



Vernal Pool Database Project

A Major Qualifying Project

Submitted to the Faculty

Of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Bachelor of Science

By

Niva Shrestha

April 24, 2008

1. Database
2. Vernal pool
3. Ruby on Rails

Approved: _____

Professor Murali Mani

Professor Stanley Selkow

Abstract

Due to the rapid loss of vernal pools, and the insufficient information shared in public, a medium to encourage participation and to enhance collaboration of the vernal pool community has become very important. In order to create such a medium, this project created a web application with the back-end database by using Ruby on Rails framework and MySQL. It also utilized Ajax technology to create an interactive environment for the users. The accomplishment of this project is great service to the vernal pool community. The created application allows the community to contribute and to share their resources, and lets the participants to search and to learn more about vernal pools.

Acknowledgements

We would like to thank several people for being part of this project directly or indirectly. First of all, thanks to our sponsor Dr Betsy Colburn from Harvard Forest, our advisors Professor Murali Mani and Professor Stanly Selkow for their continuous support and guidance throughout the project. We also wish to thank Professor Gary Pollice for introducing Ruby on Rails and for the opportunity to work on the project as a part of the Web ware project. We would also like to thank student Adwait Belsare for being part of the Web ware Project. Thanks to Michael Voorhis for his help in hosting the site.

Table of Contents

Abstract.....	2
Acknowledgements	3
Table of Figures	6
1.0 Introduction.....	7
2.0 Background	9
2.1 Harvard Forest	9
2.2 Vernal Pool	9
2.2.1 Factors Behind Pool Disappearance	10
2.2.2 Vernal Pool Dependent Animals and Plants	11
2.2.3 Involved Organizations	12
2.2.4 Rules and Regulation	12
2.3 Currently Existing Databases.....	13
2.3.1 NHESP Database	13
2.3.2 Upper Susquehanna Coalition Database	13
2.3.3 Vernal Pool Database in 1993.....	14
3.0 Languages and Tools	15
3.1 Database.....	15
3.1.1 Database Software Requirements	15
3.1.2 Database Software Choice	15
3.1.3 MySQL	15
3.2 User Interface.....	16
3.2.1 Web Application Framework Requirements.....	16
3.2.2 Web Application Framework Choice	17
3.2.3 Ruby on Rails.....	17
3.4 Tools	20
3.4.1 Net Beans	20
3.4.2 Subversion.....	20
3.4.3 Sourceforge	20
4.0 Methodology	21
4.1 Process	22
4.1.1 Iterative Development.....	22
4.2 Design Principles	22
4.2.1 Software Design Principles.....	22
4.2.2 HCI Design Principles	23

5.0 Design and Implementation	24
5.1 Database.....	24
5.1.1 Entity-Relationship Diagram/Schema.....	24
5.1.2 Relational Tables	25
5.2 Web Application.....	25
5.2.1 Application Layout	25
5.2.2 Controllers.....	25
5.2.3 Models.....	26
5.2.4 Views	26
5.2.5 MVC in Action	27
5.3 Highlighted Features from Data Entry.....	28
5.3.1 Filling Form	28
5.3.2 Bulk Upload.....	30
6.0 System in Action.....	31
6.1 Home Page.....	31
6.2 Data Entry	31
6.2.1 Filling Form	32
6.2.2 Existing Pool.....	33
6.2.3 Auto-complete.....	33
6.2.4 Query-options	34
6.2.5 Error Feedback.....	35
6.3 Bulk Upload.....	35
6.4 Google Map	36
7.0 Accomplishments	37
8.0 Future Work and Recommendation	38
8.1 Database.....	38
8.2 User Interface.....	38
8.3 Useful Topics	40
9.0 Conclusion	41
Appendix A: Final E-R Diagram	42
Appendix B: Schema of 1993 vernal pool database	43
Bibliography	44

Table of Figures

Figure 1: MVC in Ruby on Rails.....	18
Figure 2: Entity-Relationship Diagram.....	24
Figure 3: Controllers.....	25
Figure 4: Models.....	26
Figure 5: View Files Sample.....	26
Figure 6: MVC in Action.....	27
Figure 7: Bulk Upload Design.....	30
Figure 8: Home Page.....	31
Figure 9: Enter New Pool.....	32
Figure 10: Find Geographic Coordinates.....	32
Figure 11: Upload an Image.....	33
Figure 12: Data Entry Form.....	33
Figure 13: Auto-complete Feature.....	34
Figure 14: Query-option.....	34
Figure 15: Error Feedback.....	35
Figure 16: Bulk Upload.....	35
Figure 17: Bulk Upload Error.....	36
Figure 18: Google Map.....	36

1.0 Introduction

The ecological importance of vernal pools has been mostly unrecognized until very recently. A vernal pool is a temporary pool of fresh water formed during the spring season from snowmelts or spring rains. Although it is short lived in nature, it plays a vital role in the ecosystem. However, vernal pools are disappearing very quickly due to human development. Since there is an increase in awareness of the importance of vernal pools, in the last decade there has been a growing interest to protect this natural habitat.

There are very few resources available in the public on vernal pools as of now. There is also a lack of knowledge on the animals and plants that are associated with the vernal pools. There are very few databases from governments and private resources about vernal pools and its wildlife. Either these databases do not include detailed information about vernal pools and its wildlife, or the information is inadequate.

In the last decade, researchers have been involved in collecting data on vernal pools through various field studies in the Northeast region, especially in New England. There has been an increase in the number of participants, from students to researchers, who are interested in collecting data of vernal pools. The availability of this collected information is currently restricted to the individuals or small groups participating in the study of vernal pools.

In order to raise more awareness of the importance of the vernal pool, there is a need for all the participants to work together in their research. The best way to accomplish this would be to share collected information or data from participants' field studies.

Therefore, researchers at the Harvard Forest and vernal pool community throughout the Northeast are interested in a system, into which school groups, consultants, regulators, naturalists, and scientists can put together and share data on vernal pools that they observe. As mentioned earlier, due to the inadequacy of information that is currently stored in the public domain, the new system should contain detailed information on the vernal pools and its wildlife.

To achieve this goal, the project requirements have two main components: a database and a user interface. Along with Dr. Betsy Colburn from Harvard Forrest, we created a database into which vernal pool community can contribute and share their data on vernal pools. The project also implemented a web application where the community can easily search and enter their information.

Together, the components of this system will provide the community with an interactive environment where it can be used as a repository, a learning tool, and as a source of information on vernal pool and its wildlife. The team also offered recommendations to extend this system and to enable a future implementation. It will be a great addition to the already rich vernal pool community resource and will provide the people of the community with a means to contribute and to participate in the ongoing study of vernal pools.

2.0 Background

2.1 Harvard Forest

Harvard Forest, a program belonging to Harvard University, was established in 1907. The center is dedicated to “research and education in forest biology and conservation” (Major Research Topics , 2006). Based in New England, it has research facilities and a museum. Scientists, students, and collaborators at the center explore many topics ranging from conservation and environmental change to land-use history (Research and Education in Ecology, Conservation and Forest Biology , 2006).

The center has two major research programs: investigation of natural ecosystems in New England and studies of physical and biological processes relevant to climate change (Major Research Topics , 2006). Besides these two large programs, the center has many other research projects including study on vernal pools.

2.2 Vernal Pool

A vernal pool is a seasonal pool, typically filled with water in autumn or winter due to rising ground water and rainfall (Vernal Pools, 2008). It only lasts for a short period through the spring and typically dries by midsummer or the end of summer. The name, vernal, means, “spring” in Latin. Although the pool is small and temporary, its unique environment supports many animals and plants including some of the rare and endangered species.

Vernal pool has become an essential and safe habitat for certain species of frogs, salamanders, shrimp, insects, and plants. The reason behind this is that the predator of their eggs or offspring, fish, cannot survive in vernal pools because the vernal pools eventually dry up.

These animals and plants have adopted their life style to this unique environment of the vernal pool. Some of the animals lay their eggs and pass their early stages of life in vernal pools, while plants and insects produce cysts and seeds. When the pool dries up near autumn, the moist soil that was on the bottom of the pool now protects the offspring. Once completed with their early stages of life, some of the animals move away into other nearby environments such as woods and wetlands. Adult amphibians at this stage will go into hibernation and wait for the next rainy season. All animals that were associated with the vernal pool come back when it fills up with water in the next season (Vernal Pools, 2002).

Vernal pools are disappearing rapidly due to human development or natural changes. This causes animals and plants that rely on this habitat to struggle for their survival. There are more concerns about amphibians in general since they have adopted a pattern of life in these pools. “Many vernal pool amphibians go back to breed in the pools where they were born. If the pool is disturbed or destroyed by development, the amphibians show little tendency to relocate” (Vernal Pools, 2007). This specific behavior is one example that shows how the disappearance of vernal pool may lead to the extinction of pool-dependent animals and plants.

The loss of vernal pools and the decline in the numbers of its dependant animals could also affect other animals that are higher on the food chain. For example, animals such as snakes, turtles, birds, and small mammalian predators depend on the above species for food. Some endangered turtles such as the wood turtle and the spotted turtle rely on eating eggs and larvae from the vernal pools to survive.

2.2.1 Factors Behind Pool Disappearance

Vernal pools can be found in a variety of places, such as small woodland depressions, meadows, river floodplains, and large vegetated wetland complexes. There are many factors that can affect the creation of a vernal pool. Landscape and hydrologic characteristics are two major factors. Evan H. Grant has created a model that shows that the probability of forming a potential vernal pool is negatively impacted by human

interaction such as cropland, urban/commercial and high-density residential development in the landscape (Grant, 2005).

