

Web-Based Reporting System For Road Kill

A Thesis Presented to
The Faculty of the Computer Science Program
California State University Channel Islands

In (Partial) Fulfillment
Of the Requirements for the Degree
Masters of Science in Computer Science

By
Daniel Ojeda
August 2012

APPROVED FOR THE COMPUTER SCIENCE PROGRAM

Andrzej Bieszczad 8/20/12
Advisor: Dr. Andrzej Bieszczad Date

Sean Anderson 8/20/12
Dr Sean Anderson Date

Peter D Smith 8/20/12
Dr Peter Smith Date

APPROVED FOR THE UNIVERSITY

G. A. Berg 8-20-12
Dr Gary A. Berg Date

Non-Exclusive Distribution License

In order for California State University Channel Islands (CSUCI) to reproduce, translate and distribute your submission worldwide through the CSUCI Institutional Repository, your agreement to the following terms is necessary. The author(s) retain any copyright currently on the item as well as the ability to submit the item to publishers or other repositories.

By signing and submitting this license, you (the author(s) or copyright owner) grants to CSUCI the nonexclusive right to reproduce, translate (as defined below), and/or distribute your submission (including the abstract) worldwide in print and electronic format and in any medium, including but not limited to audio or video.

You agree that CSUCI may, without changing the content, translate the submission to any medium or format for the purpose of preservation.

You also agree that CSUCI may keep more than one copy of this submission for purposes of security, backup and preservation.

You represent that the submission is your original work, and that you have the right to grant the rights contained in this license. You also represent that your submission does not, to the best of your knowledge, infringe upon anyone's copyright. You also represent and warrant that the submission contains no libelous or other unlawful matter and makes no improper invasion of the privacy of any other person.

If the submission contains material for which you do not hold copyright, you represent that you have obtained the unrestricted permission of the copyright owner to grant CSUCI the rights required by this license, and that such third party owned material is clearly identified and acknowledged within the text or content of the submission. You take full responsibility to obtain permission to use any material that is not your own. This permission must be granted to you before you sign this form.

IF THE SUBMISSION IS BASED UPON WORK THAT HAS BEEN SPONSORED OR SUPPORTED BY AN AGENCY OR ORGANIZATION OTHER THAN CSUCI, YOU REPRESENT THAT YOU HAVE FULFILLED ANY RIGHT OF REVIEW OR OTHER OBLIGATIONS REQUIRED BY SUCH CONTRACT OR AGREEMENT.

The CSUCI Institutional Repository will clearly identify your name(s) as the author(s) or owner(s) of the submission, and will not make any alteration, other than as allowed by this license, to your submission.

Web-Based Reporting System for Road Kill

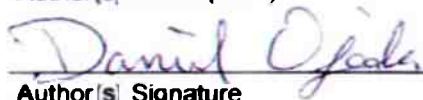
Title of Item

Web-Based Application, Road Kill, Wildlife-Vehicle Collision, Mobile Device API, Reporting System

3 to 5 keywords or phrases to describe the item

Daniel Djeda

Author(s) Name (Print)



Author(s) Signature

08/22/2012

Date

Web-Based Reporting System For Road Kill

By
Daniel Ojeda

Computer Science Program
California State University Channel Islands

Abstract

As our society continues to grow, there is a constant need to construct newer roads, so that the driving population can commute between home, school, work, and countless other destinations they wish to reach each day. In constructing these roads, city officials may not be considering the surrounding wildlife because they have insufficient information about the existing habitat. Animals in the surrounding area are crossing these treacherous roads to find food, shelter, or mates. With inadequate road planning, some animals end up colliding with commuting vehicles. This project proposes to produce a method for data entry clerks to upload data through the use of a web-based application or mobile device and for researchers to analyze data from the database on the server. City officials and engineers can implement this project to analyze the collected data, which would allow for proper planning of future and existing roads to avoid the destruction of wildlife.

The overall goal of this project is to create a repository of data related to the road kill. Such a database is critical for urban planning, with the sustainability and preservation of wildlife in mind. The purpose of the database is to provide a foundation for a variety of data mining tools that will allow researchers, planners, and the general public to obtain information that can be used for specific purposes. For example, a road can be planned through corridors that minimize the detrimental effects of wildlife-vehicle collisions upon the animals as well as the human population.

In order to implement this plan, a web-based application is needed that allows for data entry, database management, and the creation of tools for filtering and visualizing the information. Furthermore, this new system provides crowd-sourcing road kill information, through mobile applications running on devices such as the iPhone. These mobile devices utilize an Application Program Interface, which allows for data reporting and synchronization.

Acknowledgements

I would like to thank Rebekah Ojeda and Guadalupe Ojeda for their support with my thesis project. I would like to thank my mother Marisela Chávez for her support during my thesis project. I would also like to thank Dr. Andrzej Bieszczad, Dr. Sean Anderson, and Dr. Peter Smith for their participation.

Table of Contents

Chapter 1: Introduction	10
1.1 Introduction to the Situation and a Proposed Solution.....	10
1.2 The Architecture of the Proposed Solution	11
1.3 Web-Based Application	12
1.4 Mobile Device Application	14
1.5 A Summary of the Remaining Chapters.....	15
1.6 Key Terms.....	15
Chapter 2: Field Overview	17
2.1 Providing a Method for an Environmental Need	17
2.2 Structure of a Web-Based Application.....	17
2.2.1 Web-Based Application Responsibilities.....	17
2.2.2 Server Responsibilities	17
2.2.3 Database Responsibility	18
2.2.4 Communication Protocols between Client-side to Server-side.....	18
2.3 Data Mining and Potential Filters	19
2.4 Tools for Producing Web-Based Applications.....	20
2.5 Similar Applications.....	24
Chapter 3: Functions of the Web-Based Application	26
3.1 Roles for Authenticated User Login.....	26
3.2 Methods of Submitting Reports	27
3.3 Searching and Altering Reports	27
3.4 Importing Multiple Reports.....	28
3.5 Exporting Reports from Database.....	29
3.6 Displaying Data on a Map.....	29
3.7 Refining Data on a Map Using Filters.....	30
3.8 Submitting Report via Mobile Device.....	31
Chapter 4: Detailed Description of Design and Functionality of Web-Based Application	32
4.1 Web-Based Application Tools Utilized	32
4.2 Conceptual Database Design.....	33
4.2.1 Illustration of an Entity-Relationship Diagram	35
4.3 General Overview of Public and Administrator Pages	36
4.4 Login Page for Administrator.....	37
4.5 Implementation of a Single Report	37
4.5.1 Inputting Date and Time Values.....	38

4.5.2 Inputting Geospatial Values	39
4.5.3 Inputting Road Values	39
4.5.4 Inputting Kill Values	40
4.5.5 Inputting Additional Values	41
4.6 Validating, Editing, or Deleting Reports through Search	42
4.6.1 Validating Reports in the Database	42
4.6.2 Editing Reports in the Database	43
4.6.3 Omitting Reports in the Database	44
4.7 Importing and Exporting Reports	44
4.7.1 Importing Reports via a CSV File	45
4.7.2 Exporting Reports from the Web-Based Application	46
4.8 Retrieving Reports from Mobile Device	47
4.9 Mechanisms for Visually Displaying Reports on a Map	48
4.10 Generating Maps with Filters	51
Chapter 5: Conclusion	52
Chapter 6: Future Work	53
References	55
Appendix A	59
Appendix B	71
Appendix C	75

Table of Figures

Figure 1.1 Splatter Spotter Architecture	11
Figure 1.2 Web-Based Application	12
Figure 1.3 Associations with a Mobile Device Application	14
Figure 2.1 Roadkill CSUCI Website	24
Figure 2.2 CROS UCD Website	25
Figure 3.1 Roles for Authenticated User Login	26
Figure 3.2 Methods of Submitting Reports	27
Figure 3.3 Searching and Altering Reports	27
Figure 3.4 Importing Multiple Reports	28
Figure 3.5 Exporting Reports from Database	29
Figure 3.6 Displaying Data on a Map	29
Figure 3.7 Refining Data on a Map Using Filters	30
Figure 3.8 Submitting Report via Mobile Device	31
Figure 4.1 Web-Based Application Tools Utilized	32
Figure 4.2 Illustration of an Entity-Relationship Diagram	35
Figure 4.3 Overview of Public Page	36
Figure 4.4 Overview of Administrator Page	36
Figure 4.5 Login Page for Administrator	37
Figure 4.6 Implementation of a Single Report	37
Figure 4.7 Inputting Date and Time Values	38
Figure 4.8 Inputting Geospatial Values	39
Figure 4.9 Inputting Road Values	39
Figure 4.10 Inputting Kill Values	40
Figure 4.11 Inputting Additional Values	41
Figure 4.12 Validating, Editing, or Deleting Reports through Search	42
Figure 4.13 Search Legend	42
Figure 4.14 Editing Reports in the Database	43
Figure 4.15 Importing Reports via a CSV File	45
Figure 4.16 Exporting Reports from the Web-Based Application	46
Figure 4.17 Retrieving Reports from Mobile Device	47
Figure 4.18 Visually Displaying Reports on a Map	48
Figure 4.19 Informing that there are no Reports found	48
Figure 4.20 Differentiating Report Markers	49

Figure 4.21 Map Containing a Transect without a Report.....	49
Figure 4.22 Map Containing a Transect with a Single Report.....	49
Figure 4.23 Map Containing a Single Report without Transect	50
Figure 4.24 Filters that are applied to the Map	51
Figure 4.25 Generated Map with Search Keyword and Distance Applied.....	51

Chapter 1: Introduction

1.1 Introduction to the Situation and a Proposed Solution

In today's society, roads are a vital pathway for people to commute between home, school, work, hospitals and countless other locations. Roads permit all varieties of vehicles to traverse the land in order to reach their desired destinations. Some of these roads were constructed with the intention of serving human needs and not properly planned to include the needs of animals in the surrounding area. These roads may plague the wildlife of the surrounding habitat in that they interfere with the normal route of an animal seeking food, shelter, mates, and so on. In order to construct roads properly, there needs to be a strategy to eradicate such intrusions.

The application presented in this body of work provided methods of collecting and analyzing data as a tool for developing future roads and for altering existing roads so that both wildlife and humans can coexist in the same space. The collection and analysis of data is based on wildlife-vehicle collisions. A wildlife-vehicle collision is defined as an incident in which any animal in the vicinity of a road is struck by a vehicle and is deceased. Wildlife-vehicle collision data is vital in properly developing future roads and for altering existing roads. Appropriate tools for the collection, storage, mining, and analysis of relevant data need to be developed and utilized. With such information researched, an engineer could conclude which roads need altering, where to place future roads, and the effect on the surrounding wildlife from existing roads in the area.

1.2 The Architecture of the Proposed Solution

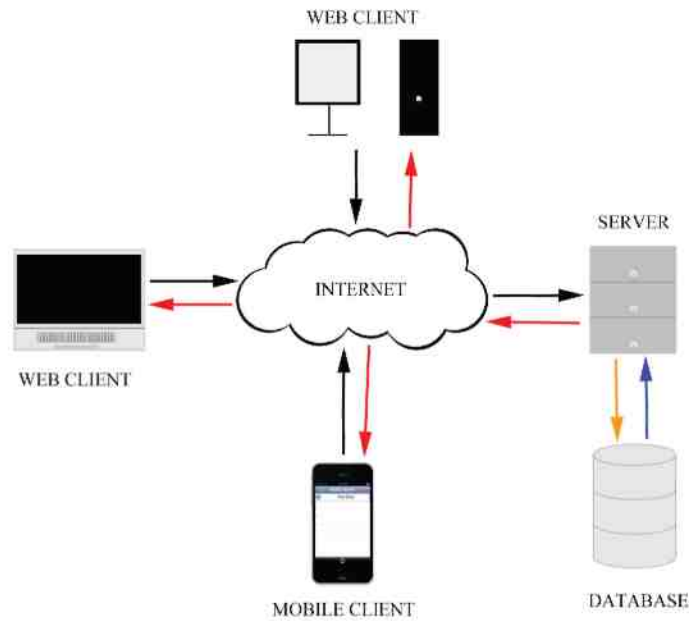


Figure 1.1 Splatter Spotter Architecture

Splatter Spotter is an application that permits an individual to collect, store, and analyze information about wildlife-vehicle collisions. There are several elements of the architecture, as shown in Figure 1.1, which are: server, database, web clients, and mobile client. Each element provides different functionalities for an end user.

SERVER

A server is present to allow proper communication between the database and the web client and for constructing the web-based application. A server is used as a place for storing data in a central location and accessing the web-based application.

DATABASE

The database allows for storing information into one central location and for retrieving and analyzing information for display. A database is used to store information about reports that were uploaded through the web client or the mobile client.

WEB CLIENT

The web client provides a data entry clerk and researcher access to a web-based application where functions permit certain tasks to be performed. The black arrows, from the web clients to the Internet in Figure 1.1, indicate access to the web-based application and the functions therein. Data entry clerks access the server through the Internet for storing information on the database. The researchers access the server through the

Internet for displaying privileged data held within the database. The orange arrow, in the Figure, is for storing information sent from the server to the database. Another use for the orange arrow is for querying data from the database. The blue arrow, in the Figure, is the queried information sent from the database to the server. The red arrows, in the Figure, are utilized for displaying the transfer of information to the web clients that were retrieved from the database. Another use of the red arrow is to display communication between the server and web clients for errors or success when utilizing the functions on the web-based application.

MOBILE CLIENT

A mobile client utilizes native applications for communicating to and from the server. The black arrow from the mobile client to the Internet, in Figure 1.1, displays access to the server through the Internet. Data entry clerks access the server through the internet for storing information on the database. The orange arrow, in the Figure, shows stored information being sent from the server to the database. Another use of the orange arrow is for querying data from the database. The blue arrow, in the Figure, is the queried information sent from the database to the server. The red arrow, in the Figure, is also utilized to display communication between the server and the mobile client, be it errors or success when attempting to store information.

1.3 Web-Based Application

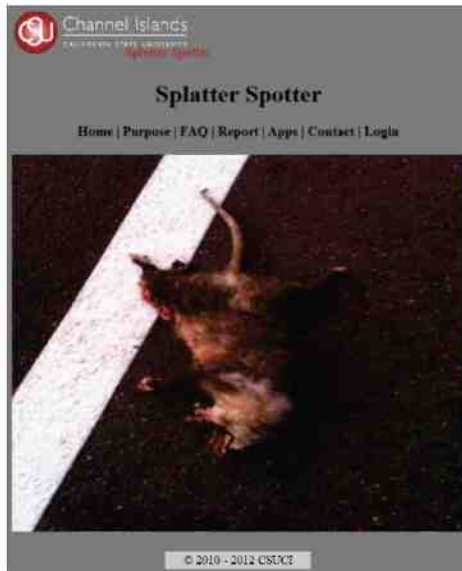


Figure 1.2 Web-Based Application

This thesis focuses on the server-side and web-based application of the Splatter Spotter project, as shown in Figure 1.2. There are three aspects of this project that comprise the server-side and client-side which are: database, server, and web-based application. These three elements help in data collection and analysis.

The database has an essential role of storing information and allowing information to be retrieved. The reporting system in this application is constructed on what attributes are in the database. Functions on the web-based application revolve around what is present in the database.

The purpose of the server is to allow communication between a web client or a mobile client with the database, storing data, and for running scripts on the operating system's environment. A custom API was constructed that allows a mobile client to submit single reports to the database on the server. There are scripts on the server that are executed for creating smaller images for a faster page load and for removing undesirable images during the single report submission. The server allows the web-based application to dynamically create web pages based on user interactions.

