5-2014

MAPIT - A Mapping Application for Freshwater Invertebrate Taxa

Sirisha Pratha
Utah State University

MAPIT – a Mapping Application for Freshwater Invertebrate Taxa

by

Sirisha Pratha

A report submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

Computer Science

Approved:

Curtis Dyreson Major Professor	Charles Hawkins Committee Member

Xiaojun Qi
Committee Member

UTAH STATE UNIVERSITY
Logan, Utah

2014

ABSTRACT

MAPIT – a Mapping Application for Freshwater Invertebrate Taxa

by

Sirisha Pratha
Utah State University, 2014

Major Professor: Dr. Curtis Dyreson

Department: Computer Science

With the increasing popularity of the World Wide Web among internet users across the world, the need for building web based applications is increasing with time. The Western Center for Monitoring and Assessment of Freshwater Ecosystems (WMC) and the National Aquatic Monitoring Center (NAMC) jointly host a central database containing biological data that is used to assess the condition of aquatic ecosystems. The information stored in the database contains biological and, - geographical data. This information is made available easily through a simple but effective tool called MAPIT. MAPIT is a search engine which can be used to search through the environmental and biological data related to aquatic invertebrates and their locations. MAPIT also produces a map of the location of where the individual taxa were collected. Users can also download the data in a standard format for further analysis.

ACKNOWLEDGMENTS

I would like to thank my major professor, Dr. Curtis Dyreson. His ideas and backing made this me materialize this project. Secondly, perhaps the most important person who helped me from the start of the project is Dr. Chuck Hawkins. His vision for what was an idea discussed in a conference room come to life, played an integral part of the completion of the project. I would like to thank Dr. Qi for her valuable support and suggestions in this project.

I would like to thank my colleagues at WMC lab, which helped with the testing and provided suggestions throughout the project.

Finally, I would like to thank my family back in India especially my father, without whom I could have never achieved anything. A special thanks to my friend Karen who motivated me to finish this project completion and graduation.

Sirisha Pratha

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The Western Center for Monitoring and Assessment of Freshwater Ecosystems (WMC) and the National Aquatic Monitoring Center (NAMC) store environmental and biological data collected from various aquatic ecosystems throughout the western United States of America. Data records contain the following information: source (or project) of data, the year the sample was collected, and the location of the sample (state and geographic coordinates). These data are a valuable resource to natural resource scientists and managers.

These data are housed in a SQL database (Microsoft SQL server 2008) but access to the data was restricted to database managers. To facilitate access to the database and use of the data, a data entry and retrieval desktop application was implemented in ASP.NET on Windows platform. An executable to this application had to be placed on the client’s machine before a client could enter or retrieve data. Even though this application served its primary purpose, it had certain disadvantages. To mention a few, security was a biggest concern with this setup, because the application was an executable. Also from a maintenance and version control perspective, it was difficult to ensure each user was using the latest version of the application after an enhancement or bug fixes were in place. A web-based application that allowed easy data retrieval would be a preferred alternative. Subsequently, a Query Tool was developed to provide this capability and serve a wider audience. This tool overcame many of the disadvantages that a typical desktop application has, namely ease of maintenance, more security and easy deployments. This tool retrieved data aggregated by sample, i.e., it retrieved all of the taxa found in each of one or more samples [1].

Some clients are only interested in certain, specific taxa, and a way to query just those taxa was desired. To achieve this functionality, a web-based tool called *__M__apping __Ap__plication for Freshwater __I__nvertebrate __T__axa (MAPIT)* was built for retrieving data on individual taxa as well as showing collection locations on a Google map. MAPIT is a full-text search engine providing an efficient, faster data retrieval with visual aids and a user friendly UI. Several UI features were incorporated such as mapping the geographical coordinates on a map, *suggest-as-you-type* of any keyword in the database while searching, ability to filter data by several attributes like state, project, date and ability to download the data.

After performance and usability testing, we were able to achieve satisfactory design and performance stability. We sought feedback from several real users to understand the system's efficiency and ease of use. We got mixed feedback and constructive suggestions, some of which were implemented. A majority of beta testers were able to use the system without much training and system behaved smoothly and quickly.

CHAPTER 2

RELATED WORK

2.1 OVERVIEW

This chapter provides a brief overview of the existing work in WMC world and the origin of MAPIT application.

2.2 Existing Applications in CS world

This chapter provides a brief overview of the existing work in WMC world and the origin of MAPIT application. Search engines are of various categories, namely web search or full text search and database search engine.

For any enterprise application that conducts data analysis and reporting, an essential component is a database or data warehouse. A data warehouse stores current and historical data and suite of applications are implemented such that users can mine useful and meaningful data and for reporting purposes [3]. Data warehouse computing has existed for over 30 years now, even before iPhone, twitter and cloud computing. Data warehouses have been monopolizing enterprise computing because of the concept of central data repository- fed by dozens or hundreds of databases, applications and other source systems continue the most efficient way for companies to manage a firm wide view of the clients, operations, reporting, etc.

Some of the growing trends in this decade are technologies such as Hadoop and in-memory processing, helping with “big data” type of data that are suitable for extremely large applications that have as much as 1000 times bigger than before. Open source systems like Hadoop framework that uses distributed framework and Map reduce algorithm, excelling at processing very big data sets.

On the other end of the spectrum is the growing usage of in-memory computations to accelerate the process of the data analysis and reporting on a data warehouse. With the evolution of a single RDMS database to frameworks like Hadoop, we are looking at the new generation of data warehouses that faster, bigger, better than ever before, transforming raw data into information and information into actionable insights, enabling businesses to move forward with unparalleled speed and agility. In short, Data warehouse belonging to any generation have the same purpose of providing a platform for data storage and data analysis. Based on the type of enterprise solution and data sets that are being handled, various data warehousing solutions are implemented.

Apache Lucene is free open source information retrieval software that is capable of full indexing and search capabilities. Its logical architecture is based on the idea of document containing fields of text [4]. Textual information can be extracted from any file format that can be indexed. Many companies have extended the usage of Lucene to implement search engines. It can be extended to various applications like searchable mail, online documentation search, searchable web pages, website search, content search, etc.

Lucene only provides indexing platform and search capabilities but not crawling or parsing capabilities. Several Lucene based projects have extended the capabilities to include web crawling and HTML parsing. Twitter is an example which uses Lucene based search for real time search.

Lucene has been implemented in various languages other than Java. MAPIT was built using Dbsight which is one such extension of Lucene project [2]. Dbsight inherits all the features of Lucene search and provides a crawler capability to perform full text search and UI to present the results. How is Dbsight different from Google/Nutch and other search engines? Dbsight can be used to implement full text search capability on your own database than internet web pages. It's the functionality but using your own data warehouse.

If one were to pit Google, Amazon and Yahoo against each other, each of these engines are better than the other. On one hand we have, Google and Yahoo build their search engines based on implementing web crawlers that efficiently and effectively crawl over the ill-defined Internet, on the other hand we Amazon which has a strikingly different use of the same concept of searching but on its market supplies, vendors, products they sell/buy and ship, etc. In short, Amazon is a search engine based on their products and vendors and suppliers whereas Google/Yahoo is primary search engines for users all over the world. In this contrast, MAPIT is a search engine built on our own data. To name a few similarities that exist with the other e-commerce search engines, one is that it is capable of searching nay keyword in the world on taxonomic data that resides within our database, second is that it is capable of filtering data by various buckets of interest to the user, third is that it is capable of allowing the users to view this data in a MAP like format, and last but the least is that is capable of allowing the users to save the data. MAPIT in all its form enables the users to perform the functionalities a user would desire out of a search engine and more.

CHAPTER 3

SYSTEM ANALYSIS

3.1 OVERVIEW

In order to build any application , one must first begin to understand requirements of the various stakeholders i.e. identify who needs data collected from stream and aquatic ecosystems, to understand why they need it, and to understand various means to access that data and the requirements that the software or system itself has. The best place to start this analysis, it to identify the various objects/systems that MAPIT is composed of and the understanding of the interaction of these components. This analysis will form a solid foundation and bring clarity to the scope of the project, actors, actions accompanying with the system, desired functionality and requirements.

There are several different ways of capturing the requirements and documenting them. In this project, we use *Unified Modeling Language (UML)*, a general-purpose object-oriented modeling language popularly used in the field of software engineering [5].

3.2 Functional Requirements

A functional requirement of an application defines what the application is supposed to do [8]. Here is the requirements specification of MAPIT.

3.2.1 Interface Requirements:

- Search Box shall accept numeric/non-numeric data entry.
- Search box shall accept any type of data entry that exists in the database.
- Search Box shall prompt suggestion of words that are searchable in the database.

- Search option button shall retrieve the results and display in google map in the bottom pane.

- Search option button shall retrieve the results and display in tabular/grid format in the bottom pane below the google map.

- Search option button shall retrieve the results and display in categories with the count of the number of entries in each category in the left pane.

- Search shall begin either by pressing “Search Button” or hitting “Enter”.

- Clicking Search button after choosing “Pins” option shall display the results in the map in “Pins” format.

- Clicking Search button after choosing “Site Clusters” option shall display the results in the map in heat map format.

- Web interface shall have a map to display the various locations where the data was collected.

- Upon clicking on the “pin” on the map, it shall display data associated with the pin in interest in a popup.

- Popup on the map shall be closeable by the user.

- Upon using mouse scroll up and down, map shall zoom in and out.

- Left hand pane shall contain filter options.

- Upon using the Filter By option, number of results shall adjust to the user’s filter setting.

- Upon using the Filter By, number of results in map shall adjust to the user’s filter setting.

- Upon using the Filter By, number of results in table shall adjust to the user's filter setting.

- Table data shall be sortable on certain columns.

- Table data shall be available in the form of comma separated file for users to be able to download.

- Upon clicking on the table data entry, its corresponding entry on the map shall be highlighted and a pop-up with the details shall appear.

- Upon clicking on the table entry, entire row on the table shall be highlighted in yellow.

3.3 Non- Functional Requirements

Since MAPIT was built on top of existing architecture, there are certain predefined constraints or non-functional requirements [9].

- Operating system: MAPIT was developed in windows operating system.
- Language and Platform: MAPIT was developed using Java based language with SQL server programing as the backend database is SQL server database.
- Web Server: MAPIT was developed using Apache HTTP server residing on a Apache Tomcat Server.
- Database System: This was an integral part of MAPIT which was the existing base for this application.