The Izaak Walton League of America, an organization that is dedicated to protecting America’s hunting, fishing, and outdoor heritage, states “more than 90% of California’s vernal pools have already been lost” (Izaak Walton League of America Fact Sheet, 2007). Large numbers of vernal pools also have disappeared from the Northeast region, although there are no exact statistics on how many pools were lost in total. Massachusetts Fisheries and Wildlife shows that the Commonwealth has four thousands three hundred and seventy two (4,372) recorded vernal pools in 2007 (Number of Certified Vernal Pools by Town, 2007).

2.2.2 Vernal Pool Dependent Animals and Plants

Animals that live in or around vernal pools can be grouped into two categories. The ones that depend on vernal pools to complete their life cycle are called *obligate species*. If vernal pools in the area disappear, these animals may face extinction. The four obligate species in Massachusetts are fairy shrimp, wood frogs, eastern spadefoot toad, and mole salamanders. The other category includes species that somewhat depend on vernal pools but can also live in other water bodies, and are called *facultative species*.

Common Name	Scientific Name
Obligate Species	
Wood frog	<i>Rana sylvatica</i>
Spotted salamander	<i>Ambystoma maculatum</i>
Eastern spadefoot toad	<i>Scaphiopus holbrooki</i>
Fairy shrimp	<i>Anostraca Eubranchipus</i>
Facultative Species	
Gray tree frog	<i>Hyla versicolor</i>
Four-toed salamander	<i>Hemidactylium scutatum</i>
Spotted turtle	<i>Clemmys guttata</i>

Wood turtle	<i>Chrysemys insculpta</i>
Water scorpion	<i>Nepidae</i>
Dragonfly larvae	<i>Odonata Anisoptera</i>
Leeches	<i>Hirundinea</i>
Freshwater (fingernail) clams	<i>Pisidiidae</i>

Table 1.1 – Examples of species relying on the vernal pool (MacCallum, 2001)

There are many species of plants that live around vernal pools. Some of them are native only to a certain vernal pool. According to a study done by Jennifer Ramp on vernal pools of California, there are more than 60 plant species, as well as dozens of invertebrates, that are exclusively native to a certain vernal pool (Ramp, 2004). Disappearance of one vernal pool may lead to the extinction of these unique plants.

2.2.3 Involved Organizations

The decrease in the number of vernal pools and its dependant wildlife has become a genuine interest and a concern for environmentalists, scientists and the government. Many states now have laws to certify and protect vernal pools. There are many organizations that are making efforts to educate people and enforce the laws on vernal pools. Some of these organizations include the U.S Environmental Protection Agency (EPA), the Natural Heritage and Endangered Species Program (NHESP) and the Vernal Pool Association.

2.2.4 Rules and Regulation

Besides these organizations, many states now are collecting data on these pools and have adopted laws to protect these temporary but important habitats. For example, in 1997, Massachusetts passed the Wetlands Protection Act to protect wetlands, including vernal pools.

To protect a vernal pool under this law, Massachusetts’s citizens should first provide the following documentation:

- 1) The Observation Form completed and signed,

- 2) Maps that precisely locate the vernal pool, and
 - 3) Evidence for the existence of the vernal pool and indicator species
- (MacCallum, 2001)

After these documentations are verified, the law will protect the pool. Massachusetts Government will fine any activity that endangers or destroys the vernal pool. “There have been two violations against vernal pools between January 2007 and August 2007. Not only were the violators fined but also their activities were made known to the general public” (Public Participation & News, 2007).

2.3 Currently Existing Databases

There are a few existing vernal pool databases that belong to the government agencies and private organizations. Although every database has its own objectives, these databases all lack interactive community involvement compared to our database.

2.3.1 NHESP Database

Massachusetts Division of Fisheries and Wildlife has established a Natural Heritage & Endangered Species (NHESP) Program that is mainly responsible for the conservation and protection of animal and plant species in Massachusetts. One of the primary functions of the NHESP is to give certification of vernal pools based on specific documentation on the pool (Natural Heritage and Endangered Species Program 2008). Once a vernal pool has been certified, it will be added to the NHESP database. The NHESP has certified a total of 4,372 vernal pools in Massachusetts as of April 2007 (Number of Certified Vernal Pools By Town, 2007). It provides a feature to view the pools in a table and a map sorted by different towns of Massachusetts. However, other informations on these pools are not available to the public.

2.3.2 Upper Susquehanna Coalition Database

Upper Susquehanna Coalition, a private organization, has an integrated data management system. It collects and maintains all wetland data, including data on vernal pools. This

information includes geographic data and the types of animals found. However, this database is also used only for internal purposes.

2.3.3 Vernal Pool Database in 1993

In 1993, two students from WPI created a vernal pool database for Massachusetts Audubon Society as a part of their IQP Project. This database was created using dBase IV software, a popular software during the 1980s. However, it is a hybrid relational and navigational database that is rarely used today.

In comparison with the above-mentioned databases, this database has more detailed information on the pool and animals and plants that rely on the pool. However, similar to above mentioned databases, this was also used for an internal purpose for Massachusetts Audubon Society.

3.0 Languages and Tools

Choosing appropriate tools is very important for any software project. The project team looked into different database software, programming languages, web technologies, web application framework and other tools. The team decided to use the following tools listed below.

3.1 Database

3.1.1 Database Software Requirements

The team created a list of important factors required for choosing the right database software. These factors included pricing, multiple user access, advanced search capabilities, compatibility, portability and ease of learning.

3.1.2 Database Software Choice

After researching current available and time tested database software, our project team chose Oracle and MySQL. Based on the important factors mentioned above, Oracle and MySQL were systematically compared. Oracle and MySQL are both very successful RDBMS software and they both implement SQL. We knew that the price of the software was one of the first priorities for our sponsor agency. Since Oracle is more expensive and MySQL is free, the project team came to the conclusion that MySQL would be the best database software for our project.

3.1.3 MySQL

MySQL is an efficient and widely known open source database system that implements SQL. Its consistent fast performance, high reliability and ease of use are the reasons why individual web developers and many of the world's largest and fastest-growing organizations including industry leaders such as Yahoo, Google, Nokia, and YouTube use it and depend on it (Why MySQL, 2008). One of the popular open source web architectures, LAMP (Linux, Apache, MySQL, PHP / Perl / Python) also uses MySQL. MySQL runs on many commonly used operating systems such as Linux, AIX, HP-UX,

Windows and Netware. It supports foreign keys, joins, views, triggers, and stored procedures. It has data types ranging from numeric, string, date, time and binary and it has API for many programming languages such as C, C++, Python, Perl, TCL etc.

3.2 User Interface

As discussed earlier, we needed to provide database access in the public domain. Therefore creating a web application became our best choice.

3.2.1 Web Application Framework Requirements

The current generation of web technology is dynamic and interactive. The collaboration between users and an interactive environment for the users are the main requirements for our project. We looked into the current web technology described below that allows an effective interactive web application.

3.2.1.1 AJAX

Asynchronous JavaScript and XML (AJAX)'s goal provides web-based applications with responsiveness closer to that of desktop application. It uses a simple idea but the result is a great enhancement for the Web experience. In a traditional model, the client browser creates a HTTP request to the server by clicking on the link or button, or submitting a form. The client then waits until it receives a response or a new document from the server. The new document will be displayed on the entire browser (Sebesta, 2007). This traditional model can be disruptive and time consuming when the client requests complicated documents. In addition, for interactive web pages, uploading new documents for very simple actions might be performance intensive.

In an Ajax web application, the communication from the browser to the server is asynchronous, so the browser does not need to wait for the server to respond. The user can continue to interact with the page while the server finds and transmits the requested document and the browser renders the new document (Sebesta, 2007).

These fundamental differences from the traditional request cycle enable Ajax applications to be significantly more responsive. The web applications become closer in performance to desktop applications and at the same time have all the benefits of being hosted (Raymond, 2006).

3.2.2 Web Application Framework Choice

The WPI web server only supports Perl CGI and HTML. The project team used these languages to create the entire web application for the first few weeks. Then, Professor Gary Pollice introduced “Ruby on Rails” in a Web ware class. He gave a demo on using this framework to create a web application. We were impressed by the framework’s adaptation of MVC architecture, ease of use, along with many other features described below. The framework also uses JavaScript framework “Prototype” to support Ajax and JavaScript model of the document. Therefore, we decided to choose Ruby on Rails as a tool for creating our web application.

3.2.3 Ruby on Rails

Ruby on Rails, commonly known as Rails, is a web development framework that uses MVC architecture to create web applications with database access. David Heinemeier Hansson developed rails in early the 2000s and it was released in July 2004. (Sebesta, 2007)

Rails is implemented using Ruby, an object oriented scripting language released in 1995. Despite some performance issues with Ruby, we decided to use Rails because of its support for JavaScript, AJAX and the following features listed below.

3.2.3.1 Model View Controller (MVC)

MVC is a software architecture that organizes the application into three parts according to their responsibilities: data, presentation, and control logic (Fitzgerald, 2007). Model is the data logic; it represents the application’s data. It is also responsible for communicating with the back end database. View is the application’s presentation logic;

it deals with the user interface by generating HTML, JavaScript code etc. Controller is the control logic; it decides the actions for the user's input and orchestrates interaction between the model and the view (Raymond, 2006).