The web-based application has an imperative role of producing functions for data entry clerks and researchers, making it easier for them to upload and analyze data. A data entry clerk has the capability of submitting a single report of an observation to the database. Another feature for a data entry clerk is to import, which allows for submitting multiple records at once. Researchers have the capability of searching all records from the database to alter or view. A researcher can also export all records from the database to analyze offline or for backup purposes. Another task for researchers is the use of maps and filters. Maps help visually display each record from the database as a point on a map. Applying filters on a map allow for data mining records from the database. Based on the records retrieved from the database, those records are overlaid on a map. Some analyses that can be made from the data include what roads need adjusting and what kinds of animals are being killed on the roads.

1.4 Mobile Device Application



Figure 1.3 Associations with a Mobile Device Application

The following is a brief overview of the mobile application part of this project. Please see Ojeda's body of work [18] for more details. Many individuals around the globe have iPhones which permit running applications on the mobile device. The mobile device application, shown in Figure 1.3, parallels the same criteria for the web-based application.

The mobile device application is capable of gathering reports while in the field. A report system on the application allows for reporting either a single report or multiple reports with the use of transect. A single report can either obtain Internet to gather some data and input automatically into the report. If there is no Internet obtained, then all data is manually input. A transect allows the submission of reports with or without waypoints to the local and remote databases. These reports are saved locally to the mobile device for the purpose of completing reports at a later time or for viewing reports that have been completed. A single report allows for an optional photo of the critter on the road.

The application also has the ability to send information to the web-based application through the use of an API that was constructed particularly for mobile devices. There are validations that are enforced to restrict certain characters that are input from the user that may become an issue for inserting reports into the database. Reports from the mobile application are sent from the device and stored into the remote database that can be later analyzed through the web-based application.

1.5 A Summary of the Remaining Chapters

Chapter two discusses the field overview of the web-based application.

Chapter three discusses functions utilized for the web-based application.

Chapter four discusses the detailed description of the design and functionality of the web-based application.

Chapter five states the conclusion of the thesis.

Chapter six provides suggestions for any future work that can be done or work that was not completed in this project.

1.6 Key Terms

API – Application Programming Interface

CSV – Comma-Separated Values

Numbers – Apple’s Numbers Spreadsheet

XLSX – Microsoft’s Excel Spreadsheet

ODS – Open Document Spreadsheet

Ubuntu – open source computer operating system

HTML – HyperText Markup Language

CSS3 – Cascading Style Sheets, third generation standard

PHP – Hypertext Preprocessor

MySQL – My Structured Query Language

JavaScript – client-side scripting language

AJAX – Asynchronous JavaScript and XML

XML – Extensible Markup Language

HTTP – Hypertext Transfer Protocol

Imagemagick – Image altering software

Zip – Data compression and archive format

SHA2 – Secure Hash Algorithm 2, cryptographic hash functions

InnoDB – Storage engine for MySQL

WWW – World Wide Web

Transect – Recorded occurrences along a path

Waypoints – Collection of coordinates along a route defined by latitude and longitude.

Chapter 2: Field Overview

This chapter provides information about creating a method for utilizing wildlife-vehicle collisions to help improve the environment with volunteer support. A general overview is provided to show the responsibilities of the server, database, and web-based application. A communication protocol between web/mobile clients (client-side) and server/database (server-side) is explained. Various filters are discussed for this project in that data that has been mined can be visually displayed onto a map. Web-based application tools are identified. Lastly, alternative methods are mentioned for building web-based applications that aid in combating wildlife-vehicle collisions.

2.1 Providing a Method for an Environmental Need

There is a need to construct methods for investigating factors that impact wildlife-vehicle collisions in order to influence the outcome of how roads are constructed [14]. The web-based application created for this project assists researchers in developing a method for investigating such occurrences. Hopefully, through the use of filters, this project can allow data mining techniques, such as predictive modeling, to be applied onto a map to show any risk factors due to animal mortality on a given road [14]. Individuals around the world are willing to help if only they had the tools to exercise their right to save their wildlife environment. A tool that has proven to work has been a web-based application. A web-based application geared to a certain topic such as wildlife-vehicle collisions can generate many visitors willing to donate their time by submitting reports.

2.2 Structure of a Web-Based Application

2.2.1 Web-Based Application Responsibilities

This project takes advantage of utilizing a web-based application to assist individuals in becoming more accountable for their surrounding environment. The web-based application allows users to interact with information held in the database. A communication protocol was implemented to send and receive data between web clients and server/database. The web-based application allows researchers to data mine information from the database which can be displayed onto a map. With the use of a web-based application, traditional methods such as pen and paper and emails can be improved.

2.2.2 Server Responsibilities

A server is a set of programs to provide services to clients on a private or public network completing tasks on behalf of the client [55]. Services that a server can provide for this project are data storage, creation of functions for the web-based application, and

computational tasks for data mining. The server allows the web-based application to access the database for storing and retrieving information in conjunction with programming languages running on the server-side. The web-based application can send commands to be executed as tasks on the server. A server allows for a web-based application to be produced in numerous programming languages, which enhances functions on the client-side. Also, a server is beneficial in centralizing information into one location for data access.

2.2.3 Database Responsibility

The responsibility of the database in this project was to retain information from multiple input locations. One location was on a web browser and another was on a mobile device. Each table in the database holds essential information or attributes specifically for this project. The database's responsibility does not only include accepting input but was used to output or display information onto a generated map.

The data that was collected for the project was separated into five categories: Date/Time, Geospatial, Road, Kill, and Additional. Each category was utilized to get the most accurate information from a data entry clerk. The Date/Time category allows for inputting information based on time and place of the report. The Geospatial category allows for inputting information of the coordinates into the report as well as explaining how these coordinates were achieved. The Road category allows for inputting information about the surrounding area around the road kill. The Kill category allows for inputting information about the animal and specifying the location. The final category was Additional, which allows for inputting extra information that was not mentioned in the previous categories.

2.2.4 Communication Protocols between Client-side to Server-side

The architecture for the project is shown in Figure 1.1. One side consists of clients (web clients and mobile client), and the other side consists of the server and database.

Web clients

A communication protocol was created to allow web clients to report road kill data. A data entry clerk or researcher inputs data into the given input areas using a web browser. When these users submit or apply the data, the data is sent through an HTTP connection as a string in plain text, represented as a black arrow, in Figure 1.1. The server receives and separates each value within the sent data to validate for inaccuracy. If there are any errors when validating, a response is sent back to the web clients through the same HTTP connection, represented as a red arrow, in Figure 1.1. If errors do not exist, then the parameters are separated and inserted into the database, represented as an orange arrow, in Figure 1.1. If the data were successfully inserted or an error occurred, a response is sent from the database to the server, represented as a blue arrow, in Figure

1.1. The server relays this response to the web clients, represented as a red arrow. The web browser displays this response to the user accordingly.

Mobile client

A communication protocol was also implemented for the mobile client to report data. A data entry clerk inputs data into the given input areas using a mobile device. When the user submits the data, the data is sent through an HTTP connection as a string in plain text, represented as a black arrow, in Figure 1.1. The server receives and separates each value within the sent data to validate for inaccuracy. If there are any errors when validating, a response is sent back to the mobile client through the same HTTP connection, represented as a red arrow, in Figure 1.1. If errors do not exist, then the parameters are separated and inserted into the database, represented as an orange arrow, in Figure 1.1. If the data were successfully inserted or an error occurred, then a response is sent from the database to the server, represented as a blue arrow, in Figure 1.1. The server relays this response to the mobile client, represented as a red arrow. The mobile device displays this response to the user accordingly.

2.3 Data Mining and Potential Filters

Data mining is the process of discovering meaningful information from raw data within data repositories such as databases [15, 35]. There are two categories for data mining which are predictive and descriptive tasks. A predictive task is making a prediction of an outcome based on information that exists within a data set. A descriptive task is finding patterns between data in a data set using four data mining tasks such as: predictive modeling, association analysis, cluster analysis, and anomaly detection.

Predictive modeling predicts the outcome of a decision based on variables either discrete (classification) or continuous (regression) from a data set where the prediction has the smallest error possible. Association analysis is discovering relationships between multiple variables within a data set based on how frequently these variables occur with each other. Cluster analysis is the process of grouping data that is based on variables with similar attributes. Anomaly detection or outlier detection is finding data that does not belong where values are different from the normal values. Researchers can find problematic areas on a given road using anyone of these data mining techniques.

A filter is extracting a smaller data set from a larger data set based on a given keyword [38]. Filters can be created to apply data mining techniques mentioned above, so that researchers can access specific data from the database. A predictive filter can help predict which areas are potentially problematic, based on variable values within the database. An associate filter can help find which animals are of highest concern due to their constant collisions. A cluster filter can group a set of data values based on distances from a certain location or animal classification. A cluster filter is slightly applied to this project using the Haversine [40] function to group only reports that exist within a given

circular range. An anomaly detection filter can detect abnormal values from within the database, removing those reports.

2.4 Tools for Producing Web-Based Applications

Operating Systems

An operating system provides services for computer programs and hardware resources [52]. Operating systems that can offer comparable features are: Linux [4], Windows Server [17] and Macintosh Server [3]. All three provide tools for creating web-based applications, a server, and the ability to maintain a database. All three have the capability to run scripts on the operating system for completing certain tasks, but Linux has the ability of running cronjobs [33], which is an essential feature that automatically removes undesirable information from the server. Linux allows for third party software to be installed such as ImageMagick [12], which converts images from their original state to a more manageable image. Windows, Linux, and Macintosh yield the same results.

Creating and Styling Web Pages

A web-based application needs to be published and an individual can go about that by creating web pages. HTML is the standard markup language for creating all web pages [41]. There are predefined markup tags which outline the specific portions of a web page.

After a web page has been created, the web page needs to be styled for a more aesthetic presentation. CSS [29] was used for creating the presentations, semantics, looks, or formats for the web-based application. CSS can be incorporated into the HTML markup tags or used as an external file. CSS helps modify the layout, colors, and fonts of a web page.

An alternative way of creating an interactive web page without following the normal standards of HTML is with the use of Flash. Flash uses ActionScript which allows web pages to become interactive with user input [25]. Developers do not need to know additional languages for creating outlines, styling pages, and client-side interactivity. Flash is an ideal candidate for replacing the standard form of creating web pages but is mainly used for creating advertisements and games [25]. Silverlight and JavaFX are also alternatives to the normal standards of HTML, as they assist developers in creating web-based applications with which users can interact.

Independent of creating a web-based application would be the use of online website builders. There exist online services where an individual can create custom web-based applications without knowing how to code such applications. Such services are Weebly [24] and Google Sites [8]. Both services provide drag and drop features and host free of charge. Online website builders are not ideal as more advanced features are needed to create the web-based application to yield specific results.

Client-Side Languages

A web page has been created and styled, but an individual needs to interact with an enhanced user interface that dynamically changes to user interaction on the client-side. JavaScript [45] aids in creating dynamic web pages and enhances the user interface. The use of JavaScript in AJAX aids in accessing the Document Object Model (DOM) [37], which are the object elements within the HTML tags. JavaScript aids in manipulating the tags or elements that can enhance the presentation. The only drawback of using JavaScript is that all functions or interactions have to be coded manually.

Another alternative client-side scripting language could be jQuery [13] which provides greater functionality when using JavaScript. jQuery was created for selecting DOM elements within HTML tags, handling events, and interacting with AJAX [47]. jQuery provides additional features through predefined libraries which make jQuery an ideal candidate for replacing JavaScript. JavaScript allows for developers to create their own functions based on their needs without a predefined library.

Server-Side Languages

A web-based application has been created, but an individual needs to interact with an enhanced user interface that dynamically changes to user interaction on the server-side. PHP [21], ASP.NET [28], JavaServer Pages (JSP) [46], and ColdFusion [30] are some server-side scripting languages which can create dynamic web pages. All languages embed code into the HTML source code and are interpreted when the page is generated. PHP and ASP.NET utilize Object-oriented programming (OOP) [50] to assist in creating dynamic or custom web pages, where JSP and ColdFusion do not. Each language could be used in combination with HTML markup tags and CSS. Each language could be used in conjunction with Relational Database Management Systems for data storing and manipulating. Each language uses the Common Gateway Interface (CGI) to run external programs under HTTP servers [10, 31]. Each language is capable of producing similar results for the web-based application created for the project.

HTTP Servers

A web-based application is created, styled, and enhanced for user interaction, but needs to be published to the Internet for everyone to view. An http server (daemon) (httpd) is used to allow communication between the client-side (browser) and the server-side (server) [44]. Apache [1], Apache Tomcat [2], and Information Services (IIS) [16] allow communications between the client-side and the server-side. Apache is a widely used http daemon under all operating systems and is written in XML and C languages. Apache Tomcat is similar to Apache except that Tomcat uses Java as its primary language. IIS is used primarily on Windows machines and is an ideal candidate for publishing web-based applications. Each HTTP Daemon would yield the same results.

Client-side to Server-side Languages

Once the client-side and server-side pages or scripts are created, there needs to be a way to communicate between both sides. AJAX can be used on the client-side for the purpose of interacting with the server-side. Data retrieval from the server-side can be retrieved asynchronously by running in the background which does not interfere with the behavior of the web page [26]. Partial page refresh would benefit the web-based application by avoiding full page reloads which can take more time to render. AJAX permits the client-side to interact with the server-side, using HTTP. Data communication between the client-side and server-side would be widely used throughout the project. AJAX is supported by all web browsers and aids in creating dynamic web pages.

XUL (XML User Interface Language) is an alternative client-side language that provides communication between the client-side and server-side. XUL uses other technologies such as CSS, JavaScript, and DOM to interact with the website. XUL has a downside in that there are no specifications and does not operate with non-Gecko applications [61]. Therefore, XUL is not an ideal candidate as it is not supported on all web browsers.

Applet is another alternative mechanism that provides communication between the client-side and the server-side which is written in Java. Applet allows for creating interaction between the user and the applet which is appropriate for demonstrations, visualizations, and teachings [27]. The only downside of implementing an applet is that the applet cannot interact with a website, therefore making the website non-dynamic. Applet is not as ideal a candidate as the web-based application would be based on users dynamically altering the web page.

Communication Protocol between Client-side and Server-side

Client-side and server-side interaction is based on HTTP. HTTP is an application level protocol and is the basis for data communication on the WWW [9, 44]. There is a flaw in using HTTP as a message from the client-side to the server-side because it is not encrypted, therefore leaving the sensitive message open for all to read. HTTPS (Hypertext Transfer Protocol Secure) remedies the un-encrypted flaw through HTTP by providing a Secure Socket Layer (SSL) to securely send messages between the client-side and the server-side. HTTPS encrypts an entire message that includes the headers and request/response load before sending information to a destination. When the encrypted message arrives at the destination, the message is then decrypted, displaying sensitive information [43].

Structured Response System from Server-side to Client-side

With the communication protocol in place, there needs to be a structure of sending data to the client-side from the server-side. XML, JSON [48], and YAML [62] are human-readable and machine-readable standards in which to send information as textual data [59]. XML, JSON, and YAML aid in creating dynamic errors or success

messages which are sent to web clients and mobile clients. XML is the default encoding for sending information from the server to the clients. XML has a drawback in that neither the standards nor the structure has been improved therefore providing no additional functionalities from the time it was conceived. XML encodes their structure using the UTF-8 standard which provides compatibility with parsers [11, 58]. JSON or YAML are not complex to implement but are similar to an XML structure, therefore making them ideal candidates for replacing XML as a communication protocol. JSON is smaller, faster, and easier to parse than XML [22]. YAML provides similar benefits as JSON, but is not widely known.