3.4 Use Cases

To understand the high level view of the system, use case diagrams come close to detail the interactions between the users (actors) and the system. We capture these interactions or user objectives in UML diagram called use cases [6]. We first identify the users and clearly list down the user goals.

Some use cases are given here below:

1. Actors: Our primary actor is any MAPIT user, who is interested in data acquired from aquatic and other ecosystem streams. MAPIT will be the system with which this user will interact with.

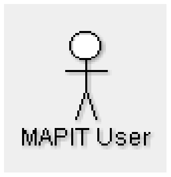


FIGURE 1: MAPIT USER

2. Primary Use cases:

   Primary use cases include searching data, viewing relevant data upon choosing filters and thereby downloading the data.

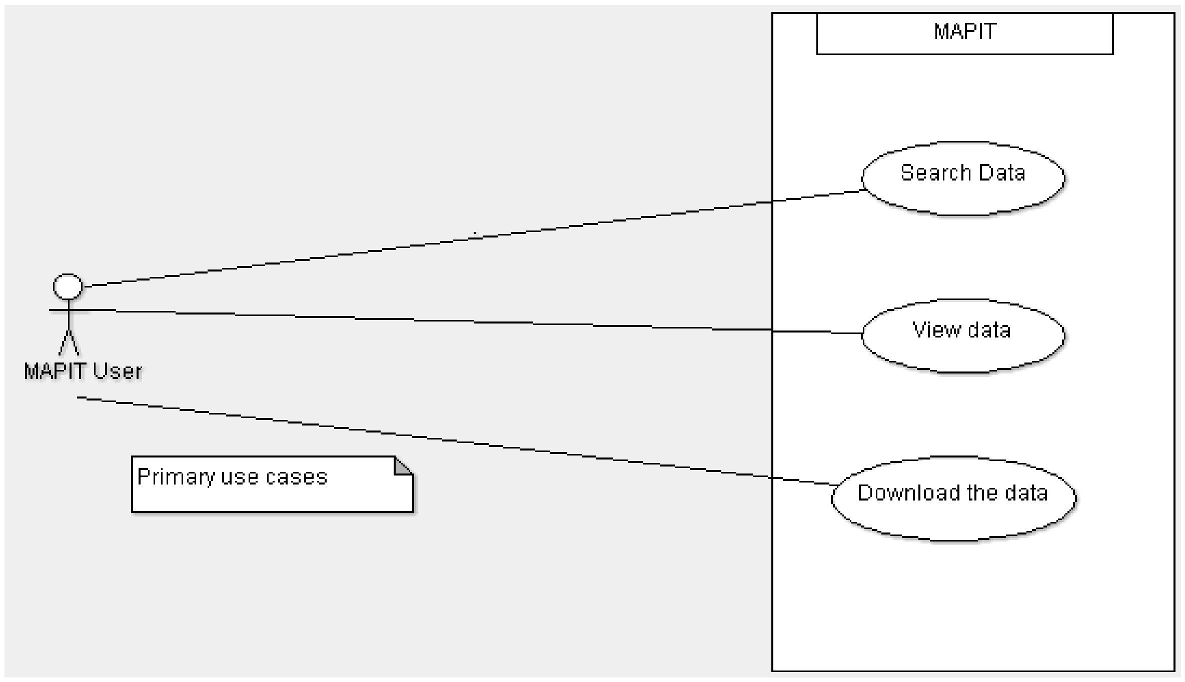


FIGURE 2: PRIMARY USE CASES

3. Search Data: MAPIT user searches data by inputting the taxa name of their choice, MAPIT makes the query and sends it to database system to fetch the data.

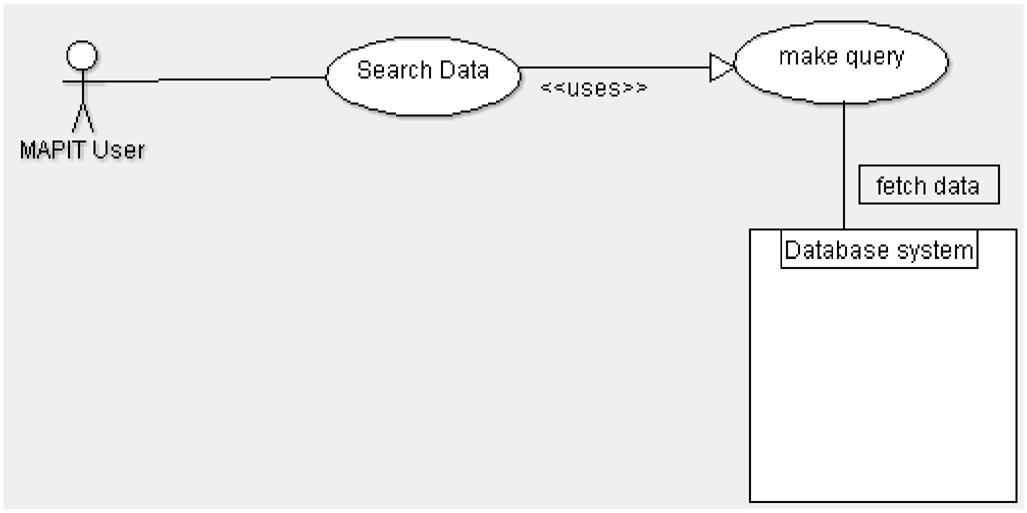


FIGURE 3: SEARCH DATA USE CASE

4. View Data: MAPIT users will be presented with data in multiple formats. User can then filter though the data categories or look at the google map to locate points of interest.

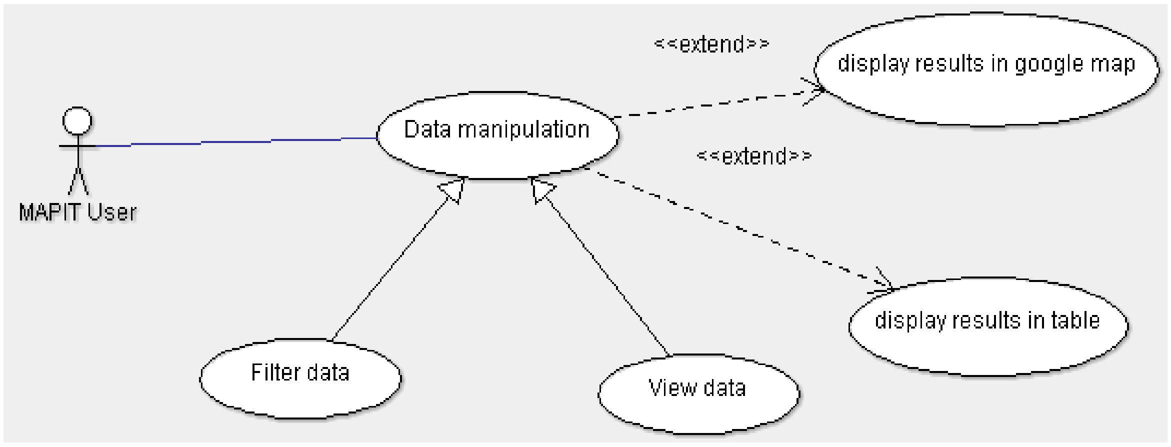


FIGURE 4: VIEW DATA USE CASE

5. Download data: MAPIT users can download only the data of their choice after applying filters. Downloaded file is a comma separated file with columns seen in the table/grid view.

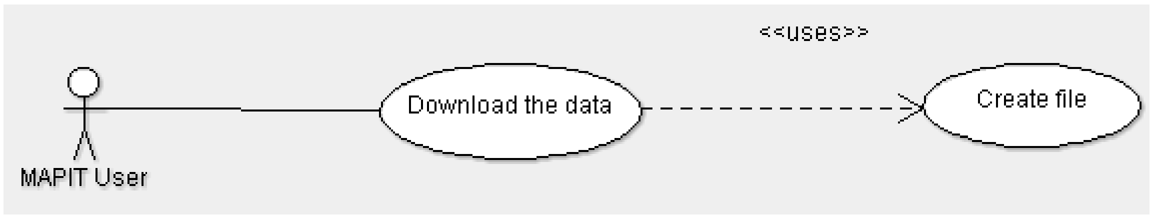


FIGURE 5: DOWNLOAD DATA USE CASE

3.5 Activity Diagram

Activity diagrams represent the workflow of the various activities and actions that are performed by the user [7]. This is basically an illustration of the flows that are the either the structural or computational.

Basic notation of the activity diagram looks very similar to a traditional flow chart. They help represent transactions that are decisions, transitions and parallel sub flows. It can basically represent a set of sequentially performed actions, a set of decisions of decisions that control the workflow based on conditional behavior of the user, a set of iterations that are encountered in a workflow and a set of terminal activities.

Below are some of the activity diagrams that help depict these nuances in the system behavior in a narrative style.