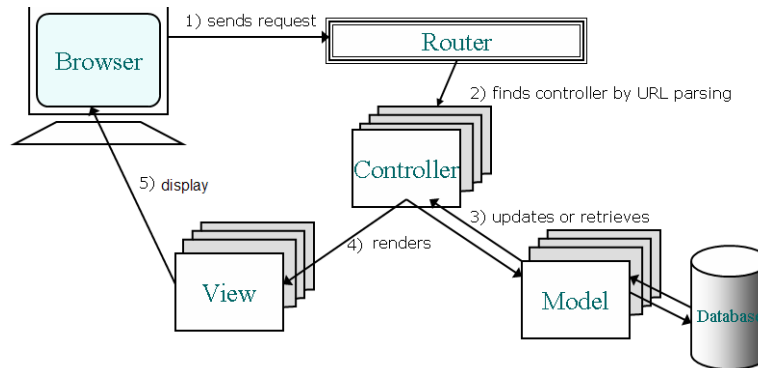


Figure 1: MVC in Ruby on Rails

Rails has multiple models, views and controllers. The router takes a browser request and parses the URL to redirect an incoming request to the corresponding controller and its actions. The controller may retrieve or update the data of the model and renders the corresponding view. The view get send to the browser for display.

The ActiveRecord module of Rails handles the responsibilities of a model. The ActionPack library has two components, ActionView and ActionController that manage view and controller parts of MVC (Lenz, 2007).

3.2.3.2 Convention over Configuration

Many frameworks require configuration files or code for mapping between URLs and methods, and between model attributes and database columns, etc. The motto of Ruby on Rails “convention over configuration” is that whenever possible, an explicit configuration is replaced by sensible defaults (Raymond, 2006).

For example, Rails uses a simple convention for URL routing: the names of a controller (a class name), action (a method name) and primary key (ID). These are used in a

consistent manner in URL. However, it is possible to override the convention. (Fitzgerald, 2007)

3.2.3.3 Object Relational Mapping

A significant characteristic of Rails is its object relational mapping (ORM) approach to connect object-oriented software with a relational database. Using ORM, Rails maps database tables to a class, row to class object, and column to object fields (Sebesta, 2007). For example, if the database contains a table called “animals”, it will be mapped to a class called “Animals”. In the “animals” table, objects of “Animals” class will represent each row and fields of this object will represent each column.

3.2.3.4 Useful Features

Automatic Conversion

The model objects are converted easily from code in HTML document and vice versa.

Automated Test Suites

Rails has three environments: development, production, and test. The test environment offers an easy solution to create test suite to verify the integrity of data model and the correctness of the web pages (Hartl & Prochazka, 2007).

Plug-in

A Rails plug-in is either an extension or a modification of the core framework. There is a large Rails community that is involved in creating plug-ins to add continuously new features in Rails.

Some of the plug-in we used for this project are:

- Auto Complete – A “Magical” Ajax goodies that can display the choices for specific field for the user by querying the database.
- Ym4R GM - This plug-in helps using the Google Maps API from Rails applications.

3.4 Tools

3.4.1 Net Beans

Net Beans is a free, open source Integrated Development environment written in Java. It is mainly used for writing, compiling, testing, and debugging desktop and web applications. It supports full-featured text editor, syntax highlighting, error checking, visual design tools, Ant support, and version control system support. Net Beans also provides support for different programming languages including Ruby and Rails (NetBeans).

3.4.2 Subversion

Subversion, also known as SVN, is a version control system. Its main function is to maintain current and historical versions of source codes. SVN is well known in the open source community and is used in many projects.

3.4.3 Sourceforge

Sourceforge is a great tool for teams to develop the software efficiently. It provides a way to communicate between the team, repository of the documents, releasing code build etc. It also has support for two version control system: SVN and CVS.

4.0 Methodology

To begin the process of creating a system for the vernal pool community, we had two primary objectives to explore: a database and a user interface. We had a weekly meeting with our advisors and our sponsor to fully understand their vision of this system. These meetings were focused on examining the purpose of the project, gathering requirements for the project, and evaluating the completed features of the system.

The meetings revolved around the following questions:

- What type of information a vernal pool community collects?
- What data will be useful to store as a resource?
- How can we facilitate the wide and diverse community?

The project first focused on understanding the useful resources for the vernal pool community. We also studied the schema of the database on vernal pools from 1993. We then designed and created a new database. This database continued to evolve throughout our project span.

The next focus of the project was creating an interactive environment for the community. The requirements were carefully analyzed and converted to user stories. A user story has an associated priority set by the sponsor and a time estimation to complete the implementation. User stories that are related were put in a same category. Due to the limited amount of time, the team chose a category with the highest priority set by the sponsor. A prototype was created for the user interface and other design decisions were made.

Implemented user stories were reviewed and evaluated during weekly meetings. New requirements, ideas, enhancements, or usability concerns were continuously added to the user stories. The database evolved continuously due to minor changes that were made for improvements or enhancements. Although there was no formal testing, each feature was tested by interacting with the system and checking for validations. At the end of the project, the site was hosted temporarily for usability testing and collecting surveys from

the vernal pool community. Because of the time constraint, we did not receive all the surveys on time for this report.

4.1 Process

For any software projects, it is crucial to choose a software development process to ensure the delivery of quality end product. The process of creating a software application requires multiple stages: analysis, design, implementation, and test. There are few different approaches on this topic and we choose the iterative development.

4.1.1 Iterative Development

An iterative development is an important process for developing software. It approaches all four stages: analysis, design, implementation, and test in multiple iterations. The team followed this principle very closely. Sourceforge was very helpful in organizing, estimating, assigning, and tracking the task through iteration.

4.2 Design Principles

Once the majority of the requirements for the project were gathered in the beginning, the team looked into a set of design principles established by professionals over many years. These principles were used as a valuable guidance for our software projects.

4.2.1 Software Design Principles

The team created user stories as a part of gathering requirements. We also created use cases in order to get a better understanding of the interaction between the user and the system.

A user story is a requirement defined from a customer's perspective as things that the system needs to do (User Stories, 1999). It can be very helpful to estimate the cost, time, and effort that need to be invested. As mentioned earlier, the requirements gathered from the weekly meetings were converted to user stories. For implementation, they were chosen based on the priority and time constraints.

A use case is a sequence of actions that provide something of measurable value to an actor. An actor is a person, organization, or external system that plays a role in one or more interactions with your system (UML 2 Use Case Diagrams, 2006). Use cases were created to observe how different members of the vernal pool community might interact with different features of the system.

4.2.2 HCI Design Principles

HCI principles are especially useful guidelines for creating a simple but highly effective user interface. Some of the principles we considered include consistency, error prevention, error handling, and good user feedback.

Consistency is one of the well-known design principles for user interface and it can be expressed through many different forms such as sequence of actions, layout, color, font, menus, etc. The team kept the user interface as consistent as possible through use of colors of a similar family, aesthetic look and feel, sequence of actions etc.

Ideally, users should be prevented from entering any invalid information. If the user makes any mistakes, they should be provided with a clear and informative instruction on how to recover from this error (Dix, Finlay, Abowd, & Beale, 2004). Therefore, the user interface has implemented easy selection options such as radio buttons and drop down menus with default choices. There is also an auto-complete feature that provides hints for the users on specific fields. Users are also provided with sample queries that can grab information from the database and display for selection.

In addition, good response and appropriate feedback are very important. For every data entry, the interface provides appropriate feedback if the user action is successful or if the user made any errors. The error messages help the users handle or prevent this mistake.

5.0 Design and Implementation

5.1 Database

The schema of the vernal pool database created by the WPI IQP team in 1993 was used as a reference to learn important attributes for a vernal pool. The previous database was not a relational database but a hybrid navigational database. Therefore, requirements for our new relational database schema were different.

For convenience, Microsoft Excel was used to get feedback from our sponsor. Appropriate changes were made to the file. After careful consideration of the type of data, entities, and constraints, an E-R diagram was designed.

5.1.1 Entity-Relationship Diagram/Schema

An E-R diagram is a very useful visual diagram to convey a conceptual data model. Each square is a table. Each diamond describes a relationship between tables. The numbers on either side of the diagram signifies the quantitative relationship between tables. This diagram also had undergone continuous modification and improvement through each weekly meeting. For details, please refer to Appendix A.