Relational Database Management Systems

With the communication protocol in place and the server scripting language setup, there needs to be a data warehouse where data can be stored in a central location. MySQL [20], Oracle Database [49], and SQLite are Relational Database Management Systems (RDBMS) [54] and can be used as the warehouse for storing data. Database Management System (DBMS) allows for controlling the creation and maintenance of the database [36]. DBMS allows for storing data into tables and retrieving data by using simple queries. The relationship of data is also stored in the tables along with attribute values. MySQL, Oracle Database, and SQLite do not provide a Graphical User Interface (GUI) [39] tool; therefore, one needs to learn how to use a command-line interface [32] to manipulate the database and the contents within. If a user does not want to learn the command-line, then there are additional tools that can be utilized to provide a GUI interface instead. MySQL allows for the use of VIEW tables to separate sensitive data from the remaining tables. MySQL has some limitations of features [49] where Oracle Database has none. Oracle Database is an ideal candidate for a relational database management system for growth and for more complex commands. The only drawback of using Oracle database is the cost of using its services. MySQL and SQLite are free to use where as Oracle databases charge per license [53]. SQLite is used in many applications such as mobile devices and web-based applications. SQLite is not a standalone database client as is MySQL or Oracle Database, but is integrated into the application making it a reasonable substitute for various test projects [57]. The downside of using SQLite is that there is no feature for producing VIEW tables which is utilized in this web-based application.

Visually Displaying information on a Map

There needs to be a way of visually displaying all reports stored in the database. For a visual display, Google Maps JavaScript API [7] or OpenStreetMap (OSM) [51] can be implemented to display all reports from within the database onto a map. Google Maps and OSM have the ability of overlaying information onto a map. Through the use of this map feature, the web-based application pinpoints reports and overlays useful transect routes or paths based on their locations. OSM allows for editing maps of the world [51] where individuals can use the maps in any manner of their choosing [19]. OSM has an advantage over Google Maps due to the cost of licenses for using enhanced features. A disadvantage of using OSM is that in order to apply route overlays, a third party software

is needed. Google Maps has an advantage over OSM, as there are predefined functions that can be applied to routes that are overlaid onto a map.

Web Browser

With the web-based application published and all features and functions applied, there needs to be a way to view this information. A web browser is an application that helps in viewing information on the WWW [60]. All web browsers are HTML formatters that render web pages. Web browsers were dominantly used on personal computers before the era of mobile devices. With mobile devices, an individual can still use a web browser to view online content anywhere with the use of an Internet connection. Web browsers follow protocols based on HTTP for retrieving information. After the information is displayed through a layout engine, the user is able to interact with the web page [59].

2.5 Similar Applications



Figure 2.1 Roadkill CSUCI Website

California State University Channel Islands (CSUCI) Roadkill, shown in Figure 2.1 is a website that brings attention to the subject of road kill [6]. This website is hosted by CSUCI. This website is strictly for informational purposes.



Figure 2.2 CROS UCD Website

California Roadkill Observation System (CROS), shown in Figure 2.2, is another website that brings attention to the subject of road kill [5] which is hosted by University of California Davis (UCD). The website permits individuals to interact with data in the database. The web-based application permits data entry clerks to input reports based on their observations. The web-based application allows information to be displayed visually with the use of a map and a filter system. There is a search feature that allows viewing reports based on keywords. Another feature is creating users and tracking reports for each user.

Chapter 3: Functions of the Web-Based Application

This chapter discusses the project as a set of functions for the web-based application. Authenticated users can log into privileged areas containing data manipulating features. To create a larger pool of data, reports can be recorded using the web-based application or mobile device. Displaying information appropriately helps researchers seek data to manipulate. Data entry clerks can import numerous reports at once. Exporting reports is beneficial for offline analyzes and backups. Creating a map helps researchers in visually displaying logged reports. Filters refine data that is visually displayed on a map.

3.1 Roles for Authenticated User Login

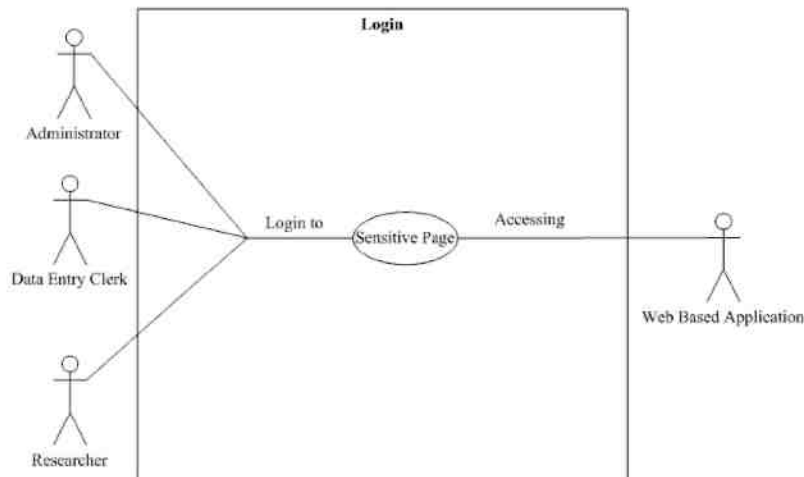


Figure 3.1 Roles for Authenticated User Login

An administrator, data entry clerk, and researcher have the capability to access secure sections for retrieving and manipulating privileged data. Login is a feature that allows the authentic user's access to restricted pages within the web-based application, as shown in Figure 3.1. An administrator accesses restricted pages to maintain issues with the web-based application. A researcher accesses restricted pages to search and validate records; also, he/she can utilize data mining and visualization tools. An administrator or researcher can upload reports as a data entry clerk.

3.2 Methods of Submitting Reports

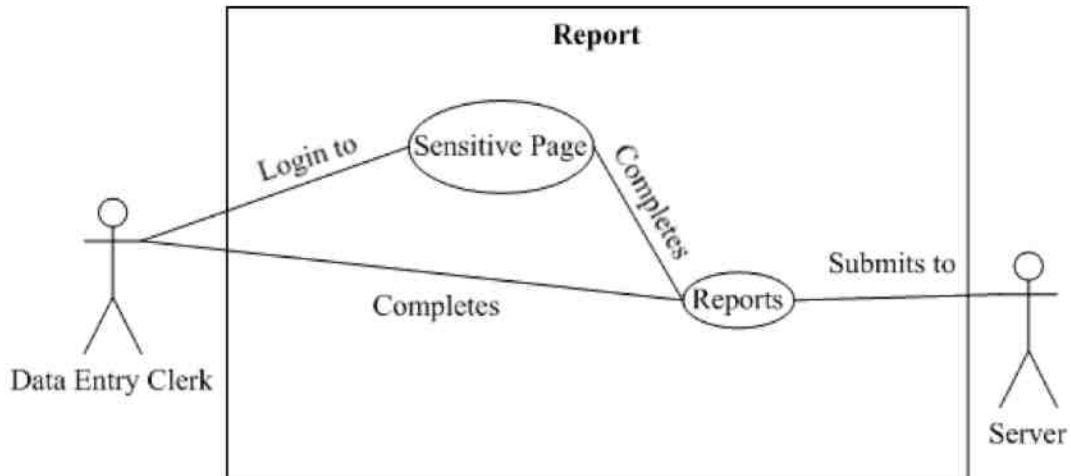


Figure 3.2 Methods of Submitting Reports

A data entry clerk has the capability to submit reports from a well-organized report system. Data entry clerks have two methods of accessing the report feature: the first is to access the feature through a login system, or the second is to access the report without a login system on the web-based application. The data entry clerk fills in all required fields in each category given in the report system. Once the data entry clerk completes the report, the report is stored into a single pool of data on the server, as shown in Figure 3.2.

3.3 Searching and Altering Reports

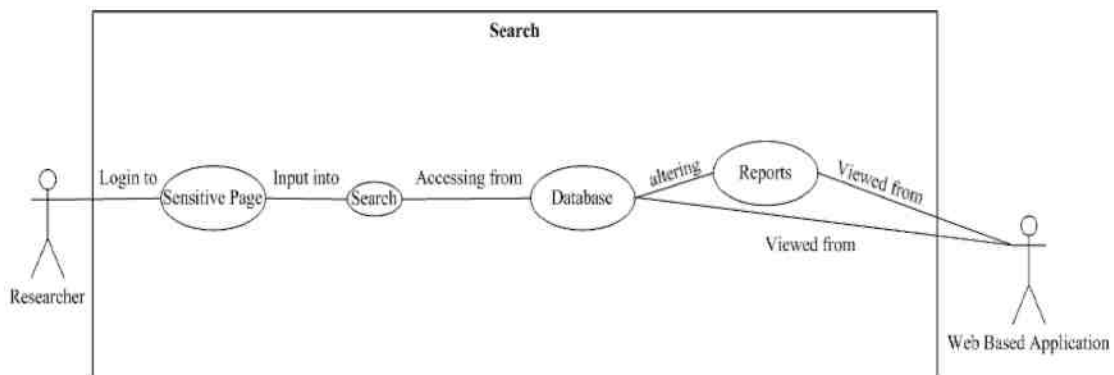


Figure 3.3 Searching and Altering Reports

A researcher has the capability to search the database for specific reports for further processing. A researcher has to login to the web-based application to access the search feature. Searching accesses data and displays each record in a comprehensible layout. Researchers are able to alter reports that are unacceptable by altering data or

omitting reports, as shown in Figure 3.3. The data entry clerk is also able to validate each record that is presented in the search results.

3.4 Importing Multiple Reports

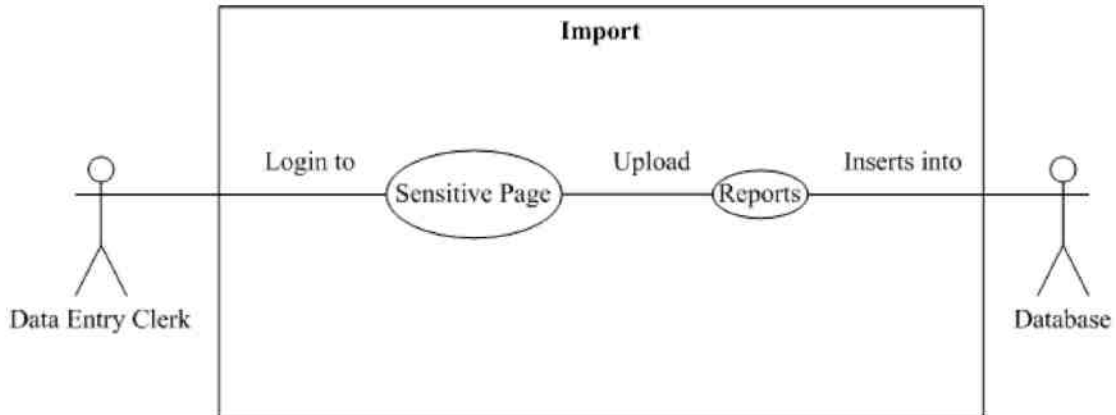


Figure 3.4 Importing Multiple Reports

If a data entry clerk has numerous sets of reports to insert, there needs to be a way to import all reports without physically inserting each one at a time. A data entry clerk has to login to the web-based application to access the import feature. The data entry clerk can upload multiple reports at once utilizing a single file. These reports are inserted into the database where other reports are gathered, as shown in Figure 3.4. Each line in the file is treated as a single record in the database consisting of multiple values based on the report system mentioned earlier.

3.5 Exporting Reports from Database

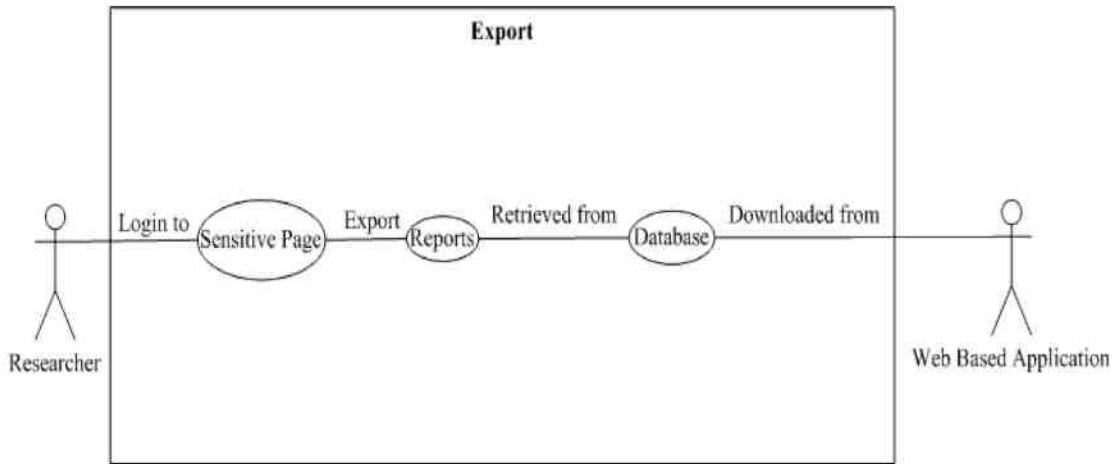


Figure 3.5 Exporting Reports from Database

Researchers need a method for reserving data and conducting offline analyzes. A researcher has to login to the web-based application to access the export feature. Export is a feature which obtains all reports from within the database which is saved into a single file and can be downloaded via the web-based application, as shown in Figure 3.5. Each record is treated as a single line in the export file using delimiters to separate each field, and an enclosure to specify what values are in each field.

3.6 Displaying Data on a Map

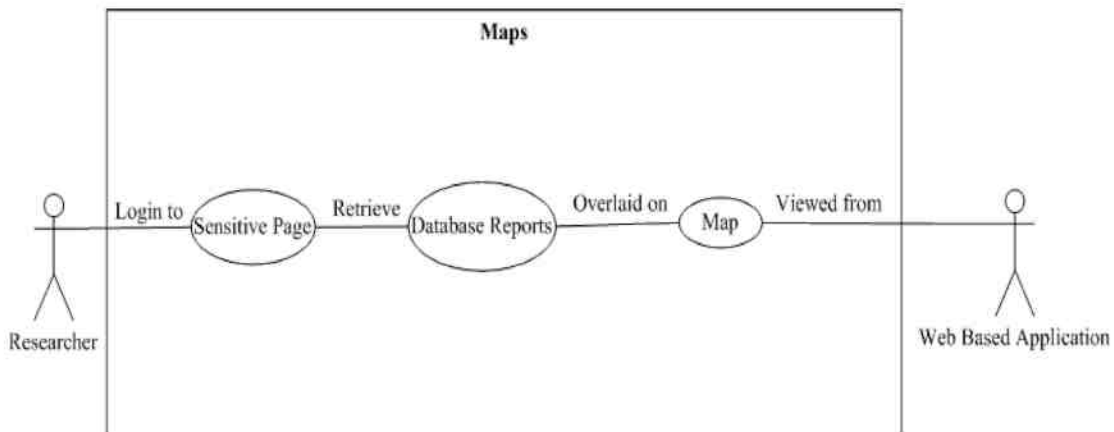


Figure 3.6 Displaying Data on a Map

Researchers need a way to visually display reports from the database. A map is utilized as an instrument to visually display information. A researcher has to login to the web-based application to access the map feature. All reports are retrieved from the

database and overlaid onto the map, as shown in Figure 3.6. The map can later be regenerated using custom filters for narrowing or expanding results that are overlaid. Each report overlaid on the map is shown to display the report type that was stored.