1. This diagram represents the choices made by the user to use MAPIT.

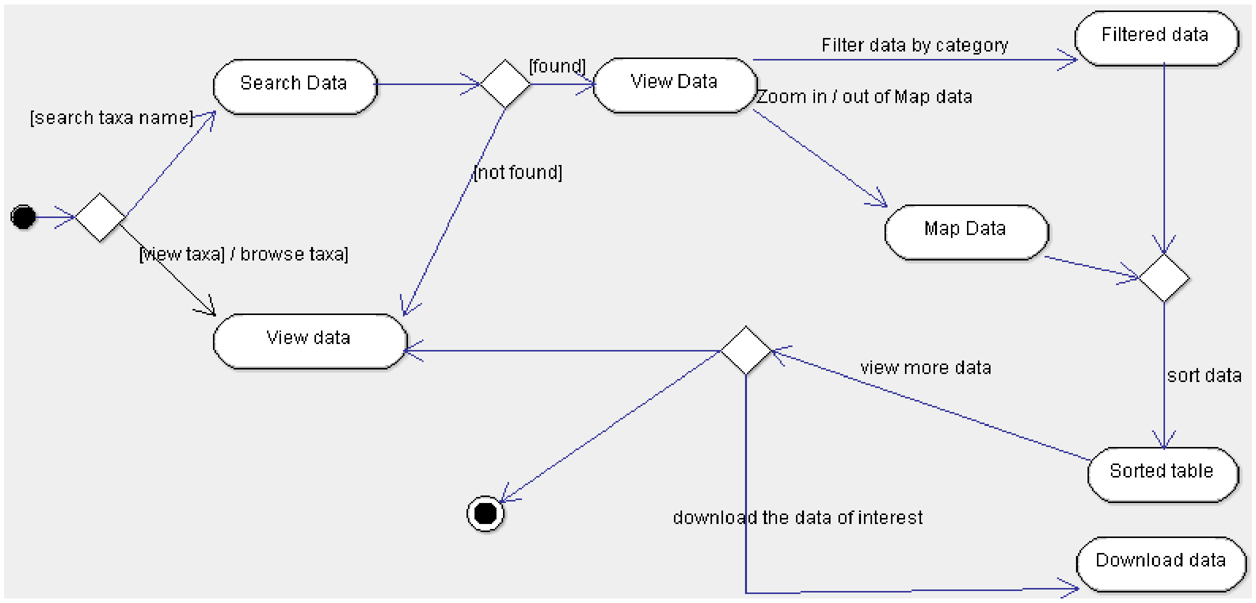


FIGURE 6: ACTIVITY DIAGRAM SHOWING THE INTERACTION BETWEEN THE USER AND THE SYSTEM

2. This below diagram represents the computational workflow of the system.

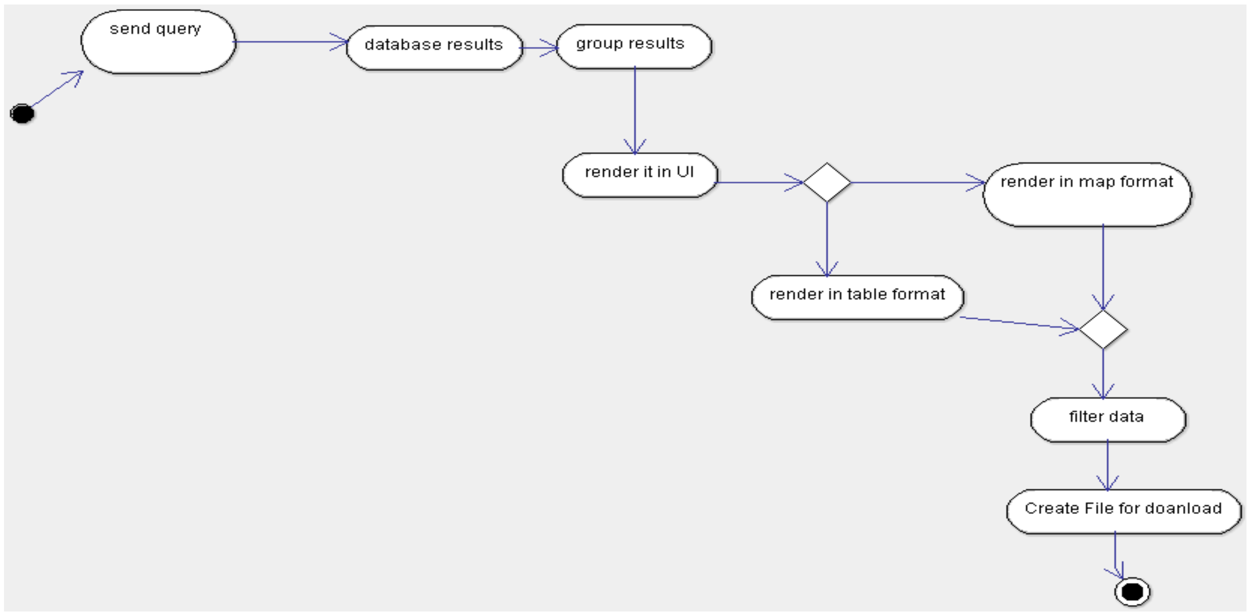


FIGURE 7: ACTIVITY DIAGRAM SHOWING THE INTERACTION WITHIN THE SYSTEM

CHAPTER 4

SYSTEM ARCHITECTURE AND FUNCTIONALITY

With evolution of software and web based applications in particular, applications tend to integrate with other already existing frameworks.

For example, you have a meeting invite in your calendar in Gmail on a particular date and location, most applications leverage that information to map it on a Google map for place and time of the meeting enabling users to quickly grasp the information thus conveyed.

Yet another popularly used search application is Amazon providing users a drill down filtering of the products by various attributes such as size, gender, price, etc.

This tool called MAPIT tool is one such tool that was built with the help of above use cases. This tool will help cater to the needs of this generation's web based GUI with filtering and visual depiction of the places where the data was collected in a map like format. In its simplest form, it serves as a querying tool like written on top of a rich repository of data.

One of the biggest challenges faced in any multi-dimensional database is to extract meaningful and important data that will help the end users make use of this data. A technique popularly used to overcome this challenge is visualization.

To make visualization helpful in this context, we needed to use tools that tightly integrate database queries and the Google map API itself. Also, support interactive refinement of the display, and can visually present a large number of tuples and dimensions.

4.1 System Architecture

In order to achieve the above goals, here is the design of the architecture in place for MAPIT. Client: MAPIT is a web application that has a simple, easy to user interface that allows the users to help query the database. Upon querying the tool, the results are displayed in two formats to the user. First the visual Google Map API of the areas where the data was collected and the second being the old school table format. Subsequent chapters will contain detailed information with the help of pictures.

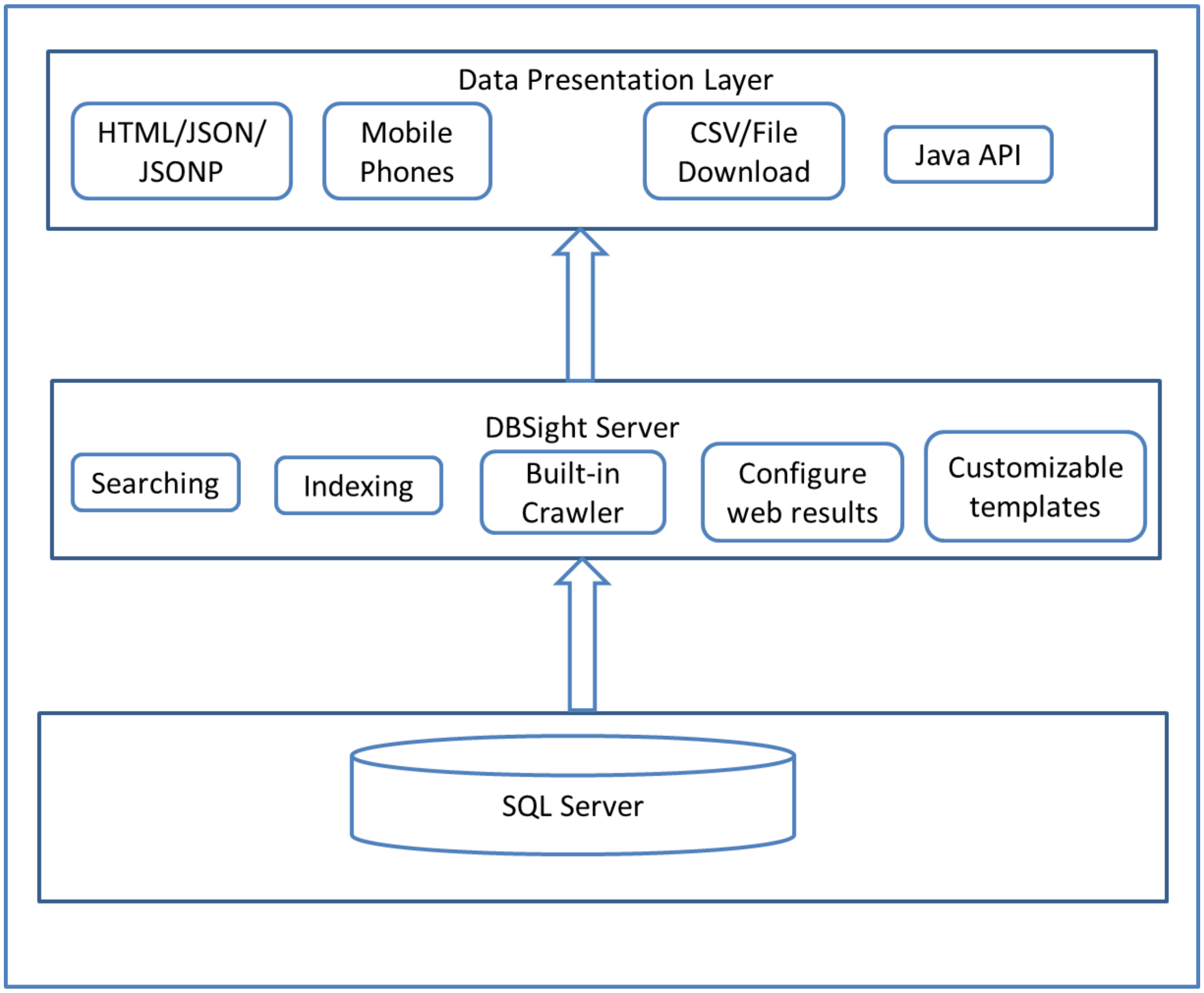


FIGURE 8: ARCHITECTURE DIAGRAM OF MAPIT

Data Presentation Layer:

This is the front end UI to render the results from the database. Two main components that constitute the web UI is Google Map Api and the sortable table to display the results fetched upon the user’s query.

Server: Java server with DBSIGHT application. DBSight is a platform that allows full text search which comes as a deployable war file. This also allows the integration with the sql database server and the web user interface.

Database: The backend infrastructure is a SQL relational database with a couple of hundred tables containing biological and environmental data. The query from MAPIT runs on this database and query results are processed on the server and are presented to the user on the browser.

4.2 Entity Relationship Diagram:

Entity relationship model is a graphical representation of entities and their relationships to each other, used to visualize the relationships of data within databases or information systems. An entity here is an object or concept about which data is stored in the databases. Multiple relationships exist between various entities. Below is a diagram representing the same.