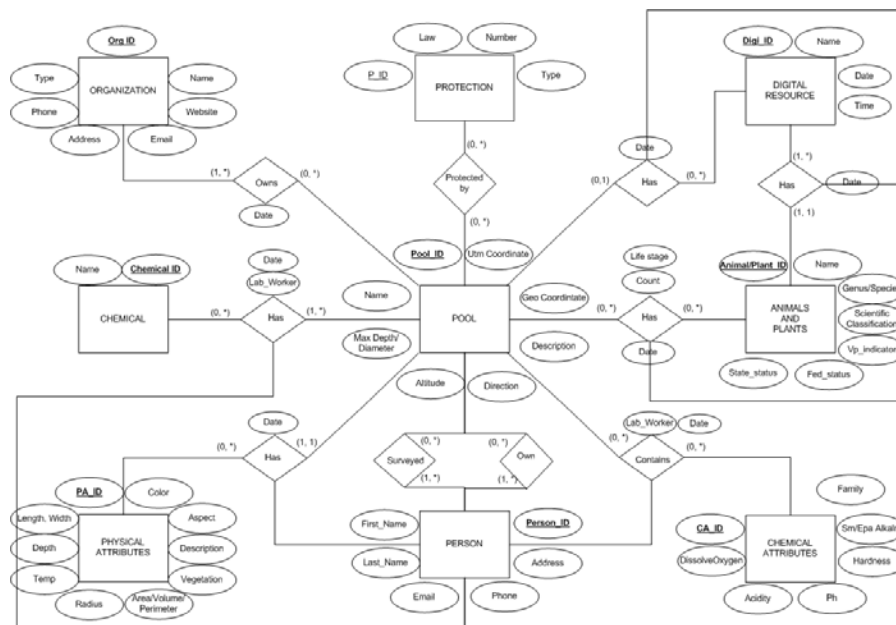


Figure 2: Entity-Relationship Diagram

5.1.2 Relational Tables

Once the E-R diagram was finalized, it was then converted to relational tables. There were altogether 10 data tables and 15 relational tables. The database was created using MySQL.

5.2 Web Application

As mentioned earlier, the team focused on the category with the highest priority: Data Entry.

5.2.1 Application Layout

Our web application is organized in the MVC pattern, provided by Ruby on Rails. MVC architecture organizes the application logic into three parts: data, presentation, and control (Fitzgerald, 2007). Two components of ActiveSupport library, ActionView and ActionController correspond to the view and controller parts of MVC. ActiveRecord module of Rails is the model of MVC.

5.2.2 Controllers

A controller is the control logic of MVC. In Rails, it is a ruby class extending from the ActionController::Base class. It is responsible for handling user's request, creating, updating or retrieving data in the model and rendering the corresponding view. The figure below is a summary of major controller classes used.

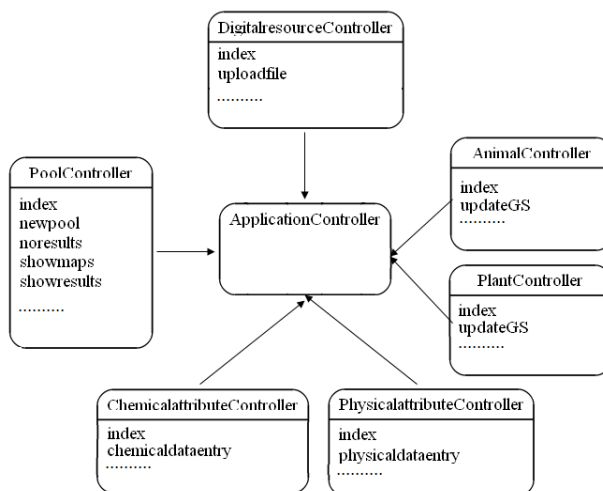


Figure 3: Controllers

5.2.3 Models

Models are the data logic of MVC. Rails uses object relational mapping (ORM) approach to map relational database to Ruby objects. Using ORM, Rails maps database tables to a class, row to class object, and column to object fields (Sebesta, 2007).

First, we need to connect to our database by modifying the 'database.yml' configuration file. Then, model classes are created to map each tables of our database. A model class is a Ruby class that extends from the ActiveRecord::Base class of ActiveRecord module. The Figure below is a summary of mapping of models and tables.

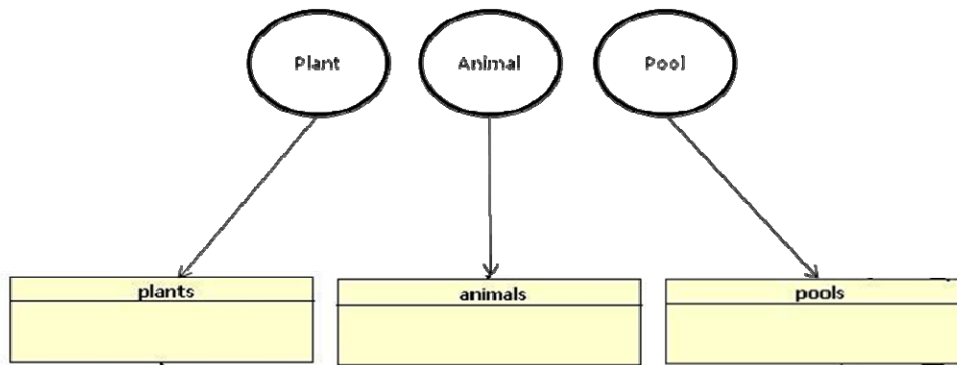


Figure 4: Models

5.2.4 Views

A view is a file with “.rhtml” extension; it consists of HTML code with the possibility of embedded Ruby code to generate dynamic web pages. Each method of a controller class may have a corresponding view file. By convention, a method name maps to the name of the view file.

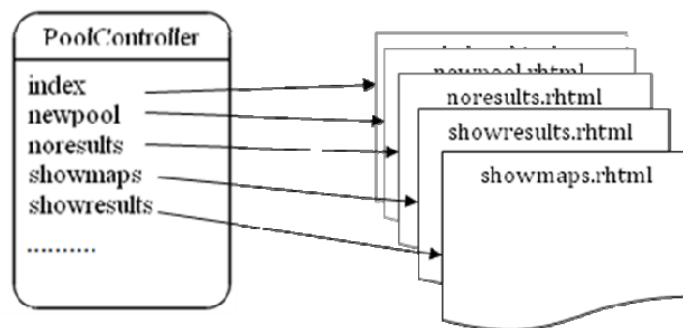


Figure 5: View Files Sample

5.2.5 MVC in Action

Rails provides a clean URL with a general pattern shown below.

`http://mywebsite.com/<controller>/<action>/<id>`

In this pattern, <controller> is the name of Controller class and actions are its methods. By convention, a view file is associated with each action from the controllers. This pattern is due to the configuration file “routes.rb” to map the URL to specific controllers and actions. It is also customizable to a few other forms.

When a user makes a request, the Router will parse the URL according to the configuration file. The Router will find the specific controller. The controller makes any necessary updates on the model and renders the corresponding view. The view then returns to the browser for display. The figure below describes this process.

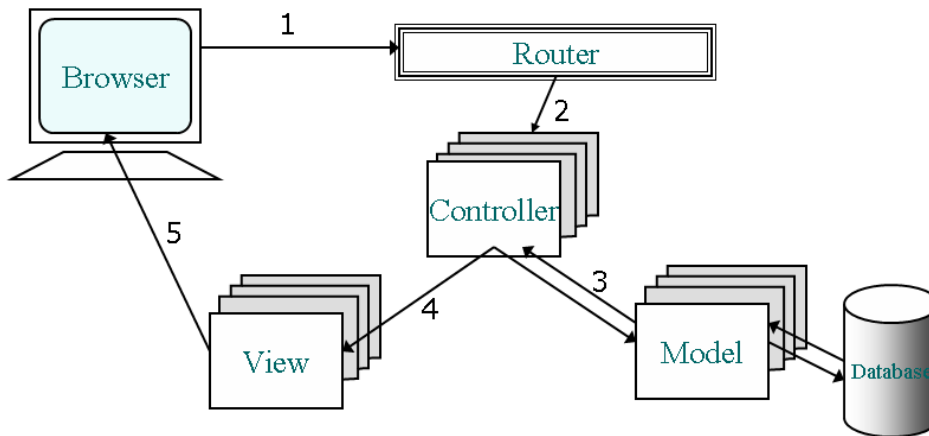


Figure 6: MVC in Action

1. For e.g. URL: <http://vernalpool/pool/search/1>
2. Router find “pool” controller
3. “pool” controller updates or retrieves appropriate models
4. “pool” controller renders “search” view
5. “search” returns to the browser for display

5.3 Highlighted Features from Data Entry

Data Entry on vernal pools is organized into five different categories or forms: pool, physical attributes, chemical attributes, animals, and plants. The pool consists of data that may not vary over time such as location, geographic co-ordinates etc. The other four categories are more time-dependant components.

For entering data, the user has two options: filling form and bulk upload. The first option is more interactive and is useful when there is limited information available. This requires the user to fill in a few forms in sequential order. The second option is more desirable if the user has already stored the information in files. This section summarizes a few highlighted features of the application.

5.3.1 Filling Form

To avoid entering multiple entries on the same pool, the decision was made to first allow the user to search for the pool by location. If the specific pool was not found, the user can enter a new pool.

Once the pool is selected from the search results or the new pool is successfully submitted, the user can now enter other information about the pool that varies over time. This information is divided into 4 forms: Physical attributes, chemical attributes, observed animals, and observed plants. Each form has a few required fields that are marked by (*) and must have valid data in order to submit the form successfully. Fields with measurable data uses the metric system and these units are visible in the form. The user is responsible for converting data into the metric unit.

The user has an option of filling all four or some of the four forms according to their available data. Some of the highlighted features of these forms are listed below.

5.3.1.1 Auto-complete

Users sometimes describe the same animals and plants by various common names. This feature shows users similar common names added by other users as a guide. An auto

complete feature is available as a Rails plugin; it handles the appearance of the floating div, the keyboard navigation, action from the mouse or Tab or complete keyboard input from the user (Hoy, 2008). These hints are the results from querying the database and the fast response is the result of Ajax technology.

5.3.1.2 Query-options

Different species of animals and plants can be given the same common name. Therefore, one common name may be associated with multiple scientific name pairs. To help users, we implemented two query-options using Rails RJS template that uses Ajax technology.