3.7 Refining Data on a Map Using Filters

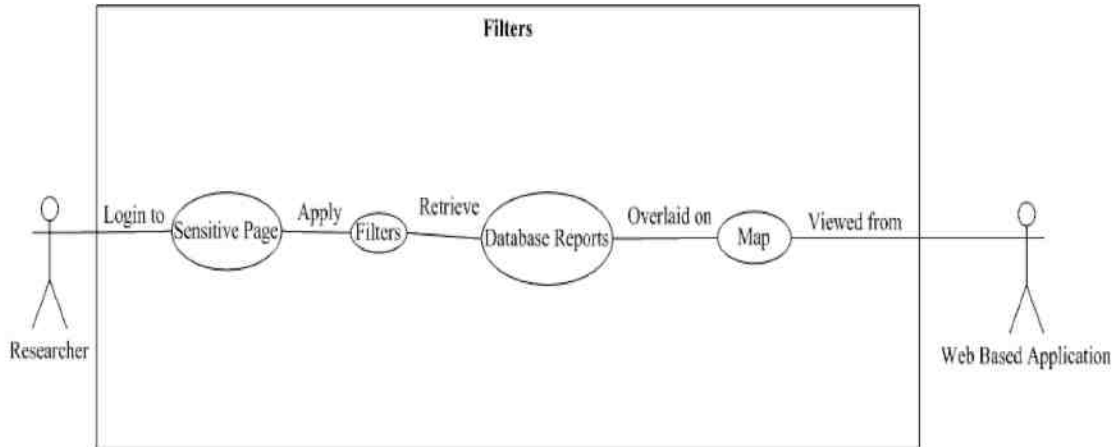


Figure 3.7 Refining Data on a Map Using Filters

Researchers need a way to refine data that is overlaid on the map. Filters are created to data mine and overlay reports on the map, as shown in Figure 3.7. A researcher has to login to the web-based application to access the map filters. Reports are retrieved from the database and overlaid on the map when the map page is generated. Researchers are able to utilize these filters to analyze a group of reports. Each balloon specifies which type of report was saved in the database. The two custom filters that are provided are to specify distance from the center of the map, and the other is to provide specific keywords that a user inputs. Filters are not applied to the map as an individual type, but have to be applied manually for effect to take place.

3.8 Submitting Report via Mobile Device

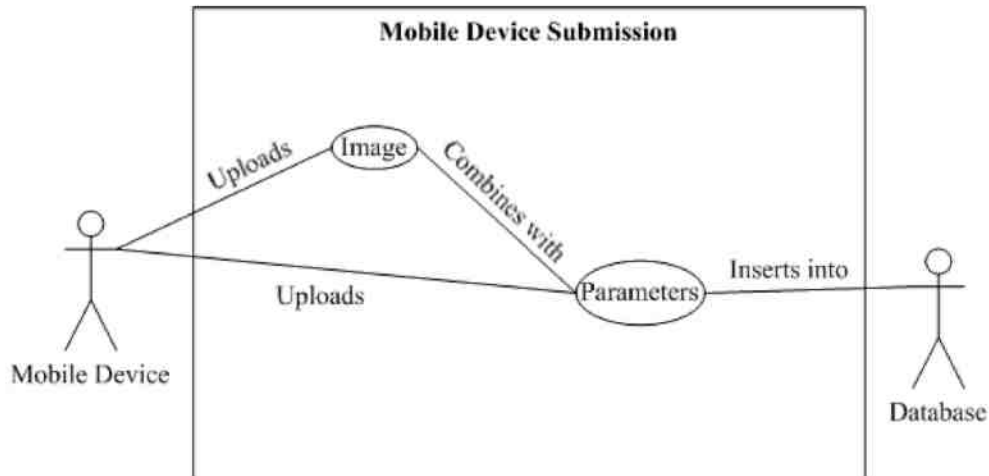


Figure 3.8 Submitting Report via Mobile Device

A mobile device is an instrument that data entry clerks can utilize to upload reports to a pool of data. A method of inserting a report is to upload an image combined with parameters to the database from the mobile device. Another method is to upload parameters without an image to the database from the mobile device, as shown in Figure 3.8. The report system is similar to the web-based applications report system, so that there is no confusion in data being stored. Methods for uploading reports are also similar to the uploading of reports on the web-based application in that an image is uploaded first, and then the remaining report gets uploaded to the database.

Chapter 4: Detailed Description of Design and Functionality of Web-Based Application

This chapter discusses the project as a whole in detail. This chapter reviews the tools employed for creating the web-based application, database, maps, and filters. An overview is given of what was specifically used to construct attributes using data types and an entity relationship diagram. The communication between mobile device and server is explained. Using data from the database, various maps are generated with the use of filters. This chapter is based on fabricated data to provide a general overview of the project.

4.1 Web-Based Application Tools Utilized



Figure 4.1 Web-Based Application Tools Utilized

Ubuntu is the operating system that was chosen. Ubuntu allows for the web-based application to execute scripts on the system and take action based on these scripts. Apache is the HTTP server that allows the web-based application to be visible and assembled on the Internet. MySQL is the database management system that executes queries, stores data, and retrieves data. ImageMagick is an image processing program for creating a swifter web page response; larger images are converted to smaller images. Zip is a compression program, compressing a set of images into a smaller single file size for exporting and downloading. Web browsers are used to view and interact with the web-based application. This software, shown in Figure 4.1, implements the server-side of the web-based application.

There are visual design tools that are implemented on the client-side for a more aesthetic look and a better user interaction. HTML5 [42] is the language used to construct the web-based application. CSS3 [34] is the language used to style the web page. PHP is

the language to dynamically generate HTML code and execute scripts on the server-side. JavaScript is the language that enhances user experience and interaction with the web-based application. AJAX is used to communicate between the client-side and the server-side without disrupting the user experience. XML 1.0 [23] is the language used to respond from server-side to client-side with information and directions on how to manipulate the web page. Google Maps JavaScript API is used to generate maps; with the use of filters the maps are customized to show data that pertain to the values in the filters. These tools, shown in Figure 4.1, are utilized to create a unique visual design for the web-based application.

4.2 Conceptual Database Design

A conceptual database design is used to determine a set of table with a list of relevant attributes under each. The relationship between tables is shown. There are seven tables: user, date/time, geospatial, kill, additional, transect, and road information. To stay within a standard form, the database engine follows InnoDB with charset UTF-8. There are numerous data types that an attribute can have to specify what can be stored within a table. A database entity set description provides a more in-depth look at each table (Appendix A, A.1). An E-R diagram shows the relationships between tables.

To construct all attributes there are a set of data types that are used, accompanied by their size in bytes.

Data Types

INT

Integer is a numeric type, having storage of 4 bytes and noted as INT (length). If the value of length is signed then the value is from -2147483648 to 2147483647. If the value of length is unsigned, then the value is from 0 to 4294967295.

DECIMAL

Decimal is a numeric type, having a storage of Total Digits + 4 bytes and noted as DECIMAL (Total Digits, Leftover Digits).

VARCHAR

Variable character is a varchar type, having storage of string length + 1 byte and noted as VARCHAR (length). The value of length is from 0 to 255.

TIME

Time is a datetime type, having storage of 3 bytes and noted as TIME.

DATE

Date is a datetime type, having storage of 3 bytes and noted as DATE.

TEXT

Text is a text type, having storage of 64 kilobytes and noted as TEXT.
The value of length is from 0 to 65535.

4.2.1 Illustration of an Entity-Relationship Diagram

Figure 4.2 illustrates an Entity-Relationship (E-R) diagram, expressing the relationships HasA between two tables. There are also multiplicity constraints of one-to-one (1:1) or one-to-many (1:*).

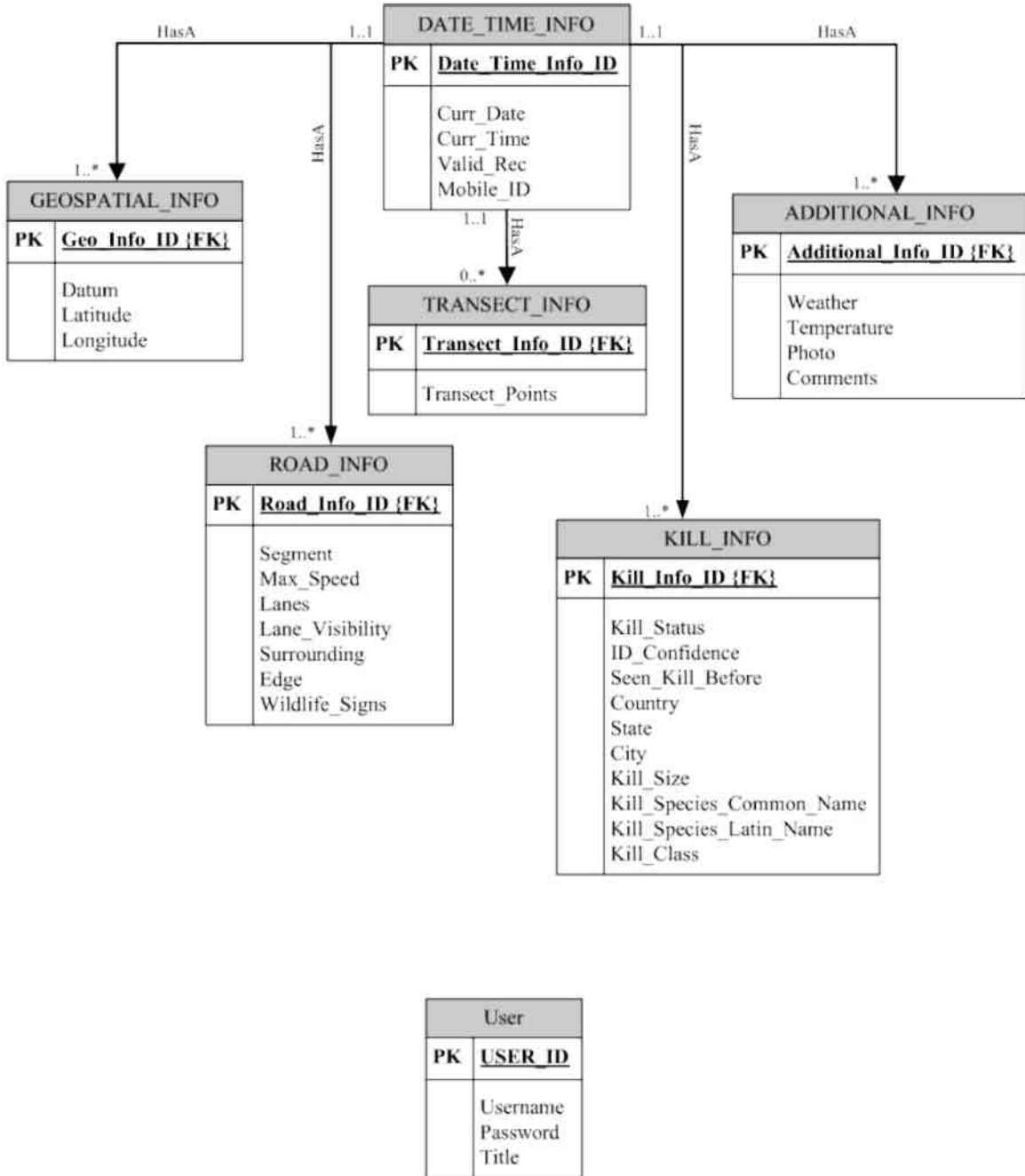


Figure 4.2 Illustration of an Entity-Relationship Diagram

4.3 General Overview of Public and Administrator Pages



Figure 4.3 Overview of Public Page

First, the public home page provides information about wildlife-vehicle collisions and whom to contact for further assistance. Second, the home page provides individuals the opportunity with an online system to contribute reports. Third, it also provides information of supported applications for mobile devices. Lastly, it provides a login system that allows for maintaining the web-based application, manipulating data within the database, and visually displaying data on a map, as shown in Figure 4.3.



Figure 4.4 Overview of Administrator Page

First, a search feature is provided to retrieve information from within the database. Second, the data entry clerk can submit reports. Third, a data entry clerk can import multiple records at once. Fourth, a researcher can export information from the database to do additional statistical analysis. Fifth, a researcher can data mine information to visually display on a map. Lastly, a logout system is provided to prevent non administrators from accessing secure information, as shown in Figure 4.4.

4.4 Login Page for Administrator



Figure 4.5 Login Page for Administrator

Figure 4.5 shows a typical administrator login with username and password as qualifications. Unwanted characters are escaped when a username and password are input as there are specific characters that are destructive to the database. The login, username, and password parameters are encoded and sent over an HTTP connection as a posted string to a PHP file.

The PHP file decodes the parameters login, username, and password on the server-side and escapes any unwanted special characters on both username and password. The login parameter is used as a flag to make sure that login was invoked. An SQL select statement is formed where username is injected and password is encrypted using SHA2 [56]; this statement is then executed in MySQL to locate user existence in the database.

If true, sessions are created for username, Admin is set to true, and username is posted. Also, an XML response is created for success containing the tags successid with a value of html tag name, successstr with a value of success message, and locstr with a value of the “admin/” directory; else an XML response is created for error containing the tags errorid with a value of html tag name and errorstr with a value of error message. The client-side parses the response and displays the success or error string and redirects the end user if found in the database to the administrator page.

4.5 Implementation of a Single Report



Figure 4.6 Implementation of a Single Report

Data entry clerks are able to submit reports on the public and the administrator pages through the single report link. The report consists of five categories: Date and

Time, Geospatial, Road, Kill, and Additional. The capability of changing units from “Imperial” or “SI” (International System) units is accomplished with two buttons above the report, as shown in Figure 4.6. Unwanted characters are escaped from each category as there are specific characters that are destructive to the database. The “submit” button is enabled for the report when all fields have valid inputs. All input parameters are encoded and sent over an HTTP connection as a posted string to a PHP file.

The PHP file decodes the parameters on the server-side and escapes any unwanted special characters on all parameters. The Submitdata parameter is used as a flag to make sure that report was invoked. Multiple SQL insert statements are formed for each category and then executed on the database.

If inserts are true, an XML response is created for success containing the tags successid with a value of html tag name and successstr with a value of success message; else an XML response is created for error containing the tags errorid with a value of html tag name and errorstr with a value of error message. The client-side parses the response and displays the success or error string next to the error input fields. The report can be done in any order. Transect is not permitted on the web-based application; therefore, an empty string is stored into the database for this value.

4.5.1 Inputting Date and Time Values



The screenshot shows a web application interface with a horizontal navigation bar at the top. The bar contains five buttons: 'DATE AND TIME' (white), 'GEOSPATIAL' (red), 'ROAD' (blue), 'KILL' (green), and 'ADDITIONAL' (orange). Below the navigation bar, the 'Date and Time' section is displayed. It features a title 'Date and Time' in a large, bold, black font. Underneath the title, there are two input fields. The first is labeled 'Date:' and contains the text '2012-05-21'. To its right, the format 'YYYY-MM-DD' is displayed. The second is labeled 'Time:' and contains the text '12:35:00'. To its right, the format 'HH:MM:SS' is displayed.

Figure 4.7 Inputting Date and Time Values

The Date and Time category allows the input of date and time, as shown in Figure 4.7. Date accepts the format of YYYY-MM-DD; a valid US calendar date is checked on the server-side when parameters are posted to the server. Time accepts the format of HH:MM:SS, a valid military standard time. Regular expressions validate input; if there are any errors, the message is displayed near the invalid input fields.