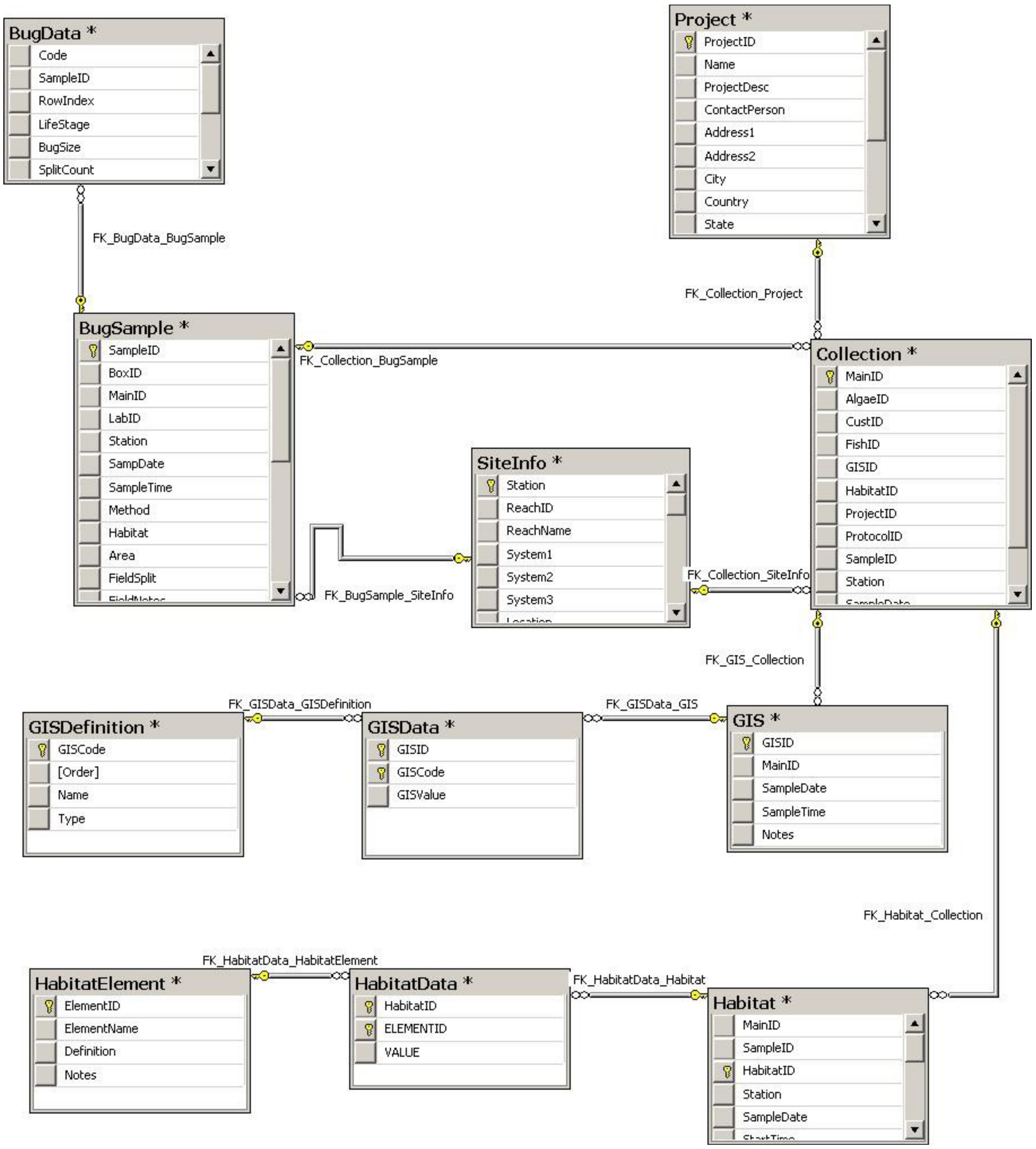


FIGURE 9: ENTITY RELATION DIAGRAM

4.2 Graphical User Interface and UI Patterns

User interface is the most crucial and integral part of this project as it is the first and the only point of contact for the users to interact with to get their data of interest.

Designing a UI is a challenging task to any project as it requires the anticipation of user interactions with the system and provides a layout allowing us as developers and sponsors to tell the users our story.

As mentioned earlier, this is a web application that has a web interface with great accessibility and Google Map API integration. This interface was designed with the help Dr. Charles Hawkins and features added to help improve the overall user experience. In order to achieve these goals several UI patterns were implemented [10].

Things you will see:

4.2.1 Read Me Link:

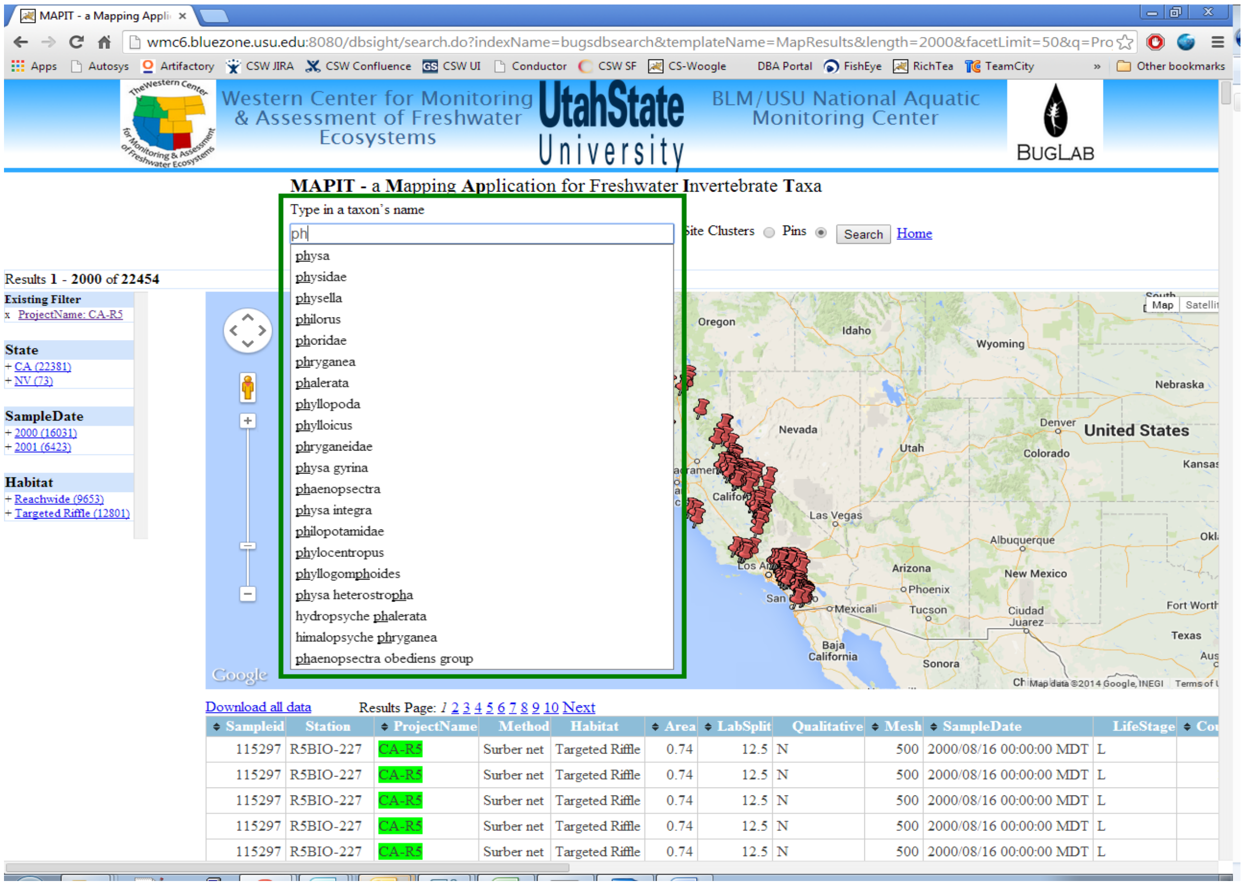
Since this application is one of a kind web tool to represent biological data, it is quite a challenge to show users our perspective of the data representation. To build this image in our users, we created this “Read Me” link before they start querying for data. This document lists features and usage of the tool.

4.2.2 Search Box:

A clear entry point to MAPIT is through this Search Box.

1. When the website is first launched by the user, this Search box will assume an empty query thus returning a cached portion of resultset run across the entire database. To avoid returning large data sets, a limit of 2000 results returned on the table with page view at the bottom.
2. Users will be required to enter their choice of taxonomic group in the search box and hit the Search Button present alongside the search box.

3. Search box also supports auto suggestion feature giving users the list of various taxa names (or other words) in the database that contain the sequence of letters the user started to type.
4. Refining search: Search box also supports Boolean operators and multiple word searches.



MAPIT - a Mapping Application for Freshwater Invertebrate Taxa

Type in a taxon's name

ph

physa
physidae
physella
philorus
phoridae
phryganea
phalerata
phyllopoda
phylloicus
phryganeidae
physa gyrina
phaenopsectra
physa integra
philopotamidae
phylocentropus
phyllogomphoides
physa heterostropha
hydropsyche phalerata
himalopsyche phryganea
phaenopsectra obediens group

Results 1 - 2000 of 22454

Existing Filter
x ProjectName: CA-R5

State
+ CA (22381)
+ NV (73)

SampleDate
+ 2000 (16031)
+ 2001 (6423)

Habitat
+ Reachwide (9653)
+ Targeted Riffle (12801)