One option helps the user to find many scientific name pairs from a given common name. In case the scientific name pair does not exist in the database or it is not a desirable choice, the user can then add the new name.

Similarly, the other option helps the user to find the scientific classification from a selected scientific name. For any scientific name, there is a unique scientific classification. Since the user login feature and edit feature are yet to be implemented, we had to make an assumption that the existing classification is unique, valid and not editable.

5.3.1.3 Upload Image

A photographic image of a pool can be uploaded into the server. The first design decision was to save the image as a binary file into the database. This may overload the database tables. Therefore, later we decided to save image files into the server's directory and the database only keeps the name pointer of the file.

5.3.1.4 Google Map

We used Rails plugin YM4r GM to manipulate a Google map to display locations of pools indicated by markers. Application also provides an option for user to find the geographic co-ordinates of the pool from the map.

5.3.2 Bulk Upload

Researchers and scientists have been involved in collecting data on vernal pools for many years. These data already exist in their personal files. Bulk upload feature allows users to upload their files instead of filling out the form. However, this feature is in an early stage. The following are the conditions and assumptions made at this time:

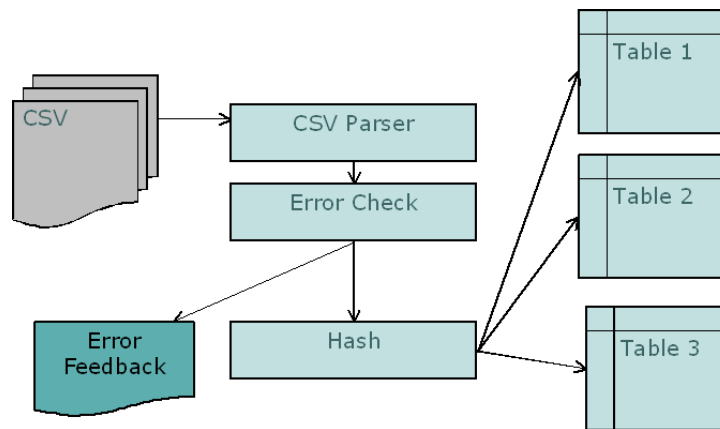


Figure 3: Bulk Upload Design

First, user must download all the standard format files and convert their existing files to follow this format. Similar to a filling a form option, there are five different possible files: pool, physical attributes, chemical attributes, animals, and plants.

The pool file may contain information for one or many pools. Adding files on multiple pools has some restrictions. All other four files of pool attributes must have an additional “poolname” column that corresponds to the “name” column in the multiple pool files. If this mapping is not correct, then it will be considered an error.

An error in any of these files will prevent any data from being saved in the database. The row of data with the error and the error details will be displayed. User must correct these errors in order to submit data successfully.

6.0 System in Action

This section highlights main features of the web application by a series of screen-shots of the website.

6.1 Home Page

The home page has four navigation options: home, search, entry, and help. Since data entry was the main focus of this project, unimplemented features are added as a placeholder for the future.

Two maps on the bottom of the page feature geographic coordinate search and pool display for the given location.

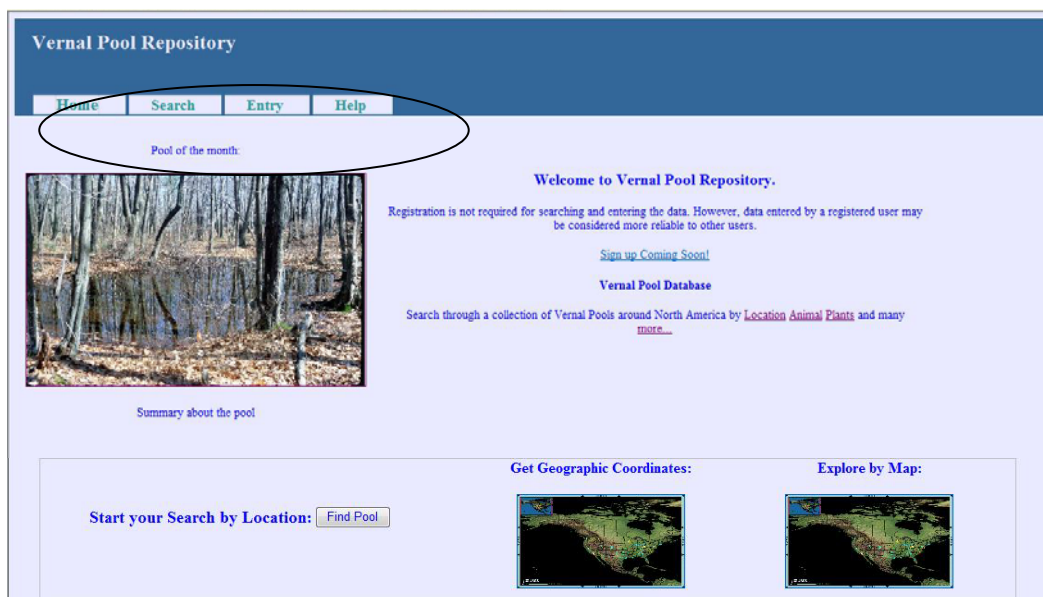


Figure 4: Home Page

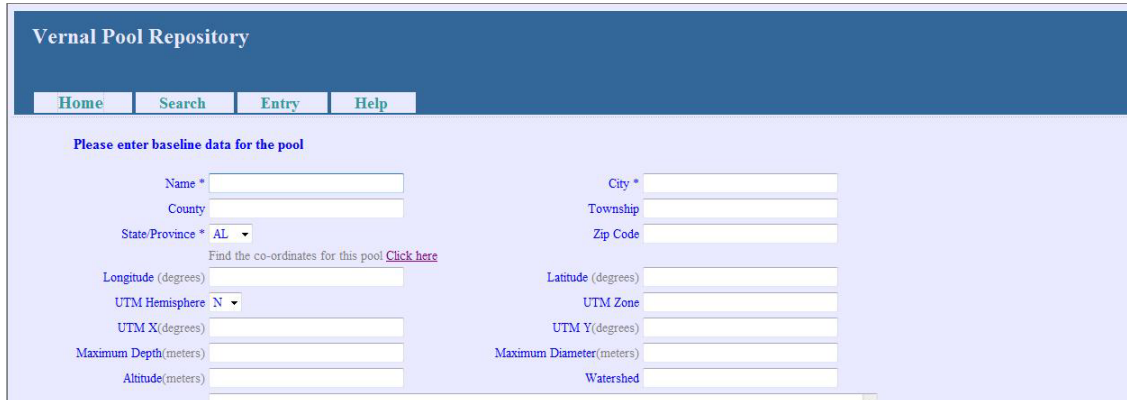
6.2 Data Entry

The entry tab in the navigation bar will give the users two options for data entry : filling form and upload file. For both options, there are few required fields that must be valid for successful submission.

6.2.1 Filling Form

First, the user must search the pool of his/her interest by location to avoid adding multiple entries on the same pool. If the pool does not exist in the database, it can be added. If the pool exists and it is selected, the user can add other information about this pool.

6.2.1.1 New Pool



The screenshot shows the 'Vernal Pool Repository' website with a navigation bar containing 'Home', 'Search', 'Entry', and 'Help'. Below the navigation bar, there is a section titled 'Please enter baseline data for the pool'. This section contains several input fields and dropdown menus arranged in two columns. The left column includes: 'Name *', 'County', 'State/Province *' (with a dropdown menu set to 'AL'), 'Longitude (degrees)', 'UTM Hemisphere' (with a dropdown menu set to 'N'), 'UTM X (degrees)', 'Maximum Depth (meters)', and 'Altitude (meters)'. The right column includes: 'City *', 'Township', 'Zip Code', 'Latitude (degrees)', 'UTM Zone', 'UTM Y (degrees)', 'Maximum Diameter (meters)', and 'Watershed'. A link labeled 'Click here' is positioned between the 'State/Province' and 'Longitude' fields, with the text 'Find the co-ordinates for this pool' above it.

Figure 5: Enter New Pool

6.2.1.2 Google Map

Geographic coordinates of a pool are optional fields, since this information is not easily available. However, this data is highly desirable. Using a Google Map API, the application provides an option for the user to find the coordinates of the pool.

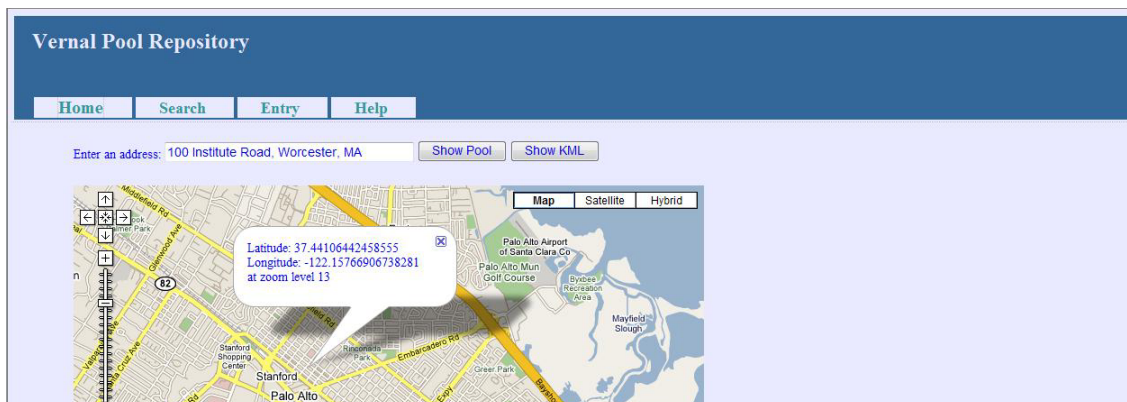


Figure 6: Find Geographic Coordinates

6.2.1.3 Upload Image

Once the form submission is successful, the user has a choice of adding an image of the pool.