4.5.2 Inputting Geospatial Values

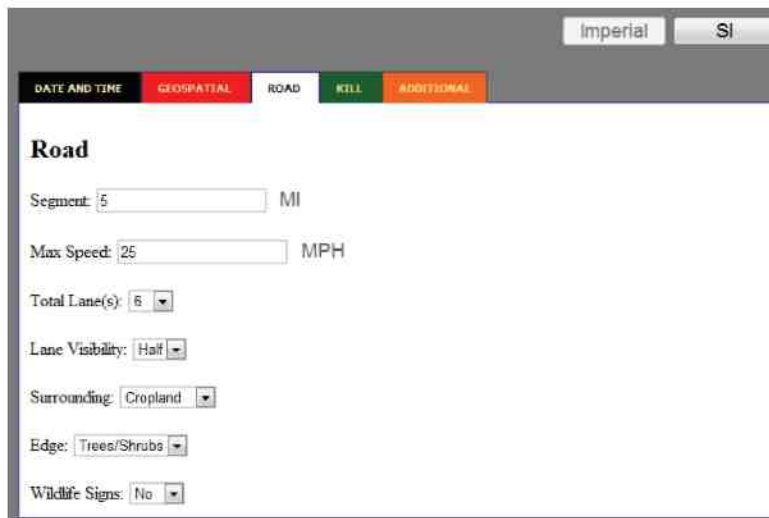


The screenshot shows a web interface with a navigation bar at the top containing five tabs: 'DATE AND TIME' (black), 'GEOSPATIAL' (white), 'ROAD' (blue), 'KILL' (green), and 'ADDITIONAL' (orange). The 'GEOSPATIAL' tab is selected. Below the navigation bar, the title 'Geospatial' is displayed. The form includes a 'Datum' dropdown menu set to 'Google'. Below this, the text 'Input in Decimal Degrees' is shown. There are two input fields: 'Latitude' with the value '45.123456' and a range of '-90.000000 to 90.000000 range', and 'Longitude' with the value '150.123456' and a range of '-180.000000 to 180.000000 range'.

Figure 4.8 Inputting Geospatial Values

The Geospatial category allows the input of datum, latitude, and longitude, as shown in Figure 4.8. Datum is a static list of values specifying how coordinates were achieved. Latitude accepts a value range between -90.000000 and 90.000000. Longitude accepts a value range between -180.000000 and 180.000000. Both latitude and longitude inputs are specified in decimal degrees. Regular expressions validate input; if there are any errors, the message is displayed near the invalid input fields.

4.5.3 Inputting Road Values



The screenshot shows a web interface with a navigation bar at the top containing five tabs: 'DATE AND TIME' (black), 'GEOSPATIAL' (red), 'ROAD' (white), 'KILL' (green), and 'ADDITIONAL' (orange). The 'ROAD' tab is selected. Below the navigation bar, the title 'Road' is displayed. The form includes several input fields and dropdown menus: 'Segment' (input field with value '5' and unit 'MI'), 'Max Speed' (input field with value '25' and unit 'MPH'), 'Total Lane(s)' (dropdown menu with value '6'), 'Lane Visibility' (dropdown menu with value 'Half'), 'Surrounding' (dropdown menu with value 'Cropland'), 'Edge' (dropdown menu with value 'Trees/Shrubs'), and 'Wildlife Signs' (dropdown menu with value 'No'). At the top right of the form, there are two buttons: 'Imperial' and 'SI'.

Figure 4.9 Inputting Road Values

The Road category allows the input of segment, maximal speed, total lanes, lane visibility, surrounding, edge, and wildlife signs, as shown in Figure 4.9. Segment accepts the format of whole numbers to the hundreds place or decimal to the tenths place, and measurement is set to either MI (Miles) or KM (Kilometer). Max Speed accepts the

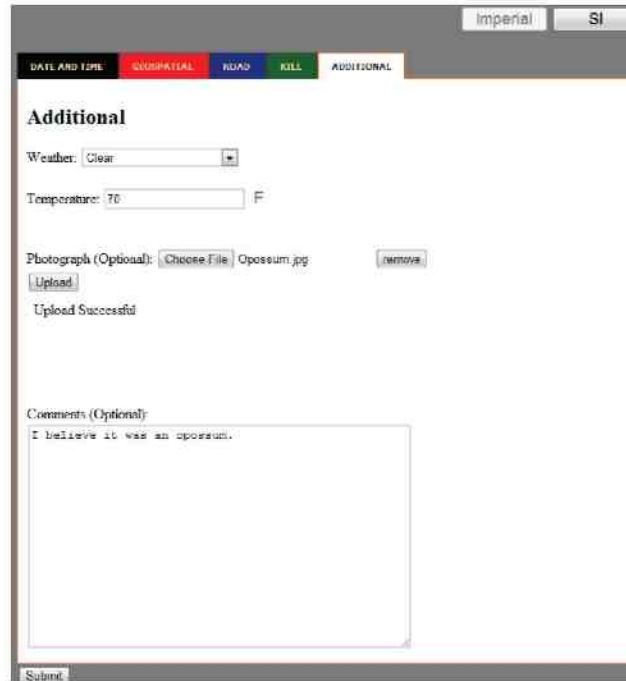
format of whole numbers to the hundreds place or decimal to the tenths place, and measurement is set to either MPH (Miles per Hour) or KM/H (Kilometer per Hour). Total lanes, lane visibility, surrounding, edge, wildlife signs are static predefined values. Regular expressions validate input; if there are any errors, the message is displayed near the invalid input fields.

4.5.4 Inputting Kill Values

Figure 4.10 Inputting Kill Values

The Kill category allows the input of status, identification confidence, seen before, country, state, city, size of kill, species common name, species Latin name, and class (classification), as shown in Figure 4.10. Country, state, city, species common name, species Latin name, and class accepts 255 characters including spaces, periods, single quotes, and hyphens. Status, identification confidence, seen before, size of kill are static predefined values. Regular expressions validate input; if there are any errors, the message is displayed near the invalid input fields.

4.5.5 Inputting Additional Values



The screenshot shows a web interface with a top navigation bar containing 'Imperial' and 'SI' units. Below this is a menu with 'DATE AND TIME', 'QUADRANTAL', 'ROAD', 'KILL', and 'ADDITIONAL'. The 'ADDITIONAL' section is active and contains the following fields: a 'Weather' dropdown menu set to 'Clear', a 'Temperature' input field with '70' and a unit selector set to 'F', a 'Photograph (Optional)' section with a 'Choose File' button, a file name 'Opossum.jpg', and a 'remove' button. Below the file selection is an 'Upload' button and a confirmation message 'Upload Successful'. At the bottom is a 'Comments (Optional)' text area containing the text 'I believe it was an opossum.' and a 'Submit' button.

Figure 4.11 Inputting Additional Values

The Additional category allows the input of weather, temperature, photograph, and comments, as shown in Figure 4.11. Weather is a set of static predefined values. Temperature accepts the format of whole numbers to the hundreds place, and temperature is set to either F (Fahrenheit) or C (Celsius). The “choose file” button allows the data entry clerk to locate an image file on their personal computer to upload to the server with extension of png, jpg, or jpeg; no other extension is supported. The image is posted to the server through HTTP using a PHP function that uploads the image to a temporary directory.

The PHP file checks the image from the temporary directory for file extension and size. The image file is renamed to a random number and is moved to a new specified temporary directory. If the image was successfully moved to the new temporary directory, then a hidden value is placed on the report page of the images’ filename. If the image was not moved to the new temporary directory or file size is 0, then a hidden value is placed on the report page of “Empty”. If an image extension is unsupported, an error message is displayed on the report page.

The “remove” button is used if the user has opted out of incorporating an image in the report which places a hidden value of “Empty”. The value of the filename or “Empty” is posted with the remainder of the report when submitted to the server.

The PHP file renames the image file to correspond with the report’s identification number and saves this string to the database. A thumbnail is created and saved to a

thumbs directory and the original image file is moved into a permanent directory. Mobile Safari does not support uploading of images, so this feature is disabled for this browser. Android devices support uploading of images, but is also disabled until all browsers support this feature. Comments accept 255 characters including spaces, periods, single quotes, and hyphens. Regular expressions validate input; if there are any errors, the message is displayed near the invalid input fields.

4.6 Validating, Editing, or Deleting Reports through Search

Date	Time	Datum	Latitude	Longitude	Segment	Max Speed	Lanes	Lane Visibility	Surrounding	Edge	Wildlife Signs	Kill Status	ID Confidence	Seen Kill Before	Country	State	City	Kill Size	Species Common Name	Species Latin Name	Kill Class	Weather	Temperature	Photo	Comments	Transit Points
2012-01-24	14:32:05	Google	34.178545	119.078590	2.3 mi	23.6 mph	7	All	Beach	Open	No	Yes	Best Guess	No	United States	California	Camarillo	Small	scouring above	Zenaidura	Avian	Showers	60 F			34.183952, -119.040251, 34.184721, -119.061819
2012-01-23	13:32:00	Google	34.177480	119.048507	1 mi	50 mph	6	All	Suburban	City	No	Yes	High	No	United States	California	Camarillo	Small	prairie dog	Cynomys	Mammal	Scattered	70 F			
2012-01-20	12:32:50	Google	34.165479	119.059308	3 km	45 km/h	2	All	Wildland	Elevated	No	No	Low	No	United States	California	Camarillo	Small	equine	Tamiasciurus	Mammal	Scattered	30 C			Scattered Snow
2012-01-21	11:32:40	Google	34.171941	119.073312	3 km	10 km/h	3	All	Crop/land	Open	Yes	Yes	Definite	No	United States	California	Oxnard	Medium	gopher snake	Pituophis	Reptilia	Scattered	36 C			Scattered Showers
2011-12-11	10:32:30	Google	34.171148	119.124639	3.5 mi	10 mph	7	Half	Orchards	Fence	Yes	Yes	High	No	United States	California	Oxnard	Medium	hatteriasuka	Emmalia	Reptilia	Rain and Snow	49 F			

Figure 4.12 Validating, Editing, or Deleting Reports through Search

Data entry clerks or researchers are able to utilize the search page to adjust data from the database. As each character is typed, results are shown making this a live search. First, data can be validated for accurate information. Second, data can be edited for inaccurate information. Third, data can be deleted for useless information. Lastly, paging is allowed to view all records that correspond to the inputted keyword, as shown in Figure 4.12.

4.6.1 Validating Reports in the Database



Figure 4.13 Search Legend

Each record contains a checkbox which permits the user to verify the record for valid information, as shown in Figure 4.13. The primary id number parameter is encoded and sent over an HTTP connection as a posted string to a PHP file.

The PHP file decodes the parameter on the server-side and escapes any unwanted special characters on the parameter. First, the record id number is verified for existence in the database; if the record exists the process continues to the next step; else the page is refreshed. Second, the record is checked for validation; if not valid the process continues

to the next step; else the page is refreshed. Lastly, the record is validated; if the record cannot be validated then an error is reported.

If the update value is true, an XML response is constructed for success containing the tags `successid` with a value of `html` tag name and `successstr` with a value of success message; else an XML response is constructed for error containing the tags `errorid` with a value of `html` tag name and `errorstr` with a value of error message. The client-side parses the response and displays a green checkmark on success next to the record or alerts an error message.

4.6.2 Editing Reports in the Database



The screenshot shows a web-based form titled 'Additional' with a tabbed interface. The tabs are 'DATE AND TIME', 'GEOSPATIAL', 'ROAD', 'KILL', and 'ADDITIONAL', with 'ADDITIONAL' being the active tab. The form contains the following fields and controls:

- Weather: Clear
- Temperature: 65 F
- Photograph (Optional): A small image of a bird in flight.
- Remove Image:
- Comments (Optional): A text area containing the text 'On pch 2.9 miles north of The Abalone Farm'.
- Validated: No
- Validate this record:
- Buttons: 'Update' and 'Cancel'.

Figure 4.14 Editing Reports in the Database

Each record contains an edit link which permits the user to edit the record, as shown in Figure 4.13. Upon clicking, the data entry clerk is taken to a single report. A hidden number is transferred from the search page to the edit page, so that the correct record can be updated. This report retrieves data from within the database and inserts the values into their respective fields, as shown in Figure 4.14.

The data entry clerk is not capable of modifying fields in Date and Time, Geospatial, and Road categories. Transect information is not displayed and is not modifiable when editing. Kill has four fields that are not enabled for modifications; the remaining fields are adjustable for better clarification. Additional has two fields which are not enabled for modification; the remaining fields are adjustable for better

clarification. Also, an option to validate the current record is enabled if not already validated; if validated, then “yes” is displayed in the validated section. There are two buttons, one to “update” modifications and another to “cancel” and return to the search page.

All input parameters are encoded and sent over an HTTP connection as a posted string to a PHP file. The PHP file decodes the parameters on the server-side and escapes any unwanted special characters on all parameters. Updatedata parameter is used as a flag to make sure that update was invoked. An SQL update statement is formed and then executed on the database.

If the update value is true, an XML response is constructed for success containing the tags successid with a value of html tag name and successstr with a value of success message; else an XML response is constructed for error containing the tags errorid with a value of html tag name and errorstr with a value of error message. The client-side parses the response and displays the success or error string next to the error input fields.

4.6.3 Omitting Reports in the Database

Each record has a delete link which permits the user to remove records from the database, as shown in Figure 4.13. When the delete link is clicked a confirmation window appears allowing the data entry clerk to confirm or deny the deletion. If confirmed to remove, the primary id parameter is encoded and sent over an HTTP connection as a posted string to a PHP file.

The PHP file decodes the parameter on the server-side and escapes any unwanted special characters on the parameter. The record id number is verified for existence in the database; if the record exists, it is omitted; else an alert message is displayed. An SQL delete statement is formed and then executed on the database.

If delete is true, an XML response is constructed for success containing the tags successid with a value of html tag name and successstr with a value of success message; else an XML response is constructed for error containing the tags errorid with a value of html tag name and errorstr with a value of error message. The client-side parses the response and alerts a successful deletion or an error message.

4.7 Importing and Exporting Reports

Import is used to upload a large number of reports into the database without having to upload each report one by one. Export is used to download reports and images from the database for backup purposes or for additional statistical analysis.

4.7.1 Importing Reports via a CSV File



Figure 4.15 Importing Reports via a CSV File

To accompany the import feature, there is a section for template files to use, as shown in Figure 4.15. If Mac operating system is detected, three different files appear: Numbers, Excel, and ODS. If Windows operating system is detected, two different files appear: Excel and ODS. If *nix (Linux or UNIX) operating system is detected, then an ODS file appears. Each file is a template for data entry clerks to use offline that is tailored to their operating system, giving them an opportunity to import multiple records at once. Instructions to create CSV files are displayed when a template file is downloaded. Transect points are not permitted in the CSV file as this feature is not supported throughout the web-based application. CSV files do not support images, so photos are set to empty.

The “Choose File” button allows the data entry clerk to locate a file on their personal computer to upload to the server with extension CSV; no other extension is supported. The file is posted to the server through HTTP using a PHP function that stores the file to a temporary directory.

The PHP file checks the file from the temporary directory for file extension and size. The file is then moved to a new temporary directory. If the file was successfully moved to the temporary directory, then parameters are set for `fgetcsv` with a max length of 2000 for the longest line, delimiter of comma which separates each field, and an enclosure of double quotes which specifies what values are in each field.