Download all data Results Page: 1 2 3 4 5 6 7 8 9 10 Next

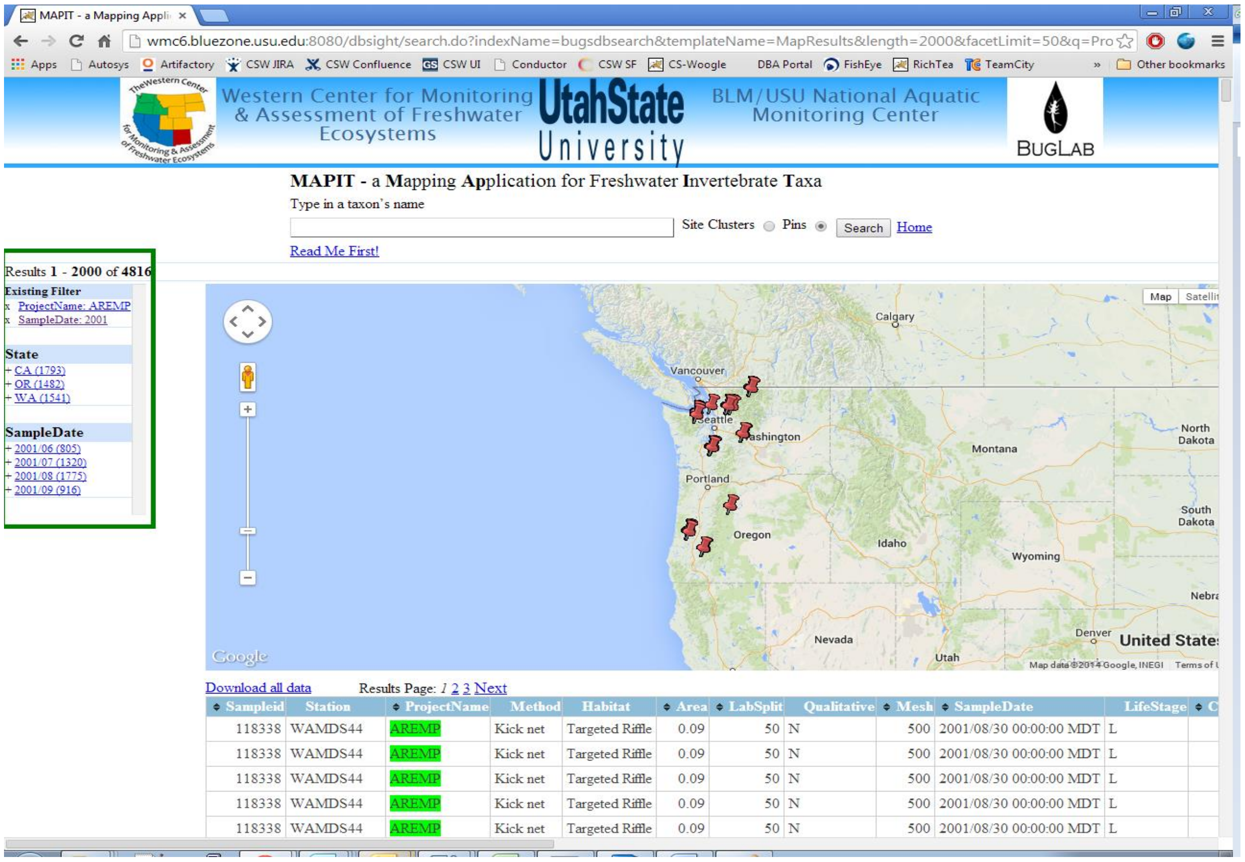
Sampleid	Station	ProjectName	Method	Habitat	Area	LabSplit	Qualitative	Mesh	SampleDate	LifeStage	Co
115297	R5BIO-227	CA-R5	Surber net	Targeted Riffle	0.74	12.5	N	500	2000/08/16 00:00:00 MDT	L	
115297	R5BIO-227	CA-R5	Surber net	Targeted Riffle	0.74	12.5	N	500	2000/08/16 00:00:00 MDT	L	
115297	R5BIO-227	CA-R5	Surber net	Targeted Riffle	0.74	12.5	N	500	2000/08/16 00:00:00 MDT	L	
115297	R5BIO-227	CA-R5	Surber net	Targeted Riffle	0.74	12.5	N	500	2000/08/16 00:00:00 MDT	L	
115297	R5BIO-227	CA-R5	Surber net	Targeted Riffle	0.74	12.5	N	500	2000/08/16 00:00:00 MDT	L	

FIGURE 10: GUI SCREENSHOT SHOWING AUTO-SUGGEST (OR SUGGEST-AS-YOU-TYPE) FEATURE

4.2.3 Filters:

The results from query fired by user are categorized in advance by a preset attribute names, giving the user a "drill-down". This is present on the left side of the page, following the convention of some of the most popular e-commerce websites E.g. Amazon. User can thus narrow the result set by using these filters like Project Name, State, Sample Date, and Method.

Faceted Navigation design pattern was implemented to design this part of the interface. This pattern allows us to leverage the metadata values giving the user options to refine queries. This pattern was used with the aim of improving the user's experience. As the queries get refined and filtered, map results and table results automatically adjust for the user.


MAPIT - a Mapping Appli
wmc6.bluezone.usu.edu:8080/dbsight/search.do?indexName=bugsdbsearch&templateName=MapResults&length=2000&facetLimit=50&q=Pro
Apps Autosys Artifactory CSW JIRA CSW Confluence CSW UI Conductor CSW SF CS-Woogle DBA Portal FishEye RichTea TeamCity Other bookmarks
Western Center for Monitoring & Assessment of Freshwater Ecosystems
UtahState University
BLM/USU National Aquatic Monitoring Center
BugLab
Map
Satellit
Calgary
Vancouver
Seattle
Washington
Portland
Oregon
Montana
Idaho
Wyoming
North Dakota
South Dakota
Nebra
Nevada
Utah
Denver
United State
Google
Map data ©2014 Google, INEGI
Terms of U

MAPIT - a **M**apping **App**lication for Freshwater **I**nvertebrate **Ta**xa

Type in a taxon's name

Site Clusters ○ [✓ Pins] Search Home

Read Me First!

Results **1** - **2000** of **4816**

Existing Filter
x ProjectName: AREMP
x SampleDate: 2001

State
+ CA (1793)
+ OR (1482)
+ WA (1541)

SampleDate
+ 2001/06 (805)
+ 2001/07 (1320)
+ 2001/08 (1775)
+ 2001/09 (916)

Download all data   Results Page: *1* 2 3 Next

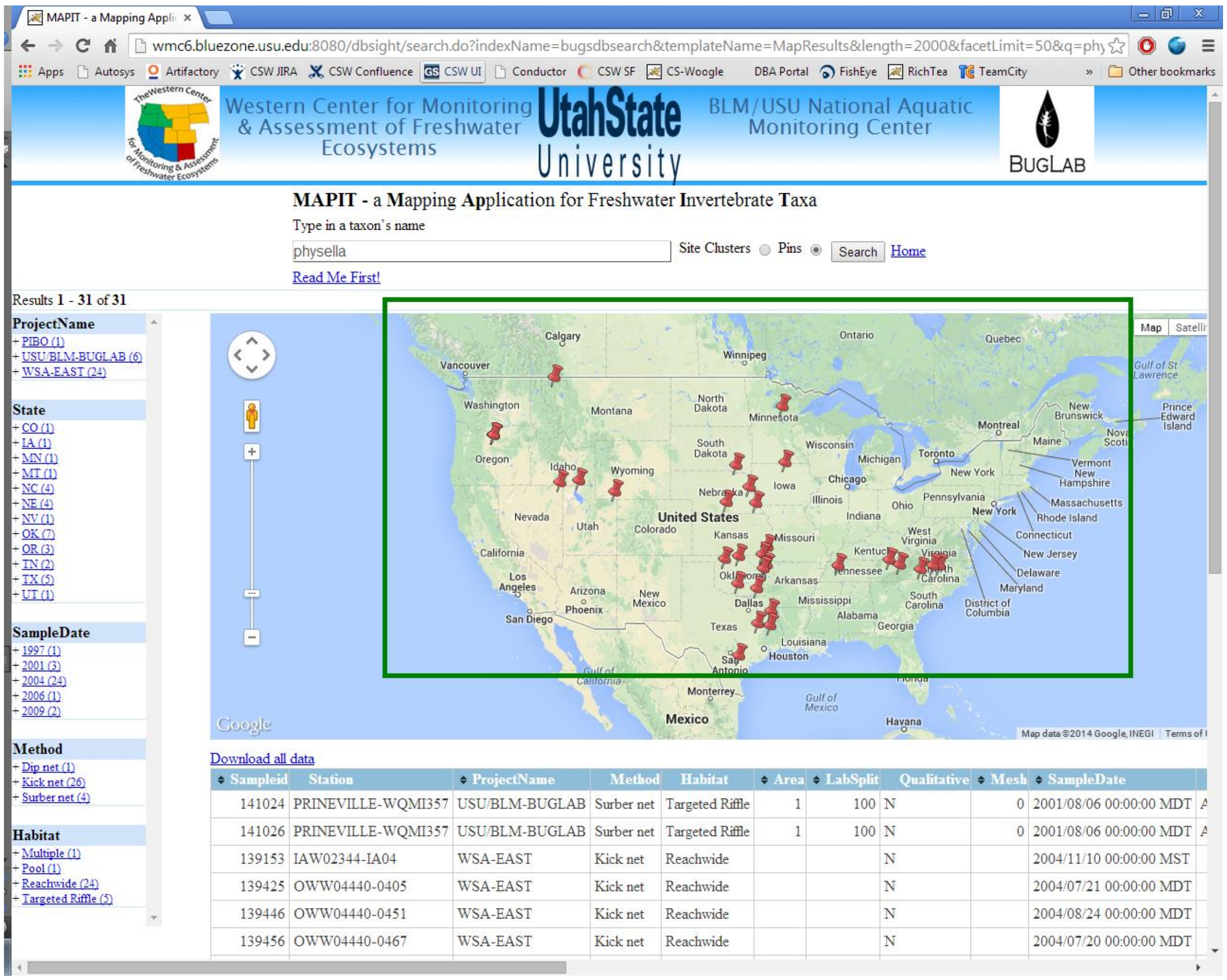
Sampleid	Station	ProjectName	Method	Habitat	Area	LabSplit	Qualitative	Mesh	SampleDate	LifeStage	C
118338	WAMDS44	AREMP	Kick net	Targeted Riffle	0.09	50	N	500	2001/08/30 00:00:00 MDT	L	
118338	WAMDS44	AREMP	Kick net	Targeted Riffle	0.09	50	N	500	2001/08/30 00:00:00 MDT	L	
118338	WAMDS44	AREMP	Kick net	Targeted Riffle	0.09	50	N	500	2001/08/30 00:00:00 MDT	L	
118338	WAMDS44	AREMP	Kick net	Targeted Riffle	0.09	50	N	500	2001/08/30 00:00:00 MDT	L	
118338	WAMDS44	AREMP	Kick net	Targeted Riffle	0.09	50	N	500	2001/08/30 00:00:00 MDT	L	

FIGURE 11: GUI SCREENSHOT SHOWING FILTERS (OR CATEGORIES)

4.2.4 Map API:

Upon querying the database via MAPIT, the results are presented to the user in the map format. This format has two visual representations for mapping options.

1. Pins: It is the standard Google MAP format where the pins represent the location of the areas where the data was collected. This is also the default format when the query is fired. Upon hovering over the pin, site information pops up.