Figure 7: Upload an Image

6.2.2 Existing Pool

For the selected pool, the user can add 4 categories: physical attributes, chemical attributes, animals, and plants. Submitting a form will navigate the user in a sequential order. The user also has an option to choose a specific form through the tab.

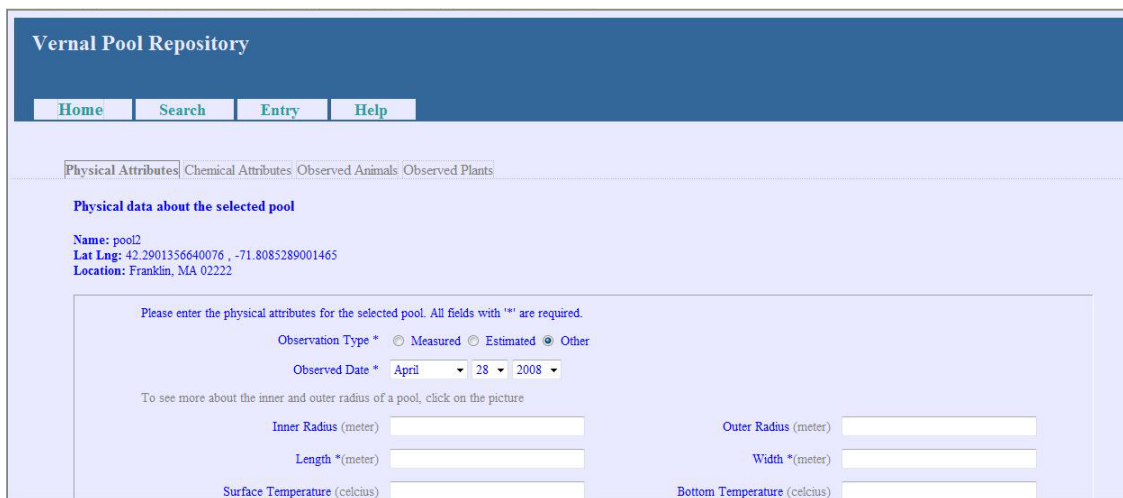


Figure 8: Data Entry Form

6.2.3 Auto-complete

An auto complete feature provides the user with hint of similar input added by other users. For example in the figure 11 below, as the user types “w”, a query to search for a pattern /*w*/ is sent to the database. This interaction is fast and the results are displayed quickly.

The hints also change in response to the user's keyboard input. The user has the option of either choosing from the hints or completing his or her input.

The screenshot shows the 'Vernal Pool Repository' website. At the top, there is a navigation bar with 'Home', 'Search', 'Entry', and 'Help' buttons. Below this, there are tabs for 'Physical Attributes', 'Chemical Attributes', 'Observed Animals', and 'Observed Plants'. The main content area is titled 'Physical data about the selected pool' and displays information for a pool named 'pool2', including its location in Franklin, MA. Below this information is a form for entering observed animals. The 'Common Name' field contains 'w' and has a dropdown menu with 'backswimmer' and 'worm' selected. The 'Observed Date' field is empty. The 'Scientific Name' field has a 'Find Genus and Species' button. There are also 'Add new Genus' and 'Add new Species' buttons.

Figure 9: Auto-complete Feature

6.2.4 Query-options

Two other options implemented using Ajax can be helpful to users.

A user can query scientific names of animals and plants by giving a common name. Either the user can select from results in the selection box, or the user can enter a new name. Similarly, using a scientific name, the user can query for the scientific classification. There is an assumption that the existing classification is unique, valid and not editable.

The screenshot shows the 'Vernal Pool Repository' website. At the top, there is a navigation bar with 'Home', 'Search', 'Entry', and 'Help' buttons. Below this, there are tabs for 'Physical Attributes', 'Chemical Attributes', 'Observed Animals', and 'Observed Plants'. The main content area is titled 'Physical data about the selected pool' and displays information for a pool named 'pool2', including its location in Franklin, MA. Below this information is a form for entering observed animals. The 'Common Name' field contains 'wood frog' and has a dropdown menu with 'wood frog' selected. The 'Observed Date' field contains 'April 28 2008'. The 'Scientific Name' field has a 'Find Genus and Species' button. There are also 'Add new Genus' and 'Add new Species' buttons.

Figure 10: Query-option

6.2.5 Error Feedback

If there are any mistakes in a user's input, the application will provide feedback. The user must correct these errors for a successful submission.

The screenshot shows the 'Vernal Pool Repository' interface. At the top, there is a navigation bar with 'Home', 'Search', 'Entry', and 'Help' buttons. Below this, there are tabs for 'Physical Attributes', 'Chemical Attributes', 'Observed Animals', and 'Observed Plants'. The main content area is titled 'Physical data about the selected pool' and displays the following information: Name: pool2, Lat Long: 42.2901356640076, -71.8085289001465, Location: Franklin, MA 02222. A red heading states '4 errors prohibited this physicalattribute from being saved'. Below this, a list of errors is provided: 'Outer radius => Must range from 0 to 16,000', 'Ice cover => Can't be blank', 'Length => Is not a number', and 'Width => Is not a number'. A form below the errors prompts the user to enter physical attributes, with a note that all fields with an asterisk are required. The form includes radio buttons for 'Observation Type' (Measured, Estimated, Other), a date selector for 'Observed Date' (April 28, 2008), and input fields for 'Inner Radius (meter)' (1.25) and 'Outer Radius (meter)' (-2.02).

Figure 11: Error Feedback

6.3 Bulk Upload

A bulk upload feature allows users to upload their files for one or multiple pools instead of filling in the form. The CSV files must follow provided standard format. An error in the files will prevent any data from being saved to the database.

The screenshot shows the 'Vernal Pool Repository' interface. At the top, there is a navigation bar with 'Home', 'Search', 'Entry', and 'Help' buttons. Below this, there are tabs for 'Physical Attributes', 'Chemical Attributes', 'Observed Animals', and 'Observed Plants'. The main content area is titled 'Bulk Upload' and contains the following text: 'If you have excel file, you must convert them to csv format.' Below this, there are five input fields for file uploads: 'Pool File', 'Physical Attribute File', 'Chemical Attribute File', 'Observed Animal File', and 'Observed Plant File'. Each field has a 'Browse...' button next to it. At the bottom of the form, there is a 'Submit Files' button.

Figure 12: Bulk Upload

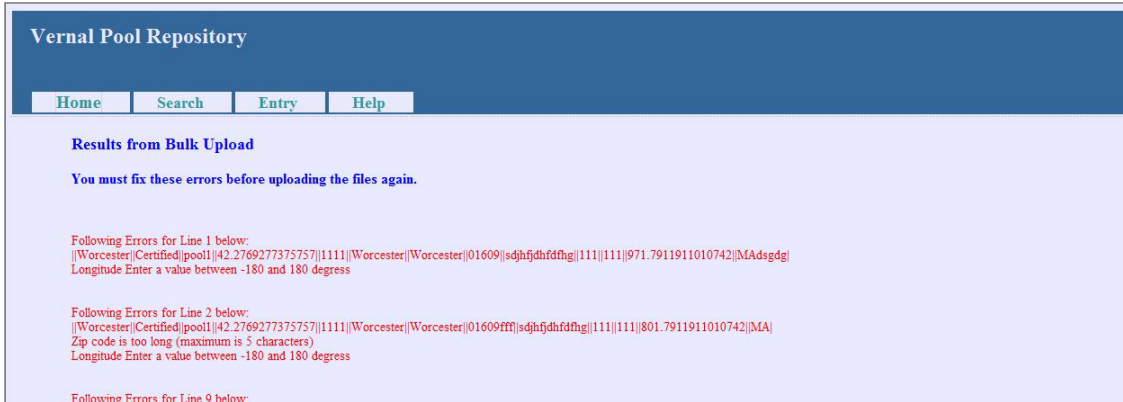


Figure 13: Bulk Upload Error

6.4 Google Map

User can view the pool location on a Google Map for a given address. Small red markers on the map indicate the location of these pools. User will see the name and geographic co-ordinates of the pools by clicking on the red markers.

