px;
        left: -20px;
        font-size: 20px
      }
      .material {

        top: 120px;
        left: -65px;
        font-size: 20px
      }
      .equality {

        top: 175px;
        left: -255px;
```

```
        font-size: 20px
      }
      .esteem {

        top: 175px;
        left: -155px;
        font-size: 20px
      }
      .empowerment {

        top: 175px;
        left: -30px;
        font-size: 20px
      }
      .diversity {

        top: 175px;
        left: 100px;
        font-size: 20px
      }
      .continuity {

        top: 205px;
        left: -249px;
        font-size: 20px
      }
      .efficacy {

        top: 205px;
        left: -155px;
        font-size: 20px
      }
      .distictiveness {

        top: 205px;
        left: -30px;
        font-size: 20px
      }
      .adaptability {

        top: 205px;
        left: 95px;
        font-size: 20px
      }
      .rivalrousness {

        top: 236px;
        left: -235px;
        font-size: 20px

      }
      .excludability {

        top: 267px;
        left: -238px;
        font-size: 20px

      }
      .marketability {

        top: 301px;
```

```
        left: -238px;
        font-size: 20px

    }
    .redundancy {

        top: 267px;
        left: -120px;
        font-size: 20px

    }
    .robustness {

        top: 267px;
        left: 0px;
        font-size: 20px

    }
    .gravity {

        top: 267px;
        left: 110px;
        font-size: 20px

    }
    .centrality {

        top: 236px;
        left: 105px;
        font-size: 20px

    }
    .substitutability {

        top: 301px;
        left: -60px;
        font-size: 20px

    }
    .connectedness {

        top: 301px;
        left: 85px;
        font-size: 20px

    }
    .social {

        top: 360px;
        left: -115px;
        font-size: 20px

    }
    .political {

        top: 360px;
        left: -20px;
        font-size: 20px

    }
    .economic {
```

```
302.
303.           top: 390px;
304.           left: -15px;
305.           font-size: 20px
306.
307.       }
308.       .human {
309.
310.           top: 390px;
311.           left: -115px;
312.           font-size: 20px
313.
314.       }
315.       .built {
316.
317.           top: 420px;
318.           left: -115px;
319.           font-size: 20px
320.
321.       }
322.       .natural {
323.
324.           top: 420px;
325.           left: -15px;
326.           font-size: 20px
327.
328.       }
329.       .cultural {
330.
331.           top: 450px;
332.           left: -65px;
333.           font-size: 20px
334.
335.       }
336.
337.       .desc{
338.           padding-top:40px;
339.           width:760px;
340.           margin:0px auto
341.       }
```