MAPIT - a Mapping Application for Freshwater Invertebrate Taxa

Type in a taxon's name: physella

Site Clusters ○ Pins ●

Results 1 - 31 of 31

ProjectName
+ PIBO (1)
+ USU/BLM-BUGLAB (6)
+ WSA-EAST (24)

State
+ CO (1)
+ IA (1)
+ MN (1)
+ MT (1)
+ NC (4)
+ NE (4)
+ NV (1)
+ OK (7)
+ OR (3)
+ TN (2)
+ TX (5)
+ UT (1)

SampleDate
+ 1997 (1)
+ 2001 (3)
+ 2004 (24)
+ 2006 (1)
+ 2009 (2)

Method
+ Dip net (1)
+ Kick net (26)
+ Surber net (4)

Habitat
+ Multiple (1)
+ Pool (1)
+ Reachwide (24)
+ Targeted Riffle (5)

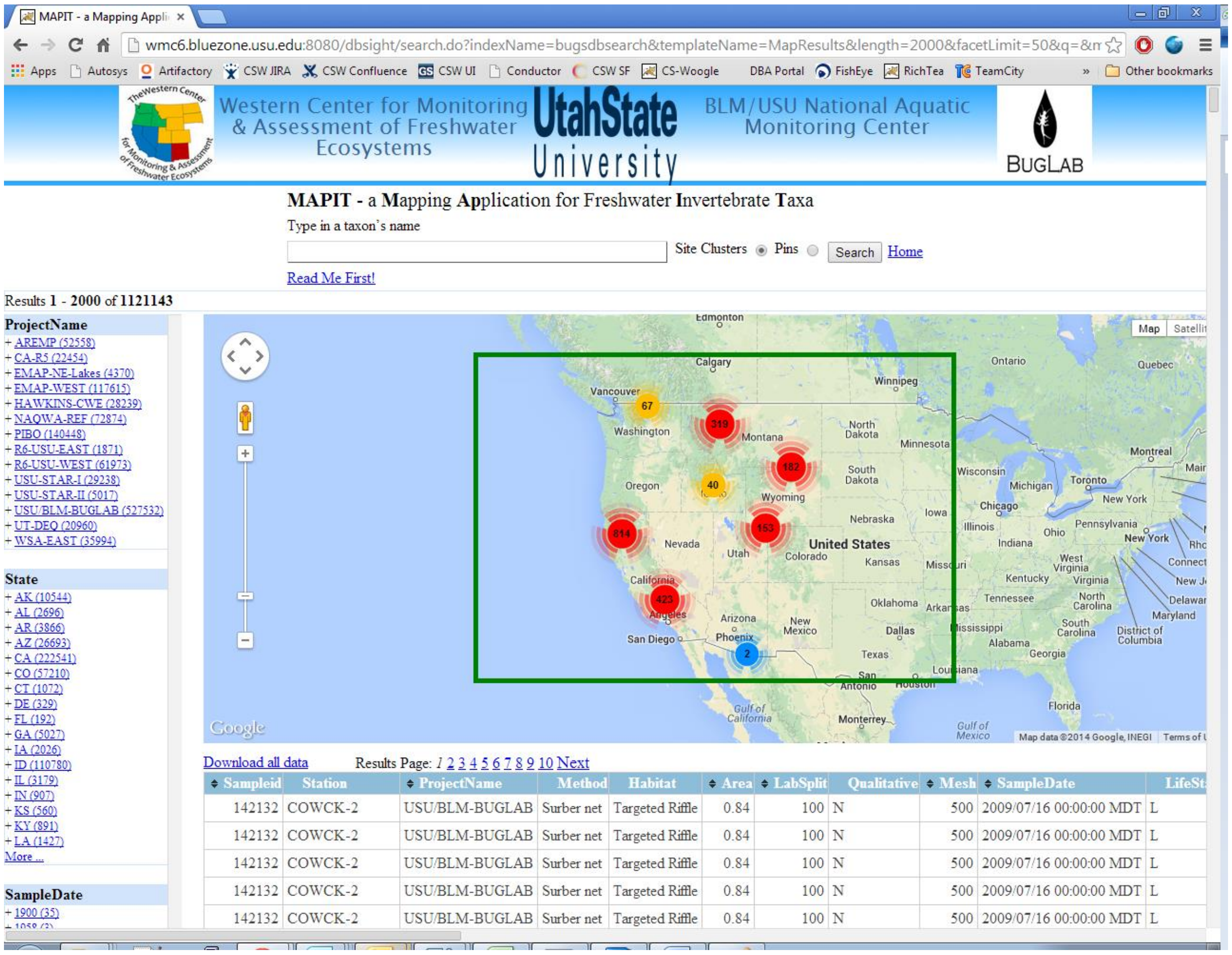
Download all data

Sampleid	Station	ProjectName	Method	Habitat	Area	LabSplit	Qualitative	Mesh	SampleDate
141024	PRINEVILLE-WQMI357	USU/BLM-BUGLAB	Surber net	Targeted Riffle	1	100	N	0	2001/08/06 00:00:00 MDT
141026	PRINEVILLE-WQMI357	USU/BLM-BUGLAB	Surber net	Targeted Riffle	1	100	N	0	2001/08/06 00:00:00 MDT
139153	IAW02344-IA04	WSA-EAST	Kick net	Reachwide			N		2004/11/10 00:00:00 MST
139425	OWW04440-0405	WSA-EAST	Kick net	Reachwide			N		2004/07/21 00:00:00 MDT
139446	OWW04440-0451	WSA-EAST	Kick net	Reachwide			N		2004/08/24 00:00:00 MDT
139456	OWW04440-0467	WSA-EAST	Kick net	Reachwide			N		2004/07/20 00:00:00 MDT

wmc6.bluezone.usu.edu:8080/dbsight/search.do?indexName=bugsdbsearch&templateName=MapResults&length=2000&facetLimit=50&q=phy
Western Center for Monitoring & Assessment of Freshwater Ecosystems
Utah State University
BLM/USU National Aquatic Monitoring Center
BugLab
Search
Home
Read Me First!
Map
Satellite

FIGURE 12: GUI SCREENSHOT SHOWING PINS OPTION ON GOOGLE MAP

2. Site Clusters: This is the second format to represent the same data, but it creates heat map of clusters of sites/areas where the data was collected. By clicking on individual clusters, the resolution of that cluster increases, thus increasing the details on that area. This mapping option performs better than the Pins option, because of data grouping.


MAPIT - a Mapping Appli
wmc6.bluezone.usu.edu:8080/dbsight/search.do?indexName=bugsdbsearch&templateName=MapResults&length=2000&facetLimit=50&q=&m
Apps
Autosys
Artifactory
CSW JIRA
CSW Confluence
CSW UI
Conductor
CSW SF
CS-Woogle
DBA Portal
FishEye
RichTea
TeamCity
Other bookmarks
Western Center for Monitoring & Assessment of Freshwater Ecosystems
UtahState University
BLM/USU National Aquatic Monitoring Center
BugLab
Map
Satelli
Google
Map data ©2014 Google, INEGI
Terms of U

MAPIT - a Mapping Application for Freshwater Invertebrate Taxa

Type in a taxon's name

Site Clusters [✓ ●] Pins ○ Search Home

Read Me First!

Results **1 - 2000** of **1121143**

ProjectName
+ AREMP (52558)
+ CA-R5 (22454)
+ EMAP-NE-Lakes (4370)
+ EMAP-WEST (117615)
+ HAWKINS-CWE (28239)
+ NAQWA-REF (72874)
+ PIBO (140448)
+ R6-USU-EAST (1871)
+ R6-USU-WEST (61973)
+ USU-STAR-I (29238)
+ USU-STAR-II (5017)
+ USU/BLM-BUGLAB (527532)
+ UT-DEQ (20960)
+ WSA-EAST (35994)

State
+ AK (10544)
+ AL (2696)
+ AR (3866)
+ AZ (26693)
+ CA (222541)
+ CO (57210)
+ CT (1072)
+ DE (329)
+ FL (192)
+ GA (5027)
+ IA (2026)
+ ID (110780)
+ IL (3179)
+ IN (907)
+ KS (560)
+ KY (891)
+ LA (1427)

More ...

SampleDate
+ 1900 (35)

Download all data   Results Page: *1* 2 3 4 5 6 7 8 9 10 Next

Sampleid	Station	ProjectName	Method	Habitat	Area	LabSplit	Qualitative	Mesh	SampleDate	LifeSt
142132	COWCK-2	USU/BLM-BUGLAB	Surber net	Targeted Riffle	0.84	100	N	500	2009/07/16 00:00:00 MDT	L
142132	COWCK-2	USU/BLM-BUGLAB	Surber net	Targeted Riffle	0.84	100	N	500	2009/07/16 00:00:00 MDT	L
142132	COWCK-2	USU/BLM-BUGLAB	Surber net	Targeted Riffle	0.84	100	N	500	2009/07/16 00:00:00 MDT	L
142132	COWCK-2	USU/BLM-BUGLAB	Surber net	Targeted Riffle	0.84	100	N	500	2009/07/16 00:00:00 MDT	L
142132	COWCK-2	USU/BLM-BUGLAB	Surber net	Targeted Riffle	0.84	100	N	500	2009/07/16 00:00:00 MDT	L

FIGURE 13: GUI SCREENSHOT SHOWING SITE CLUSTERS OPTION.