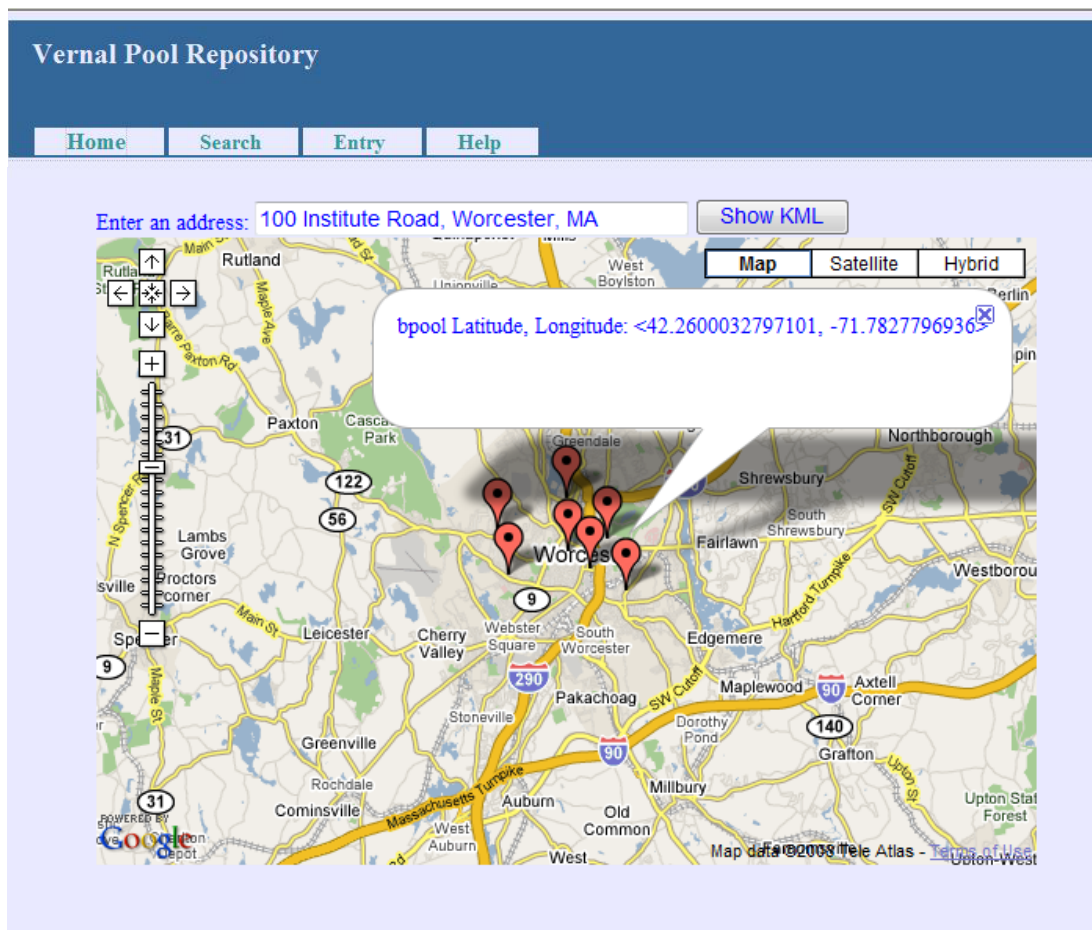


Figure 14: Google Map

7.0 Accomplishments

At the end of our project, we reevaluated our original goals. Overall, each objective was met. However, due to the limited period, our implementation is not a complete system. This section provides a summary on our completed work.

New Database - A vernal pool database was created using MySQL. There are 10 data tables and 15 relational tables, altogether 25 tables. It is capable of tracking the date and the contributing source (person) for each entry.

Basic Search - Pools can be searched by location.

Data Entry – Data entry is the main focus of this project. It has two options, form filling and bulk upload. However, bulk upload is still in its early stage.

Google Map Mash-up – User can find the geographic coordinates on a Google map as well as view the location of the pools indicated by the markers on the map.

Testing – Although no formal testing were established, simple testing was done by interacting with the application on a regular basis. The project was also made available for vernal pool community for usability testing.

8.0 Future Work and Recommendation

Due to time limitation, the project is yet to be complete for deployment. The project only focused on highly prioritized requirement from the sponsor. Remaining features, enhancements, and improvement on certain problems are some of the few possibilities for the future implementation to make this project a reality. This section provides recommendations and guidelines for future completion of the project.

8.1 Database

The database has evolved throughout our project. At the beginning, we had many procedures, functions, and a few triggers for easy search, insertion, and validation. As we advanced our project to Ruby on Rails, its object relational mapping and model validation made these procedures, functions and triggers unnecessary. Therefore, they were removed.

In retrospect, triggers and functions should be added to the database to make it fully capable of error checking and validating data on its own. Therefore, the database can exist alone without other framework layers. The database will be more efficient and easily accessible and reusable by other framework and projects and

8.2 User Interface

Below is the summary of user stories that were not completed during this project and is recommended for the future completion.

Registration and Login – The application should provide a registration process and a login mechanism for the registered users. Users should have at least three possible hierarchy and access levels: admin, regular, and guest. An unregistered user will be considered a guest.

Search – Search feature should expand from the basic search to a full-fledged advanced search. The results from the search should be downloadable into common format files.

Data Editing – After the registration and login process is complete, the contributed users should be able to edit their own entries. An appropriate policy for an editing option should be designed. One option is similar to the nature of Wikipedia, where the application does not make any claims on the validity of the data, but the user is responsible for making valid inputs. A proper disclaimer should be provided for this option.

Data Validation – The current project validation is done on the server side. A simple validation process should also be added on the client side.

Enhancements

Subscription - Users can subscribe to receive automatic notification about new entries relating to certain animal and plant species of their interest. The subscriber should be notified if any new entry is made for that species.

Reverse Entry from the Map – The application should provide an option for user to input the pool information through a Google map.

KML – An enhancement in the Google map, KML should be added so that the pool summary could be displayed in the map.

Bulk Upload – Since this feature is in an early stage, not all the ideas mentioned above are in the completed stage. Later, this feature should also support different formats besides CSV. The data validation should be done more rigorously to check for invalid or malicious input.

Test – Take advantage of test environment and automated test suite provided by Ruby on Rails. The vernal pool community should also take part on more usability tests in the future.

8.3 Useful Topics

These are some of the very important topics the team had wished to investigate thoroughly but could not due to time constraints. These topics should be considered fully in the future in order to make this project efficient.

Security – The web application needs to be secure and protected from a range of security issues. A few common security issues include SQL injection and Cross-site Scripting Attack (XSS) etc. SQL injection is one of the most common methods of attacking web applications with back-end database. The results could be extreme, from exposure to total destruction of the data. Therefore, filtering user input for any potentially malicious input is very important. In addition, redisplaying user's submitted content on an HTML page could lead to another type of vulnerability: XSS attack. A goal of the XSS attack is to get the victims' private information by running malicious code when they interact with a trusted site (Orsini, 2007).

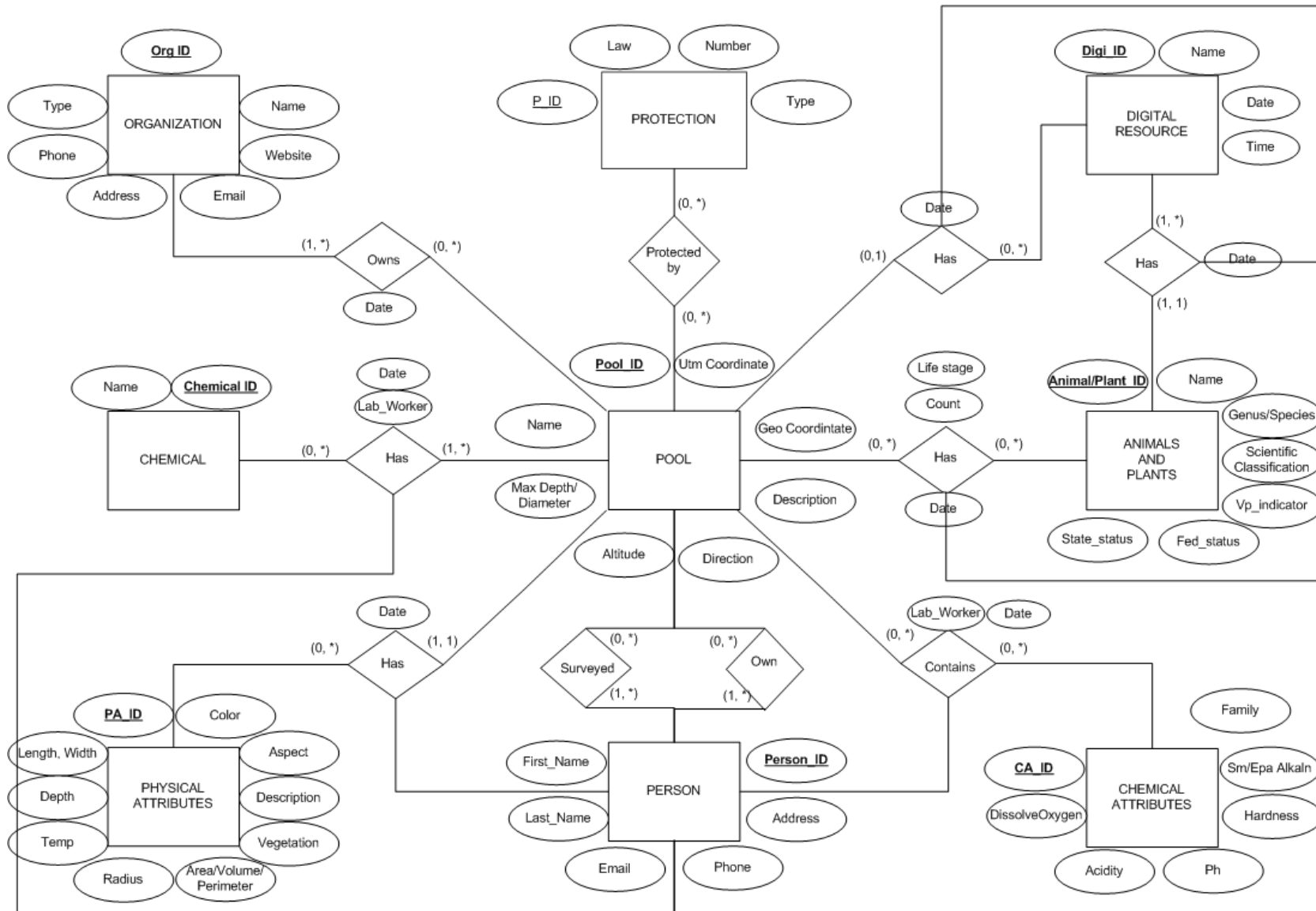
Performance and Scalability Because Ruby is an interpreted language, it is slow. However, the team should look into solutions to overcome these issues to make it possible to handle high volume of clients in the future.