First, a CSV file is opened and read starting on line three. Second, each field is validated for specific values. If any values are incorrectly inserted, the next step is set to false, the loop is stopped, and the file is closed. After closing the file, an error message is displayed noting that one or more fields have errors. If next step is set to true, the file is reopened with the same parameters for `fgetcsv` function and the CSV file is read again starting on line three. The file is closed when the end of file is reached. Lastly, the database checks for the current record. If the record is found, the next line is processed

skipping the found record. If the record is not found, then multiple SQL insert statements are constructed for each category and then executed on the database. Error messages are displayed if records could not be inserted, all records already exist in the database, CSV file is empty, or file could not be opened. A success message is displayed if records were successfully inserted.

4.7.2 Exporting Reports from the Web-Based Application



Figure 4.16 Exporting Reports from the Web-Based Application

Instructions on creating a Numbers file, an Excel file, or an ODS file is displayed, as shown in Figure 4.16. Exportdata parameter is used as a flag to make sure that export was invoked when the “Generate File” button is pressed. The export parameter is sent over an HTTP connection as a posted string to a PHP file.

An SQL view table is generated with all reports from the database. If the view table already exists, the view table is dropped and regenerated. All field names and data are selected from the view table. Double quotes enclose field names and data, and a comma separates field data as a delimiter. A CSV file is created and opened and each record is transcribed as a single line in the file. When there are no more reports to write, the file is closed. A zip file is generated of all images with extensions png, jpg, and jpeg. These two files are stored in an export directory for downloading.

If a CSV file was created, an XML response is constructed for success containing the tags successid with a value of html tag name and successstr with a value of success message; or else an XML response is constructed for errors containing the tags errorid with a value of html tag name and errorstr with a value of error message. The client-side parses the response and displays links upon success or error message.

4.8 Retrieving Reports from Mobile Device



Figure 4.17 Retrieving Reports from Mobile Device

Data entry clerks on the mobile device are able to upload reports, as shown in Figure 4.17. The report consists of four categories: Time/Place, Road, Kill, and Additional. Also, transect points are permitted as a parameter when uploading the report. All input parameters are encoded and sent over an HTTP connection as a posted string to a PHP file.

The PHP file decodes the parameters on the server-side and escapes any unwanted special characters on all parameters. Submitdata parameter is used as a flag to make sure that submit was invoked. Multiple SQL insert statements are formed for each category and then executed on the database. If inserts are true, an XML response is constructed for success containing tag name `insertsuccess` with a data value of success message; else an XML response is constructed for an error containing tag name `inserterror` with a data value of error message. `inserterror` contains various nested tag names for errors in different fields on the mobile device. The client-side parses the response and processes successful reports or displays an error message next to the invalid fields. A description of the Mobile API is provided, so that alternative mobile devices could submit reports to the server (Appendix C).

4.9 Mechanisms for Visually Displaying Reports on a Map

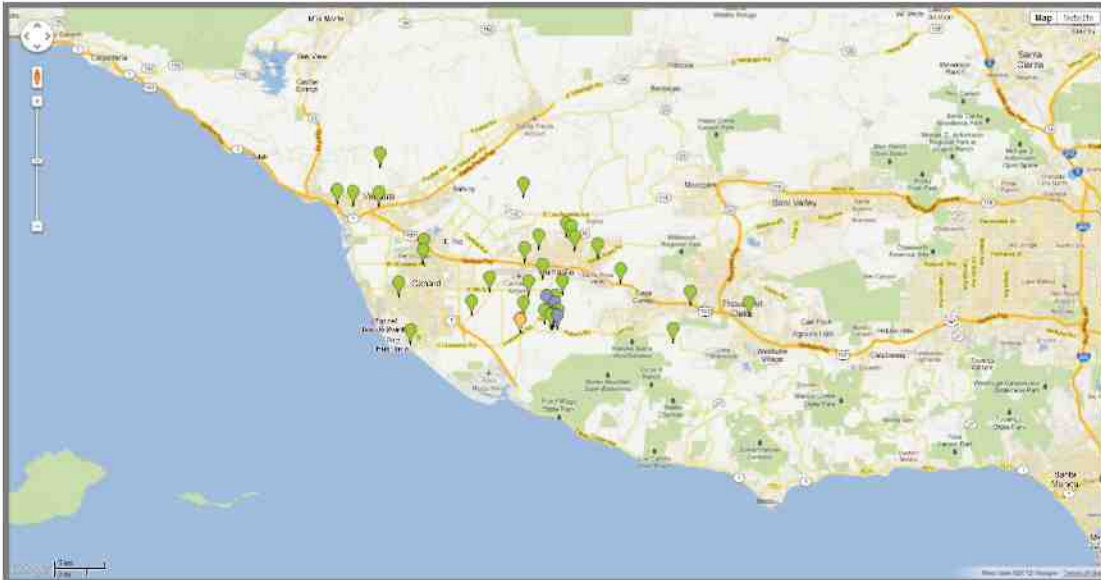


Figure 4.18 Visually Displaying Reports on a Map

A map API is used to generate maps with overlays, routes, and report information. The map allows for zooming in and out, showing different terrains, and street views. Geo location is asked when rendering the page; if allowed, the map is centered at the researcher’s current location (latitude and longitude). If not allowed, the map is centered at a fixed location (latitude and longitude), as shown in Figure 4.18.



Figure 4.19 Informing that there are no Reports found

An SQL view table is generated using the parameters search, distance, latitude, and longitude. If the view table already exists, the view table is dropped and regenerated using the same parameters. The PHP file on the server-side escapes any unwanted special characters from search, distance, latitude, and longitude parameters. An SQL select statement is formed and then executed on the database. If the query returns reports, an array is produced using the values from each field in the view table. Using this array a map is generated dropping markers as an overlay according to the latitude and longitude points, as shown in Figure 4.18. If the array is empty, the map is hidden and a display of “no locations found” is shown, as shown in Figure 4.19.



Figure 4.20 Differentiating Report Markers

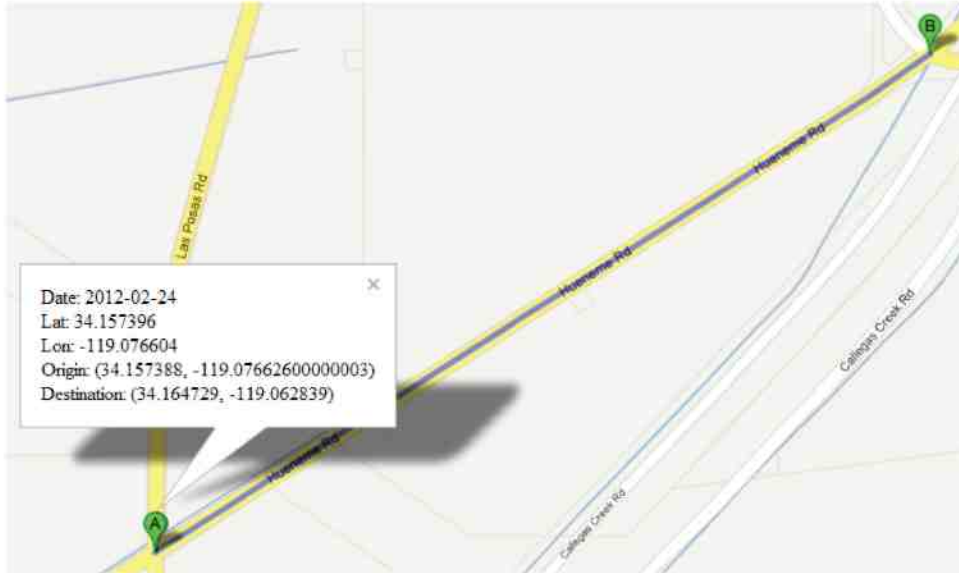


Figure 4.21 Map Containing a Transect without a Report

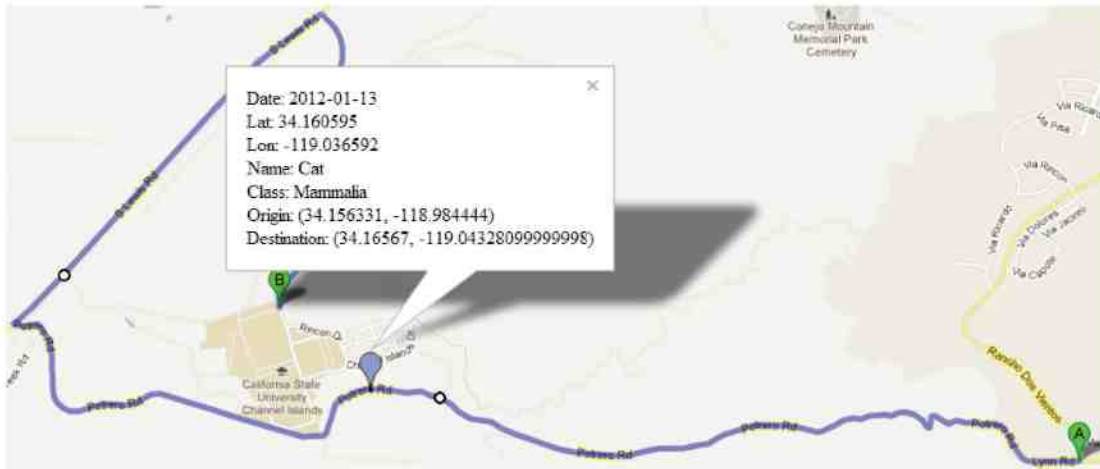


Figure 4.22 Map Containing a Transect with a Single Report

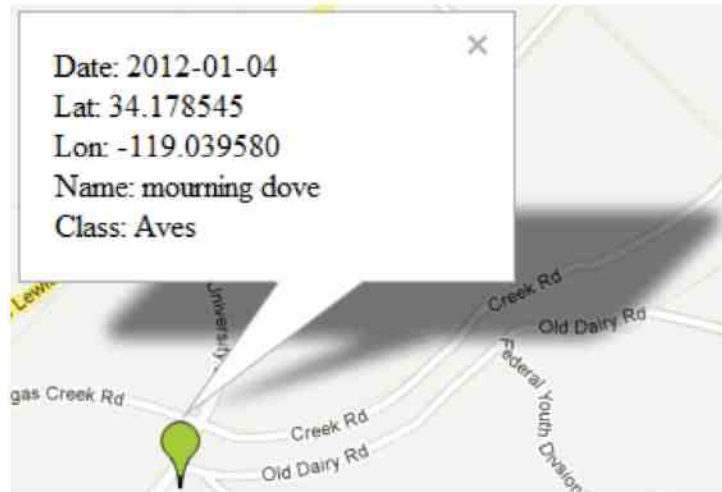


Figure 4.23 Map Containing a Single Report without Transect

There are three different colored markers that are shown on the map, as shown in Figure 4.20 which display different information in a window. Orange signifies transect values without reports. Blue signifies transect values with reports. Green signifies reports with no transect values. Attaining an orange marker is done by checking the transect field in the array for not empty; if not empty the weather field is checked for “none”. The information window displays date, latitude, longitude, origin, and destination, as shown in Figure 4.21. Attaining a blue marker is done by checking the transect field in the array for not empty; if not empty, the weather field is not checked. The information window displays date, latitude, longitude, name, classification, origin, and destination, as shown in Figure 4.22. Attaining a green marker is done by checking the transect field in the array for empty. The information window displays date, latitude, longitude, name, and classification, as shown in Figure 4.23.

To display information and a route (if any) of a specific report, an event listener is triggered when a marker is clicked. Attaining a route is done by checking the transect field in the array for not empty. If not empty, two arrays are created to hold latitude and longitude based on the comma delimiter. The first latitude and longitude are set as origin and the last latitude and longitude are set as destination; the remaining points in between are considered as waypoints. Each waypoint on the map is represented as a white circle, as shown in Figure 4.22. A request is generated for displaying the driving route with default parameters in the settings. The directions server object is called to overlay route on the map, as shown in Figure 4.22, or errors are alerted if any occurred. If the transect field is empty, a route is not overlaid on the map, as show in Figure 4.23.

4.10 Generating Maps with Filters



Figure 4.24 Filters that are applied to the Map

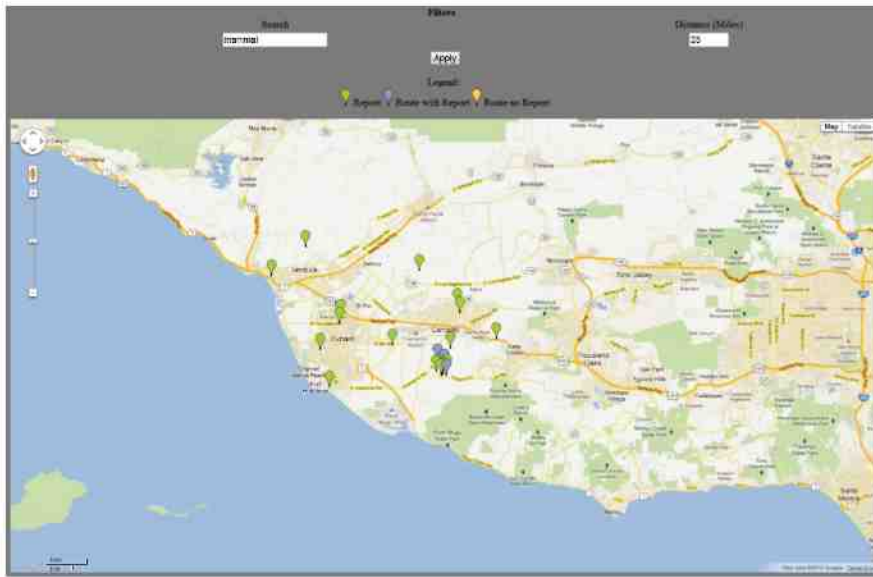


Figure 4.25 Generated Map with Search Keyword and Distance Applied

Currently, search and distance (measured in miles) are two filters that assist researchers in visually displaying information on a map, as shown in Figure 4.24. Search processes a keyword that researchers input to query the database for particular reports. Distance can also be input by the researcher to query the database for a certain distance from the center of the map. When these filters are applied, the parameters of search, distance, current latitude, and current longitude are sent over an HTTP connection as a posted string to a PHP file. Parameters are processed using the same steps as generating and dropping markers on a map as mentioned above. If additional data mining filters are created, a description of the filters API (Appendix B) provides a way of implementing the additional filter onto the web-based application. Figure 4.25 shows an example of a search query with search and distance input.

Chapter 5: Conclusion

There were two goals that were achieved in this project. First, a repository was created for collecting data related to road kill. Second, a web-based application was developed which allows data entry, database management, and creation of tools for filtering and visualization of information. Data entry was achieved by creating a reporting system for data entry clerks accessible through the Internet, which accepts values based on wildlife-vehicle collisions. Database management was achieved in allowing administrators and researchers to retrieve reports through searching. Based on the information in the reports, either the report is updated for inaccuracy or omitted. A visualization tool was created allowing researchers to analyze and visually display reports that are overlaid onto a map. A filtering tool was implemented to allow researchers to data mine information; the refined information becomes overlaid based on specific keywords and/or with distance from a given point on a map. An additional feature that was produced for this project was to allow crowdsourcing road kill information through the use of an iPhone device, which allows submitting reports to the server using an API. This project could be transplanted to other similar systems that would benefit from a crowd-sourced database. For example, a botanist or taxonomist can apply such a crowdsourcing database to categorize plants and animals.