4.2.5 Table/Grid Format:

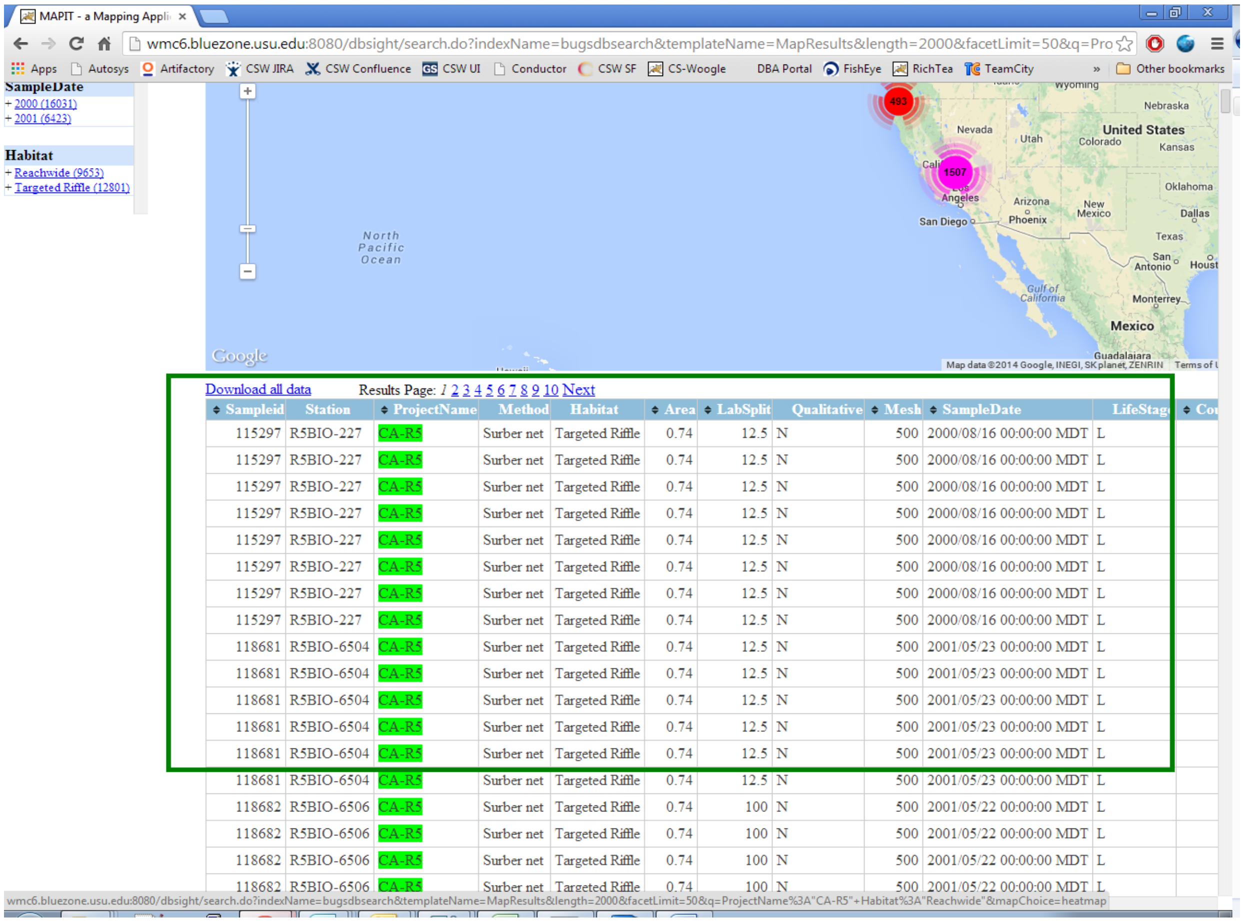
Google Map Integration allows users to visually get an understanding of the data spread, but the table/grid at the bottom was designed for two purposes:

1. To get the full details of the sample collected and post processing information.
2. To access and be able to download the data for further analysis on the users side.

Pagination design pattern was implemented in displaying the contents of the results. This particular pattern was used considering the large datasets that are returned and the challenge to display all of it on a browser page without hurting the performance.

By dividing the large dataset into smaller, manageable data portions for users to read and cope with. Also, it gives the user an idea of the table contents and the relational context that exists in the table. By using this pattern, users can reevaluate their choice of query or use the "narrowby" option to further filter the data of interest.

The other pattern commonly used on tables that was implemented in MAPIT as well is "Sort by Column" pattern. This feature was implemented particularly on certain handpicked columns, which we would provide the user to sort the data of their interest.



Sampleid	Station	ProjectName	Method	Habitat	Area	LabSplit	Qualitative	Mesh	SampleDate	LifeStag	Co
115297	R5BIO-227	CA-R5	Surber net	Targeted Riffle	0.74	12.5	N	500	2000/08/16 00:00:00 MDT	L	
115297	R5BIO-227	CA-R5	Surber net	Targeted Riffle	0.74	12.5	N	500	2000/08/16 00:00:00 MDT	L	
115297	R5BIO-227	CA-R5	Surber net	Targeted Riffle	0.74	12.5	N	500	2000/08/16 00:00:00 MDT	L	
115297	R5BIO-227	CA-R5	Surber net	Targeted Riffle	0.74	12.5	N	500	2000/08/16 00:00:00 MDT	L	
115297	R5BIO-227	CA-R5	Surber net	Targeted Riffle	0.74	12.5	N	500	2000/08/16 00:00:00 MDT	L	
115297	R5BIO-227	CA-R5	Surber net	Targeted Riffle	0.74	12.5	N	500	2000/08/16 00:00:00 MDT	L	
115297	R5BIO-227	CA-R5	Surber net	Targeted Riffle	0.74	12.5	N	500	2000/08/16 00:00:00 MDT	L	
115297	R5BIO-227	CA-R5	Surber net	Targeted Riffle	0.74	12.5	N	500	2000/08/16 00:00:00 MDT	L	
118681	R5BIO-6504	CA-R5	Surber net	Targeted Riffle	0.74	12.5	N	500	2001/05/23 00:00:00 MDT	L	
118681	R5BIO-6504	CA-R5	Surber net	Targeted Riffle	0.74	12.5	N	500	2001/05/23 00:00:00 MDT	L	
118681	R5BIO-6504	CA-R5	Surber net	Targeted Riffle	0.74	12.5	N	500	2001/05/23 00:00:00 MDT	L	
118681	R5BIO-6504	CA-R5	Surber net	Targeted Riffle	0.74	12.5	N	500	2001/05/23 00:00:00 MDT	L	
118681	R5BIO-6504	CA-R5	Surber net	Targeted Riffle	0.74	12.5	N	500	2001/05/23 00:00:00 MDT	L	
118681	R5BIO-6504	CA-R5	Surber net	Targeted Riffle	0.74	12.5	N	500	2001/05/23 00:00:00 MDT	L	
118682	R5BIO-6506	CA-R5	Surber net	Targeted Riffle	0.74	100	N	500	2001/05/22 00:00:00 MDT	L	
118682	R5BIO-6506	CA-R5	Surber net	Targeted Riffle	0.74	100	N	500	2001/05/22 00:00:00 MDT	L	
118682	R5BIO-6506	CA-R5	Surber net	Targeted Riffle	0.74	100	N	500	2001/05/22 00:00:00 MDT	L	
118682	R5BIO-6506	CA-R5	Surber net	Targeted Riffle	0.74	100	N	500	2001/05/22 00:00:00 MDT	L	

wmc6.bluezone.usu.edu:8080/dbsight/search.do?indexName=bugsdbsearch&templateName=MapResults&length=2000&facetLimit=50&q=ProjectName%3A"CA-R5"+Habitat%3A"Reachwide"&mapChoice=heatmap

FIGURE 14: GUI SCREENSHOT SHOWING THE TABLE/GRID FORMAT

4.2.6 Download the data:

The data thus filtered is easily available in comma separated flat file which can be downloaded by clicking on the link "Download all data". Sample data format for the same is.

CHAPTER 5

TESTING

Software testing is systematic methodology to validate the work done by the computer program. It also serves as a means to test the quality of the application or the program.

In the recent past, software testing has gained more than ever importance. Many industries also encourage the practice of writing tests before actually writing code also known as Test Driven Development [11].

Testing can range from as basic as unit tests to more complex system testing and user acceptance testing. Here in MAPIT project two type of testing were mainly used.

5.1 System Testing

This type of a test is mainly performed when various systems (client/server architecture) interact with each other and there is data flowing in and out of these systems. System testing is the best form of testing to identify any leaks or breaks in the system flow [12]. From the previous chapter, it is clear that there is more than one system interacting in MAPIT architecture. How did MAPIT benefit with this test?

1. System testing helped in testing end to end flow of MAPIT and correct bugs with each iteration of this test.
2. System testing helped bring clarity to what the system should behave like and what is capable of.
3. System testing help understand potential bugs in other parts of the system, that will break MAPIT. Example: java server down, database server down.

4. System testing helped discover some real time scenarios with multi-user capability and such.

5.2 Usability Testing

Usability testing is typically performed when application is User Interface Centric application. It is real time test that is evaluated and verified by the users usually done before going live with the product or service [13].

There are several different methods are employed to achieve the same and is typically done in a “draft phase” of the project .When we started out this project, it was mainly one user: Dr. Chuck Hawkins and who was also the user to contribute to the business requirements. There was no better place to start usability testing, but with Dr. Chuck. With several iterations of this test, there was considerable amount of UI improvements that were implemented. For example, Read Me link, Logos, Table columns etc.

While several changes and modifications were implemented and System testing was performed several times to reach a satisfactory performance from all aspects, we were ready to perform Hallway Testing. As interesting as the name sounds, so was the process of performing it.

This is a test done to involve users who were not contributing to the requirements, but random users who are asked to test the product or service. From the definition quoted here, it literally means testes that are passing by in the hallway. In theory, we should reach out to random users. In practice, it would have been hard to approach random users in the hallway, because the subject matter in MAPIT requires an expert understanding of our type of specialized data.

To bring a variation to the above mentioned process of testing, Dr. Chuck reached out to several of his colleagues who were not aware of this MAPIT tool existence or even the fact that it was being developed. We achieved "random" in this context. We received several appreciations and constructive feedback. Several of those were incorporated into the project and some were not due to deadlines and ad to not allow scope creep in requirements.

5.3 Performance Testing

In software engineering, performance testing is a general testing technique to determine the system's stability and responsiveness under a particular workload [14]. It is the process of assessing the effectiveness of the system, program or application usually measured by several quantitative tests. For web application such as MAPIT, good measures of determining the system's speed and efficiency are data transfer, bandwidth or throughput.

Below are some of the graphs to represent the same:

Tests were performed by simulating 20 virtual users to access the website for the largest data set retrieval on this database.

1) Number of pages per second, successful hits per second and number of active users for a period of 10 minutes.