9.0 Conclusion

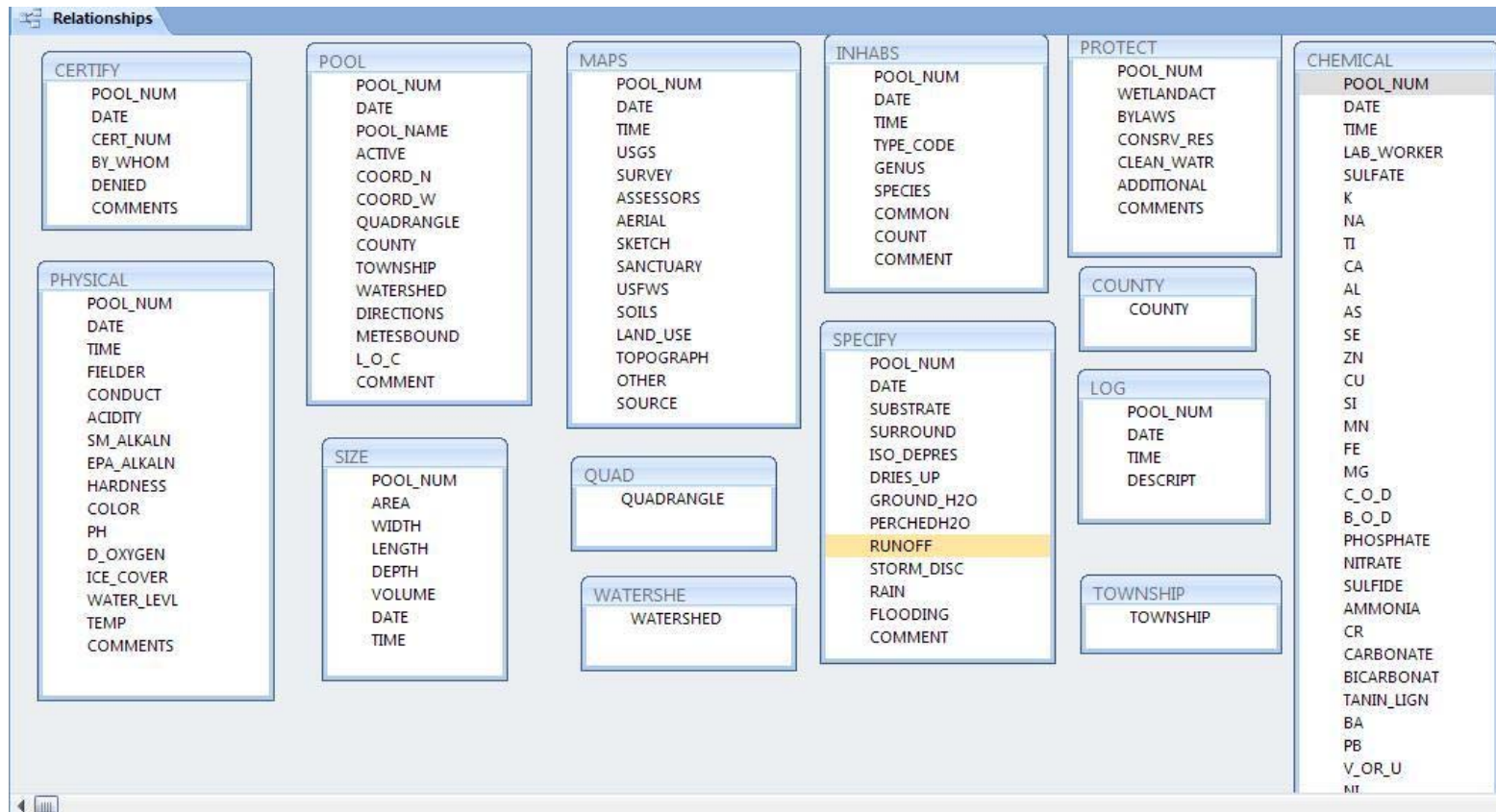
This project provided a system that aids the interaction and resource sharing for vernal pool community to our sponsor, Dr Betsy Colburn from Harvard Forest. The system includes the web application with the back-end database. The system should make the resource easily accessible for the community with an interactive learning environment.

The system will provide the community with a medium to contribute their resource, and an interactive environment for the community. It will serve as a learning tool for the general public, historians, genealogists, and other academics. The completion of this project will be a great addition to the already rich resource of vernal pool community.

Appendix A: Final E-R Diagram



Appendix B: Schema of 1993 vernal pool database



Bibliography

Dix, A., Finlay, J., Abowd, G. D., & Beale, R. (2004). *Human-Computer Interaction*. Pearson Education Limited.

Fitzgerald, M. (2007). *Learning Ruby*. O'Reilly Media Inc.

Grant, E. H. (2005). Correlated of Vernal Pool occurrence in the Massachusetts, USA Landscape. *Wetlands* , 25 (2), 48 - 87.

Hartl, M., & Prochazka, A. (2007). *RailsSpace*. Addison-Wesley Professional.

Hoy, A. (2008). *Ajaxariffic Autocomplete with Scriptaculous*. Retrieved February 10, 2008, from Slash7: <http://www.slash7.com/articles/2005/8/13/ajaxariffic-autocomplete-with-scriptaculous>

Introduction to CSS. (2008). Retrieved March 5, 2008, from W3 Schools: http://www.w3schools.com/Css/css_intro.asp

Izaak Walton League of America Fact Sheet. (2007, January 10). Retrieved September 12, 2007, from Izaak Walton League of America: http://www.iwla.org/publications/watersheds/Types_of_Wetlands_Fact_Sheet.pdf

Lenz, P. (2007). *Build your own Ruby on Rails web applications*. SitePoint .

MacCallum, W. F. (2001, January 1). *Commonwealth of Massachusetts* . Retrieved September 5, 2007, from Guidelines for the Certification of Vernal Pool Habitat: http://www.mass.gov/dfwele/dfw/nhosp_temp/vernal_pools/pdf/vpcert.pdf

Major Research Topics . (2006). Retrieved November 25, 2007, from Harvard Forrest: <http://harvardforest.fas.harvard.edu/research.html>

Natural Heritage and Endangered Species Program. (2008, January 24). Retrieved April 12, 2008, from Mass Wildlife: <http://www.mass.gov/dfwele/dfw/nhosp/nhosp.htm>

NetBeans. (n.d.). Retrieved February 20, 2008, from NetBeans IDE 6.1 Information: <http://www.netbeans.org/index.html>

Number of Certified Vernal Pools By Town. (2007, May 2). Retrieved September 21, 2007, from Massachusetts Division of Fisheries & Wildlife: http://www.mass.gov/dfwele/dfw/nhosp_temp/vernal_pools/vernal_pool_data.htm

Orsini, R. (2007). *Rails Cookbook*. O'Reilly Media, Inc.

Public Participation & News. (2007, September 12). Retrieved September 12, 2007, from Mass DEP: <http://www.mass.gov/dep/public/press/enforce.htm>

Ramp, J. M. (2004, January 1). *Abstract Detail of Pollination and seed set of the endangered vernal pool annual Lasthenia conjugens (Asteraceae)*. Retrieved September 10, 2007, from Botany 2004:
<http://www.2004.botanyconference.org/engine/search/index.php?func=detail&aid=657>

Raymond, S. (2006). *AJAX on Rails*. O'Reilly Media Inc.

Research and Education in Ecology, Conservation and Forest Biology . (2006). Retrieved November 20, 2007, from Harvard Forrest: <http://harvardforest.fas.harvard.edu>

Sebesta, R. W. (2007). *Programming the World Wide Web*. Addison-Wesley.

UML 2 Use Case Diagrams. (2006, April 3). Retrieved January 4, 2008, from Agile Modeling: <http://www.agilemodeling.com/artifacts/useCaseDiagram.htm>

User Stories. (1999). Retrieved 15 2008, January, from Extreme Programming:
<http://www.extremeprogramming.org/rules/userstories.html>

Vernal Pools. (2007, September 10). Retrieved September 29, 2007, from Woodstock Conservation Commission: <http://www.woodstockconservation.org/vernalpools.htm>

Vernal Pools. (2002, January 17). Retrieved September 28, 2007, from California Wetlands Information System: http://ceres.ca.gov/wetlands/whats_new/vernal_sjq.html

Vernal Pools. (2008, March 12). Retrieved April 10, 2008, from Mass Wildlife:
http://www.mass.gov/dfwele/dfw/nhesp/vernal_pools/vernal_pools.htm

Why MySQL. (2008). Retrieved March 12, 2008, from MySQL:
<http://www.mysql.com/why-mysql/>