Chapter 6: Future Work

There are other implementations that could be done if a developer chooses to further pursue this project. The public portion of this project could be more robust, both in client-side and server-side. An administrator could allow extra features that permit data entry clerks to interact with the data. The reporting system could be improved to allow enhanced interactions while documenting reports and permit more devices to report. Some enhancements could be made through testing the application for various situations.

The public portion of this project could be more robust in that the server could be moved onto personal hardware or a cloud system. The web-based application code could be modified to meet HTML5 standards throughout as more browsers support the new standard. In improving such features, both the server-side and the client-side should become more responsive and stable with user interaction.

The administrator portion of this project could allow extra features that permit data entry clerks to interact with the data such as: user registration, enhanced export, an alternate mapping system, and data mining filters. In allowing users to register, they will be able to track their reports as they submit them to the database. In producing an enhanced export feature, a data entry clerk will be able to export reports and images based on specific keywords. Editing reports could be enhanced in that an administrator could upload alternative images, therefore improving the report. The mapping system could be changed to OpenStreetMaps, so that there could be fewer restrictions and more modifications towards accessing map data. More filters could be constructed in that different data mining techniques could be applied, thus providing various data results to be overlaid onto a map.

The reporting system could be improved to allow enhanced interactions while documenting reports and permitting more devices to report. The reporting system could provide auto filling/completing of the remaining fields based on coordinates (latitude and longitude), animal common name, or animal Latin name. Both auto fill/complete could be achieved by retrieving information from a file held on the server, specified by the administrator. This file should hold information such as animal common names, Latin names, or classifications. Also, a data entry clerk could be permitted to input various datum values, therefore providing alternative coordinate. The report could allow data entry clerks to upload Transect reports on the web-based application instead of only through the mobile device. To provide a better communication protocol, JSON could be used instead of XML as there is more support provided for JSON. The client-side language could be changed to jQuery instead of JavaScript in that there are more enhanced client-side features. The protocol for sending information could be changed to HTTPS instead of HTTP, therefore securing the communication protocol for sending sensitive data at login or while uploading a report. Lastly, there could be more mobile devices allowed to upload reports such as Android and Windows Phones.

Some enhancements could be made by testing the application in various situations. Beta testers providing feedback or suggestions could improve various features of the web-based application. Other enhancements could be based on stress testing the web-based application, server, and the database for user activity, multiple report uploads, and multiple report insertions. Scaling could be implemented based on the stress test results for the web-based application, server, or database to meet user demands.

References

1. Apache Software Foundation, The. (2012, April). *Apache*. Retrieved from Apache website <http://httpd.apache.org/>.
2. Apache Software Foundation, The. (2012, July). *Apache Tomcat*. Retrieved from Apache website <http://tomcat.apache.org/>.
3. Apple Inc. (2012, May). *OS X Lion Server*. Retrieved from <http://www.apple.com/macosx/server/>.
4. Canonical Ltd. (2012, April). *Ubuntu*. Retrieved from Ubuntu website <http://www.ubuntu.com/>.
5. CROS. (n.d.). *California Roadkill Observation System*. Retrieved from <http://www.wildlifecrossing.net/california/>.
6. CSUCI. (n.d.). *Roadkill*. Retrieved from CSUCI website <http://roadkill.csuci.edu>.
7. Google, Inc. (2012, March). *Google Maps JavaScript API v3*. Retrieved from <http://code.google.com/apis/maps/documentation/javascript/>.
8. Google, Inc. (n.d.). *Google Sites*. Retrieved from Google website <http://www.sites.google.com>.
9. IETF. (1999, June). *Hypertext Transfer Protocol -- HTTP/1.1*. Retrieved from <http://datatracker.ietf.org/doc/rfc2616/>.
10. IETF. (2004, October). *The Common Gateway Interface (CGI) Version 1.1*. Retrieved from <http://datatracker.ietf.org/doc/rfc3875/>.
11. IETF. (2003, November). *UTF-8, a transformation format of ISO 10646*. Retrieved from <http://datatracker.ietf.org/doc/rfc3629/>.
12. ImageMagick Studio LLC. (2012, July). *Image Magick*. Retrieved from Image Magick website <http://www.imagemagick.org/script/index.php>.
13. jQuery Foundation, The. (2012, March). *jQuery*. Retrieved from jQuery website <http://jquery.com/>.
14. Kolowski, J., & Nielsen C. (2008). Using penrose distance to identify potential risk of wildlife–vehicle collisions. *Biological Conservation*, 141, 1119-1128.

15. Kumar, V., Steinbach, M., & Tan P. (2005). *Introduction to data mining*. New York: Addison-Wesley Longman, Incorporated.
16. Microsoft Corporation. (2009, October). *IIS*. Retrieved from Microsoft website <http://www.iis.net/>.
17. Microsoft. (2011, February). *Windows Server 2008 R2*. Retrieved from <http://www.microsoft.com/en-us/server-cloud/windows-server/default.aspx>.
18. Ojeda, G. (2012). Reporting System for road kill utilizing mobile devices.
19. OpenStreetMap. (2004, July). *Main Page*. Retrieved from OpenStreetMap website http://wiki.openstreetmap.org/wiki/Main_Page.
20. Oracle. (2012, July). *MySQL*. Retrieved from MySQL website <http://www.mysql.com/>.
21. PHP Group, The, (2012, July). *PHP*. Retrieved from PHP website <http://www.php.net/>.
22. W3Schools. (n.d.). *JSON Tutorial*. Retrieved from <http://w3schools.com/JSON/>.
23. WC3. (1994, October). *Extensible Markup Language (XML) 1.0 (Fifth Edition)*. Retrieved from <http://www.w3.org/TR/2008/REC-xml-20081126/>.
24. Weebly, Inc. (n.d.). *Weebly*. Retrieved from Weebly website <http://www.weebly.com/>.
25. Wikipedia. (2012, July). *Adobe Flash*. Retrieved from http://en.wikipedia.org/wiki/Adobe_Flash.
26. Wikipedia. (2012, July). *Ajax (programming)*. Retrieved from [http://en.wikipedia.org/wiki/Ajax_\(programming\)](http://en.wikipedia.org/wiki/Ajax_(programming)).
27. Wikipedia. (2012, July). *Applet*. Retrieved from <http://en.wikipedia.org/wiki/Applet>.
28. Wikipedia. (2012, July). *ASP.NET*. Retrieved from <http://en.wikipedia.org/wiki/ASP.NET>.
29. Wikipedia. (2012, July). *Cascading Style Sheets*. Retrieved from http://en.wikipedia.org/wiki/Cascading_Style_Sheets.
30. Wikipedia. (2012, July). *ColdFusion*. Retrieved from <http://en.wikipedia.org/wiki/ColdFusion>.

31. Wikipedia. (2012, July). *Common Gateway Interface*. Retrieved from http://en.wikipedia.org/wiki/Common_Gateway_Interface.
32. Wikipedia. (2012, July). *Command-line interface*. Retrieved from http://en.wikipedia.org/wiki/Command-line_interface.
33. Wikipedia. (2012, July). *Cron*. Retrieved from <http://en.wikipedia.org/wiki/Cron>.
34. Wikipedia. (2012, July). *CSS3*. Retrieved from http://en.wikipedia.org/wiki/Cascading_Style_Sheets#CSS_3.
35. Wikipedia. (2012, July). *Data mining*. Retrieved from http://en.wikipedia.org/wiki/Data_mining.
36. Wikipedia. (2012, July). *Database management system*." Retrieved from http://en.wikipedia.org/wiki/Database_management_system.
37. Wikipedia. (2012, July). *Document Object Model*. Retrieved from http://en.wikipedia.org/wiki/Document_Object_Model.
38. Wikipedia. (2012, May). *Filter (higher-order function)*. Retrieved from [http://en.wikipedia.org/wiki/Filter_\(higher-order_function\)](http://en.wikipedia.org/wiki/Filter_(higher-order_function)).
39. Wikipedia. (2012, July). *Graphical user interface*. Retrieved from http://en.wikipedia.org/wiki/Graphical_user_interface.
40. Wikipedia. (2012, June). *Haversine formula*. Retrieved from http://en.wikipedia.org/wiki/Haversine_formula.
41. Wikipedia. (2012, July). *HTML*. Retrieved from <http://en.wikipedia.org/wiki/HTML>.
42. Wikipedia. (2012, July). *HTML5*. Retrieved from <http://en.wikipedia.org/wiki/HTML5>.
43. Wikipedia. (2012, July). *HTTP Secure*. Retrieved from http://en.wikipedia.org/wiki/HTTP_Secure.
44. Wikipedia. (2012, July). *Hypertext Transfer Protocol*. Retrieved from http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol.
45. Wikipedia. (2012, July). *JavaScript*. Retrieved from <http://en.wikipedia.org/wiki/JavaScript>.
46. Wikipedia. (2012, June). *JavaServer Pages*. Retrieved from http://en.wikipedia.org/wiki/JavaServer_Pages.

47. Wikipedia. (2012, June). *JQuery*. Retrieved from <http://en.wikipedia.org/wiki/JQuery>.
48. Wikipedia. (2012, July). *JSON*. Retrieved from <http://en.wikipedia.org/wiki/JSON>.
49. Wikipedia. (2012, July). *MySQL*. Retrieved from <http://en.wikipedia.org/wiki/MySQL>.
50. Wikipedia. (2012, July). *Object-oriented programming*. Retrieved from http://en.wikipedia.org/wiki/Object-oriented_programming.
51. Wikipedia. (2012, July). *OpenStreetMap*. Retrieved from <http://en.wikipedia.org/wiki/OpenStreetMap>.
52. Wikipedia. (2012, July). *Operating system*. Retrieved from http://en.wikipedia.org/wiki/Operating_system.
53. Wikipedia. (2012, July). *Oracle Database*. Retrieved from http://en.wikipedia.org/wiki/Oracle_Database.
54. Wikipedia. (2012, July). *Relational database management system*. Retrieved from http://en.wikipedia.org/wiki/Relational_database_management_system.
55. Wikipedia. (2012, July). *Server (computing)*. Retrieved from [http://en.wikipedia.org/wiki/Server_\(computing\)](http://en.wikipedia.org/wiki/Server_(computing)).
56. Wikipedia. (2012, July). *SHA-2*. Retrieved from <http://en.wikipedia.org/wiki/SHA-2>.
57. Wikipedia. (2012, July). *SQLite*. Retrieved from <http://en.wikipedia.org/wiki/SQLite>.
58. Wikipedia. (2012, July). *UTF-8*. Retrieved from <http://en.wikipedia.org/wiki/UTF-8>.
59. Wikipedia. (2012, July). *XML*. Retrieved from <http://en.wikipedia.org/wiki/XML>.
60. Wikipedia. (2012, July). *Web browser*. Retrieved from http://en.wikipedia.org/wiki/Web_browser.
61. Wikipedia. (2012, July). *XUL*. Retrieved from <http://en.wikipedia.org/wiki/XUL>.
62. Wikipedia. (2012, July). *YAML*. Retrieved from <http://en.wikipedia.org/wiki/YAML>.

Appendix A: Database Entity Set Descriptions

This appendix discusses the different entity tables in the database and the attributes that comprise the tables.

A.1 Entity: User

1. Description: The user table holds information about users. If given permission to access the raw data, then a user or the administrator can use a username and password to login to a secure page.

2. Attributes:

1. Name: USER_ID
2. Description: This holds the identification number of each user.
3. Domain/Type: integer
4. Value-Range: 0 – 9999999999
5. Signed, Unsigned, None: Unsigned
6. Default value: ""
7. Null Value Allowed or Not: Not
8. Unique: Yes
9. Single or Multiple-Value: Single
10. Simple or Composite: Simple

1. Name: Username
2. Description: This holds the username of a user.
3. Domain/Type: varchar
4. Value-Range: 8 – 20
5. Signed, Unsigned, None: None
6. Default value: "Username"
7. Null Value Allowed or Not: Not
8. Unique: Yes
9. Single or Multiple-Value: Single
10. Simple or Composite: Simple

1. Name: Password
2. Description: This holds the password of a user.
3. Domain/Type: varchar
4. Value-Range: 8 – 50
5. Signed, Unsigned, None: None
6. Default value: "Password"
7. Null Value Allowed or Not: Not

8. Unique: No
9. Single or Multiple-Value: Single
10. Simple or Composite: Simple

1. Name: Title
2. Description: Specify if the user is Admin or a public user.
3. Domain/Type: varchar
4. Value-Range: 5 – 13
5. Signed, Unsigned, None: None
6. Default value: “Public”
7. Null Value Allowed or Not: Not
8. Unique: No
9. Single or Multiple-Value: Single
10. Simple or Composite: Simple

3. Candidate Key: User_ID
4. Primary Key: User_ID
5. Strong/Weak Entity: Strong
6. Field to be indexed: None
7. Additional Comments: There are no connections with other tables in this database.

A.2 Entity: Date Time Info

1. Description: The Date Time Information table keeps records of a calendar date and time.

2. Attributes:

1. Name: Date_Time_Info_ID
2. Description: This holds the identification number of each survey.
3. Domain/Type: integer
4. Value-Range: 0 – 9999999999
5. Signed, Unsigned, None: Unsigned
6. Default value: “”
7. Null Value Allowed or Not: Not
8. Unique: Yes
9. Single or Multiple-Value: Single
10. Simple or Composite: Simple

1. Name: Curr_Date
2. Description: Specify the date of observation.
3. Domain/Type: date
4. Value-Range: 10
5. Signed, Unsigned, None: None

6. Default value: "0000-00-00"
7. Null Value Allowed or Not: Not
8. Unique: No
9. Single or Multiple-Value: Single
10. Simple or Composite: Simple

1. Name: Curr_Time
2. Description: Specify the time of observation.
3. Domain/Type: time
4. Value-Range: 8
5. Signed, Unsigned, None: None
6. Default value: "00:00:00"
7. Null Value Allowed or Not: Not
8. Unique: No
9. Single or Multiple-Value: Single
10. Simple or Composite: Simple

1. Name: Valid_Rec
2. Description: Specify if the record has been validated.
3. Domain/Type: varchar
4. Value-Range: 2 – 3
5. Signed, Unsigned, None: None
6. Default value: "No"
7. Null Value Allowed or Not: Not
8. Unique: No
9. Single or Multiple-Value: Single
10. Simple or Composite: Simple

1. Name: Mobile_ID
2. Description: Generate an id for a mobile device.
3. Domain/Type: varchar
4. Value-Range: 0 – 50
5. Signed, Unsigned, None: None
6. Default value: ""
7. Null Value Allowed or Not: Allowed
8. Unique: Yes
9. Single or Multiple-Value: Single
10. Simple or Composite: Simple

3. Candidate Key: Date_Time_Info_ID
4. Primary Key: Date_Time_Info_ID
5. Strong/Weak Entity: Strong
6. Field to be indexed: None

A.3 Entity: Geospatial Info

1. Description: The Geospatial Information table keeps records of the location based on latitude and longitude.

2. Attributes:

1. Name: Geo_Info_ID
 2. Description: This holds the identification number of each survey.
 3. Domain/Type: integer
 4. Value-Range: 0 – 9999999999
 5. Signed, Unsigned, None: Unsigned
 6. Default value: ""
 7. Null Value Allowed or Not: Not
 8. Unique: Yes
 9. Single or Multiple-Value: Single
 10. Simple or Composite: Simple
-
1. Name: Datum
 2. Description: Specify datum of decimal degrees of coordinates.
 3. Domain/Type: varchar
 4. Value-Range: 5 – 255
 5. Signed, Unsigned, None: Signed
 6. Default value: "180.000000"
 7. Null Value Allowed or Not: Not
 8. Unique: No
 9. Single or Multiple-Value: Single
 10. Simple or Composite: Simple
-
1. Name: Latitude
 2. Description: Specify latitude decimal degree location of observation.
 3. Domain/Type: decimal
 4. Value-Range: (8, 6)
 5. Signed, Unsigned, None: Signed
 6. Default value: "90.000000"
 7. Null Value Allowed or Not: Not
 8. Unique: No
 9. Single or Multiple-Value: Single
 10. Simple or Composite: Simple
-
1. Name: Longitude
 2. Description: Specify longitude decimal degree location of observation.
 3. Domain/Type: decimal
 4. Value-Range: (9, 6)
 5. Signed, Unsigned, None: Signed
 6. Default value: "180.000000"

7. Null Value Allowed or Not: Not
8. Unique: No
9. Single or Multiple-Value: Single
10. Simple or Composite: Simple

3. Candidate Key: Geo_Info_ID

4. Primary Key: Geo_Info_ID

5. Strong/Weak Entity: Strong

6. Field to be indexed: None

7. Additional Comments: Geo_Info_ID references the Date_Time_Info tables primary key Date_Time_Info_ID. Also, if the foreign key is deleted, then use the cascade effect to remove any orphaned records.