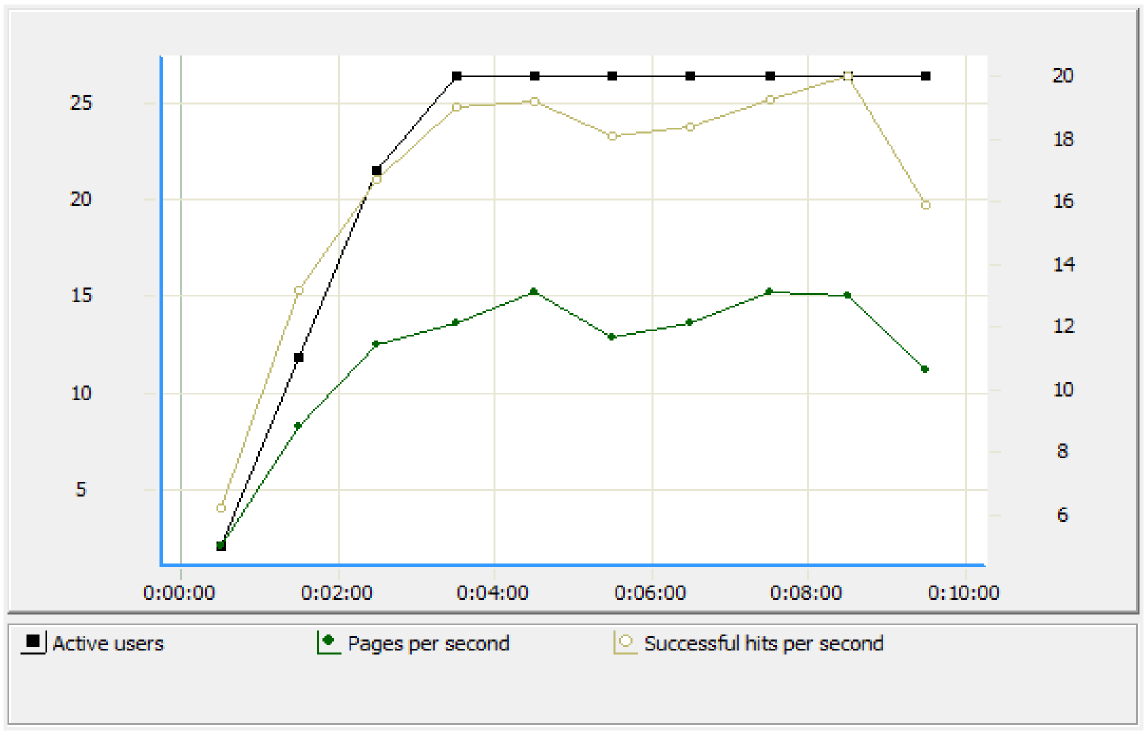


FIGURE 15: GRAPH SHOWING NUMBER OF PAGES PER SECOND AND SUCCESSFUL HIT PER SECOND FOR 20 ACTIVE USERS FOR 10 MINUTES.

2) Average response time, Average 90% response time and Number of Active users for a period of 10 minutes.

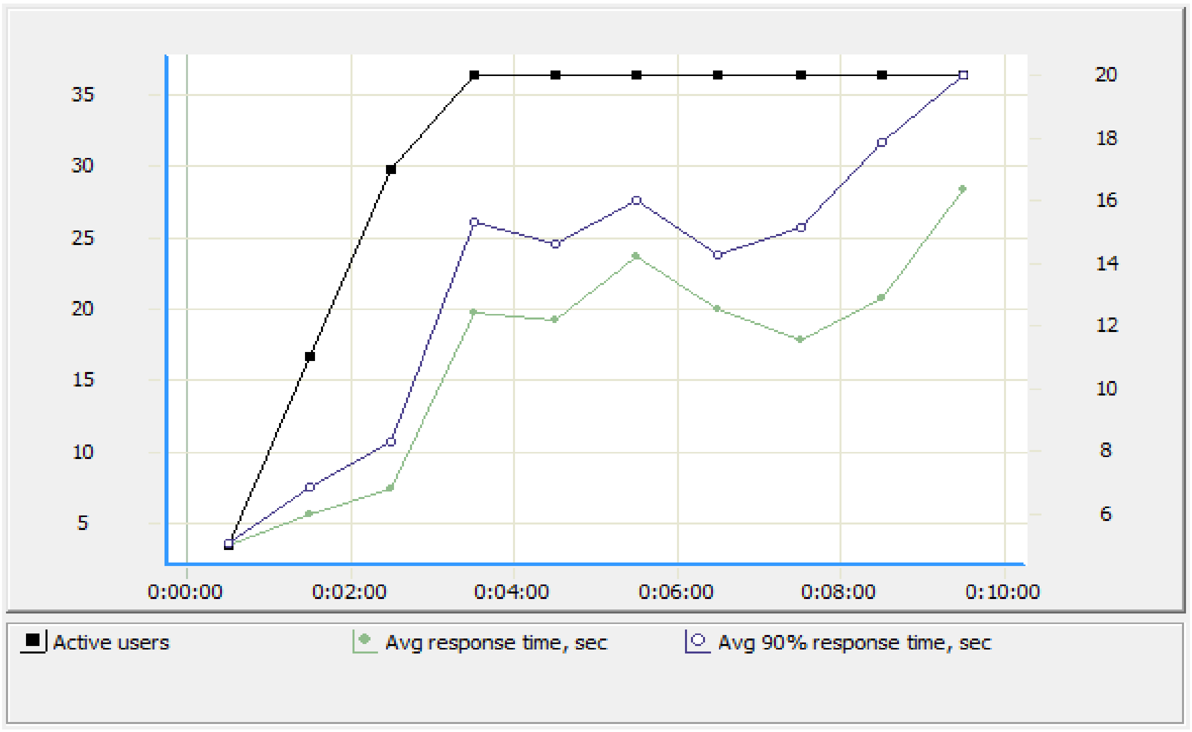


FIGURE 16: GRAPH SHOWING AVERGAGE RESPONSE TIME, 90% REPONSE TIME FOR 20 ACTIVE USERS FOR 10 MINUTES.

3) HTTP Errors, Total number of errors and number of active users for a period of 10 minutes.

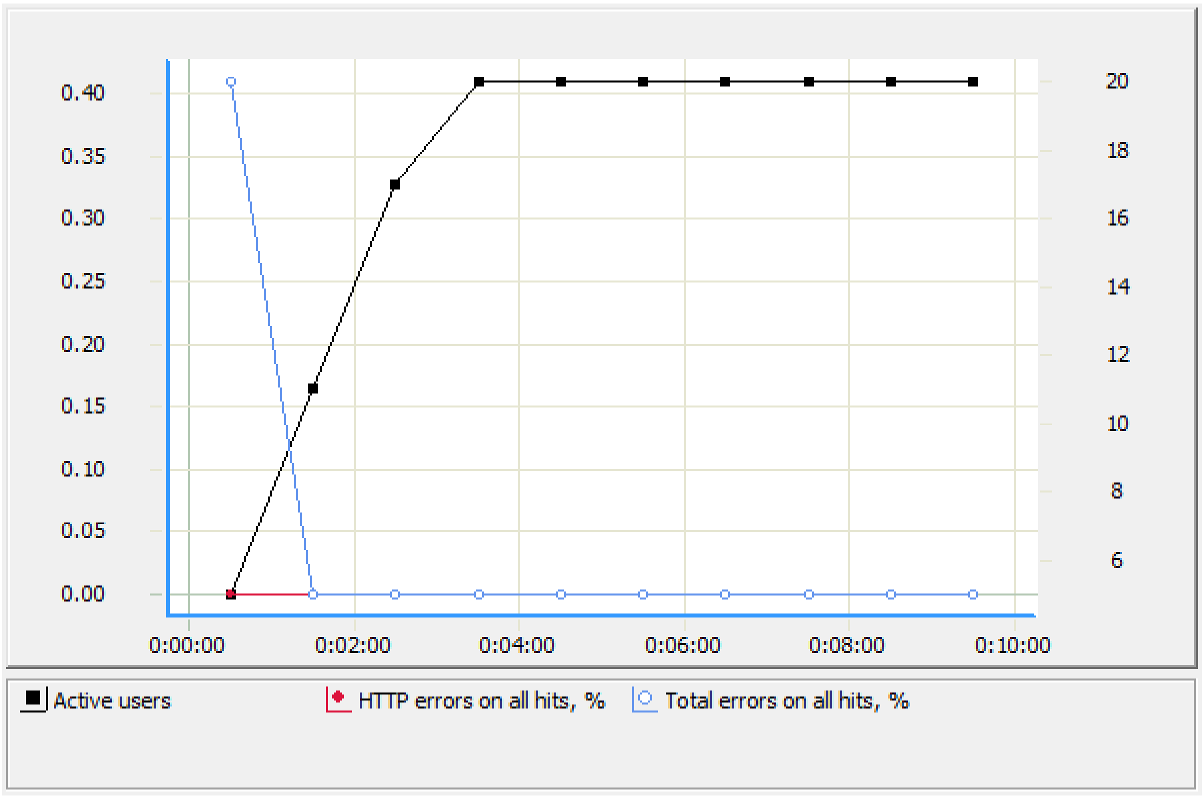


FIGURE 17: GRAPH SHOWING PERCENTAGE OF HTTP ERRORS AND ALL ERRORS FOR 20 ACTIVE USERS FOR 10 MINUTES.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

MAPIT serves as basic yet powerful search engine to easily and effectively access the environmental and biological data and also to save the data of interest in a readable format. This tool avoids the need for any manual, ad-hoc based data requests to the DBA maintaining the database. Since this data in on a web platform, independent application, data collected is available and used even more by many users. Multiple users can access the data at the same time without any performance impact, thus making it even more effective.

6.2 Future work

At present, data is available to anybody who accesses the MAPIT tool. Even though we added security enhancements to flag what data is viewable and not, may be “Login” option for authorized users is a better approach. This would require us to create an authentication layer and prevent any data leakages.

Another simple enhancement would be to check spelling errors and suggest an alternative correct spelling. This is already a feature provided by Dbsight, it would be an enhancement to the project.

REFERENCES

[1] WMC USU, http://www.cnr.usu.edu/wmc/

[2] Dbsight Homepage http://www.dbsight.net/index.php?q=

[3] Data warehouse http://en.wikipedia.org/wiki/Data_warehouse

[4] Lucene API, https://lucene.apache.org/

[5] UML Resource Page by Official Modeling Group (OMG), http://www.uml.org/

[6] Use Case Diagrams, Wikipedia, http://en.wikipedia.org/wiki/Use_case_diagram

[7] Activity Diagrams, Wikipedia, http://en.wikipedia.org/wiki/Activity_diagram

[8] Functional Requirements, Wikipedia, http://en.wikipedia.org/wiki/Functional_requirements

[9] Non-Functional Requirements, Wikipedia,http://en.wikipedia.org/wiki/Non-functional_requirement

[10] UI design patterns, http://ui-patterns.com/

[11] Software Testing, Wikipedia, http://en.wikipedia.org/wiki/Software_testing

[12] System Testing, Wikipedia, http://en.wikipedia.org/wiki/System_testing

[13] Usability Testing, Wikipedia, http://en.wikipedia.org/wiki/Usability_testing

[14] Performance Testing, Wikipedia, http://en.wikipedia.org/wiki/Software_performance_testing