A.4 Entity: Road Info

1. Description: The Road Information table keeps records of segment, max speed, lanes, lanes visible, surrounding, edge, and wildlife signs.

2. Attributes:

1. Name: Road_Info_ID

2. Description: This holds the identification number of each survey.

3. Domain/Type: integer

4. Value-Range: 0 – 9999999999

5. Signed, Unsigned, None: Unsigned

6. Default value: ""

7. Null Value Allowed or Not: Not

8. Unique: Yes

9. Single or Multiple-Value: Single

10. Simple or Composite: Simple

1. Name: Segment

2. Description: Specify the segment the individual has observed.

3. Domain/Type: varchar

4. Value-Range: 6 – 8

5. Signed, Unsigned, None: None

6. Default value: "5.0 mi"

7. Null Value Allowed or Not: Not

8. Unique: No

9. Single or Multiple-Value: Single

10. Simple or Composite: Composite

1. Name: Max_Speed

2. Description: Specify the speed when the observation took place.

3. Domain/Type: varchar

4. Value-Range: 9 – 10
5. Signed, Unsigned, None: None
6. Default value: “25.0 mph”
7. Null Value Allowed or Not: Not
8. Unique: No
9. Single or Multiple-Value: Single
10. Simple or Composite: Composite

1. Name: Lanes
2. Description: Specify how many total lanes existed.
3. Domain/Type: integer
4. Value-Range: 1 – 2
5. Signed, Unsigned, None: Unsigned
6. Default value: “5”
7. Null Value Allowed or Not: Not
8. Unique: No
9. Single or Multiple-Value: Single
10. Simple or Composite: Simple

1. Name: Lane_Visibility
2. Description: Specify if all or half of the lanes were visible.
3. Domain/Type: varchar
4. Value-Range: 3 – 4
5. Signed, Unsigned, None: None
6. Default value: “all”
7. Null Value Allowed or Not: Not
8. Unique: No
9. Single or Multiple-Value: Single
10. Simple or Composite: Simple

1. Name: Surrounding
2. Description: Specify the surrounding.
3. Domain/Type: varchar
4. Value-Range: 5 – 9
5. Signed, Unsigned, None: None
6. Default value: “Mixed”
7. Null Value Allowed or Not: Not
8. Unique: No
9. Single or Multiple-Value: Single
10. Simple or Composite: Simple

1. Name: Edge
2. Description: Specify the edge.
3. Domain/Type: varchar
4. Value-Range: 4 – 12
5. Signed, Unsigned, None: None

6. Default value: "Cliff"
7. Null Value Allowed or Not: Not
8. Unique: No
9. Single or Multiple-Value: Single
10. Simple or Composite: Simple

1. Name: Wildlife_Signs
2. Description: Specify if there were wildlife signs.
3. Domain/Type: varchar
4. Value-Range: 2 – 3
5. Signed, Unsigned, None: None
6. Default value: "Yes"
7. Null Value Allowed or Not: Not
8. Unique: No
9. Single or Multiple-Value: Single
10. Simple or Composite: Simple

3. Candidate Key: Road_Info_ID
4. Primary Key: Road_Info_ID
5. Strong/Weak Entity: Strong
6. Field to be indexed: None
7. Additional Comments: Road_Info_ID references the Date_Time_Info tables primary key Date_Time_Info_ID. Also, if the foreign key is deleted, then use the cascade effect to remove any orphaned records.

A.5 Entity: Kill Info

1. Description: The Kill Information table keeps records of what the observer has observed about the animal and the area of the animal. This also includes the confidence of the observer.

2. Attributes:

1. Name: Kill_Info_ID
2. Description: This holds the identification number of each survey.
3. Domain/Type: integer
4. Value-Range: 0 – 9999999999
5. Signed, Unsigned, None: Unsigned
6. Default value: ""
7. Null Value Allowed or Not: Not
8. Unique: Yes
9. Single or Multiple-Value: Single
10. Simple or Composite: Simple

1. Name: Kill_Status

2. Description: Specify if the animal is dead or alive.
3. Domain/Type: varchar
4. Value-Range: 2 – 3
5. Signed, Unsigned, None: None
6. Default value: “Yes”
7. Null Value Allowed or Not: Not
8. Unique: No
9. Single or Multiple-Value: Single
10. Simple or Composite: Simple

1. Name: ID_Confidence
2. Description: Specify the confidence level of the observation.
3. Domain/Type: varchar
4. Value-Range: 3 – 10
5. Signed, Unsigned, None: None
6. Default value: “Definite”
7. Null Value Allowed or Not: Not
8. Unique: No
9. Single or Multiple-Value: Single
10. Simple or Composite: Simple

1. Name: Seen_Kill_Before
2. Description: Specify if the animal has been observed before.
3. Domain/Type: varchar
4. Value-Range: 2 – 3
5. Signed, Unsigned, None: None
6. Default value: “Yes”
7. Null Value Allowed or Not: Not
8. Unique: No
9. Single or Multiple-Value: Single
10. Simple or Composite: Simple

1. Name: Country
2. Description: Specify the country where the animal was observed.
3. Domain/Type: varchar
4. Value-Range: 1 – 255
5. Signed, Unsigned, None: None
6. Default value: “USA”
7. Null Value Allowed or Not: Not
8. Unique: No
9. Single or Multiple-Value: Single
10. Simple or Composite: Simple

1. Name: State
2. Description: Specify the state where the animal was observed.
3. Domain/Type: varchar

4. Value-Range: 1 – 255
5. Signed, Unsigned, None: None
6. Default value: “CA”
7. Null Value Allowed or Not: Not
8. Unique: No
9. Single or Multiple-Value: Single
10. Simple or Composite: Simple

1. Name: City
2. Description: Specify the city where the animal was observed.
3. Domain/Type: varchar
4. Value-Range: 1 – 255
5. Signed, Unsigned, None: None
6. Default value: “Camarillo”
7. Null Value Allowed or Not: Not
8. Unique: No
9. Single or Multiple-Value: Single
10. Simple or Composite: Simple

1. Name: Kill_Size
2. Description: Specify the animal size at observation.
3. Domain/Type: varchar
4. Value-Range: 5 – 6
5. Signed, Unsigned, None: None
6. Default value: “Small”
7. Null Value Allowed or Not: Not
8. Unique: No
9. Single or Multiple-Value: Single
10. Simple or Composite: Simple

1. Name: Kill_Species_Common_Name
2. Description: Specify the animal’s common name otherwise unknown.
3. Domain/Type: varchar
4. Value-Range: 1 – 255
5. Signed, Unsigned, None: None
6. Default value: “Unknown”
7. Null Value Allowed or Not: Not
8. Unique: No
9. Single or Multiple-Value: Single
10. Simple or Composite: Simple

1. Name: Kill_Species_Latin_Name
2. Description: Specify the animal’s Latin name otherwise unknown.
3. Domain/Type: varchar
4. Value-Range: 1 – 255
5. Signed, Unsigned, None: None

6. Default value: "Unknown"
7. Null Value Allowed or Not: Not
8. Unique: No
9. Single or Multiple-Value: Single
10. Simple or Composite: Simple

1. Name: Kill_Class
2. Description: Specify the animal's classification otherwise unknown.
3. Domain/Type: varchar
4. Value-Range: 1 – 255
5. Signed, Unsigned, None: None
6. Default value: "Unknown"
7. Null Value Allowed or Not: Not
8. Unique: No
9. Single or Multiple-Value: Single
10. Simple or Composite: Simple

3. Candidate Key: Kill_Info_ID
4. Primary Key: Kill_Info_ID
5. Strong/Weak Entity: Strong
6. Field to be indexed: None
7. Additional Comments: Kill_Info_ID references the Date_Time_Info tables primary key Date_Time_Info_ID. Also, if the foreign key is deleted, then use the cascade effect to remove any orphaned records.

A.6 Entity: Additional Info

1. Description: The Additional Information table keeps records of weather, temperature, photos, and comments.

2. Attributes:

1. Name: Additional_Info_ID
2. Description: This holds the identification number of each survey.
3. Domain/Type: integer
4. Value-Range: 0 – 9999999999
5. Signed, Unsigned, None: Unsigned
6. Default value: ""
7. Null Value Allowed or Not: Not
8. Unique: Yes
9. Single or Multiple-Value: Single
10. Simple or Composite: Simple

1. Name: Weather
2. Description: Specify the weather at observation.

3. Domain/Type: varchar
4. Value-Range: 3 – 23
5. Signed, Unsigned, None: None
6. Default value: “Clear”
7. Null Value Allowed or Not: Not
8. Unique: No
9. Single or Multiple-Value: Single
10. Simple or Composite: Simple

1. Name: Temperature
2. Description: Specify the temperature at observation.
3. Domain/Type: varchar
4. Value-Range: 3 – 6
5. Signed, Unsigned, None: None
6. Default value: “212 F”
7. Null Value Allowed or Not: Not
8. Unique: No
9. Single or Multiple-Value: Single
10. Simple or Composite: Composite

1. Name: Photo
2. Description: Optional photo at the user’s disposal.
3. Domain/Type: varchar
4. Value-Range: 0 – 19
5. Signed, Unsigned, None: Unsigned
6. Default value: “”
7. Null Value Allowed or Not: Allowed
8. Unique: Yes
9. Single or Multiple-Value: Single
10. Simple or Composite: Simple

1. Name: Comments
2. Description: Optional comments at the user’s disposal.
3. Domain/Type: varchar
4. Value-Range: 1 – 255
5. Signed, Unsigned, None: None
6. Default value: “”
7. Null Value Allowed or Not: Allowed
8. Unique: No
9. Single or Multiple-Value: Single
10. Simple or Composite: Simple

3. Candidate Key: Additional_Info_ID
4. Primary Key: Additional_Info_ID
5. Strong/Weak Entity: Strong
6. Field to be indexed: None

7. Additional Comments: Additional_Info_ID references the Date_Time_Info tables primary key Date_Time_Info_ID. Also, if the foreign key is deleted, then use the cascade effect to remove any orphaned records.

A.7 Entity: Transect Info

1. Description: The Transect Information table keeps records of origin, way, and destination points of the road being observed for transect.

2. Attributes:

1. Name: Transect_Info_ID
 2. Description: This holds the identification number of each survey.
 3. Domain/Type: integer
 4. Value-Range: 0 – 9999999999
 5. Signed, Unsigned, None: Unsigned
 6. Default value: ""
 7. Null Value Allowed or Not: Not
 8. Unique: Yes
 9. Single or Multiple-Value: Single
 10. Simple or Composite: Simple
-
1. Name: Transect_Points
 2. Description: Specify origin, way, destination points of the road being observed for transect.
 3. Domain/Type: text
 4. Value-Range: 0 – 65535
 5. Signed, Unsigned, None: None
 6. Default value: ""
 7. Null Value Allowed or Not: Allowed
 8. Unique: No
 9. Single or Multiple-Value: Multiple
 10. Simple or Composite: Composite

3. Candidate Key: Transect_Info_ID
4. Primary Key: Transect_Info_ID
5. Strong/Weak Entity: Strong
6. Field to be indexed: None
7. Additional Comments: Transect_Info_ID references the Date_Time_Info tables primary key Date_Time_Info_ID. Also, if the foreign key is deleted, then use the cascade effect to remove any orphaned records.

Appendix B: Description of the API for Additional Filters

This appendix describes adding additional API filters.

In the `gmaps.htm` file located in the administrator's main directory, additional code is added, so that the browser can access the new map filter. A label and input box should be present within the table structure. The input box should have a specific keyword for identification, which is to be used later. If there are any specific characters that need to be omitted from the input box, add a JavaScript function to validate for either `onkeyup` or `onchange` within the HTML tag, function to be defined later.

In the `mtajax.js` file located in the administrator's JavaScript directory, additional code is added. JavaScript validation functions are defined for sending all necessary values to the server. If the new map filter needs to be validated, define the function and display errors on the web browser accordingly. Once validation has been handled, modify the `updateMap` function, so that the value(s) of the newest map filter can be posted to the server along with other values from existing filters. To retrieve the value of the new input box, defined above by specific identification keyword, the following is done:

```
var variablename = document.getElementById('identification_keyword').value;
```

After the value has been retrieved, update the `iframe` source string (in three locations) to append the additional parameter, so that the string can be posted to the server as follows:

```
uiframe.src = './class/gmaps.php?fsearch=' + fsearch + '&dist=' + dist + '&clat=' + latitude + '&clon=' + longitude + '&newparameter=' + variablename + '&pospj=AJS';
```

In the `gmaps.php` file located in the class directory, additional code is added, so that the map can be regenerated with the new filter values. The parameters from the JavaScript code (part two above) need to be retrieved and desensitized. Using a ternary operation, the value is retrieved; if the value is set, retrieve the value; else set variable name to 'Any'. For example,

```
$variablename = isset($_GET['newparameter']) ? $_GET['newparameter'] : 'Any';
```

First, desensitize the value retrieved using the `mysql_real_escape` (PHP function) and remove any slashes with the `stripslashes` PHP function. For example,

```
$newvariablename = mysql_real_escape_string(stripslashes($variablename));
```


Second, create a statement to handle if/else cases of the newly retrieved values to form a MySQL statement. Check only if newvariablename is not equal to 'Any'. For example,

```
else if(($isearch == Any' && ($dist == Any' && ($newvariablename = Any'))
{
    $mysql = FROM gmdbTable WHERE Kill_Species_Common_Name LIKE
'%%$newvariablename% OR Kill_Class LIKE %%$newvariablename% OR Country LIKE
'%%$newvariablename% OR State LIKE %%$newvariablename% OR City LIKE
'%%$newvariablename%' OR Kill_Species_Latin_Name LIKE %%$newvariablename%'
OR Weather LIKE %%$newvariablename%' OR Surrounding LIKE
'%%$newvariablename% OR Edge LIKE %%$newvariablename%";
}
```

Third, check if distance and newvariablename are not equal to 'Any'. Next, check the distance with a regular expression. If there is a match, disterror is set to 0; else disterror is set to 1. Using the Haversine function and the newvariablename values, the following is achieved:

```
else if(($isearch == Any' && ($dist = Any' && ($newvariablename = Any'))
{
    $distpat = '/^(? [0-9]{1,4}[.][0-9]{1} [0-9]{1,4})$/';
    if(preg_match($distpat $dist))
    {
        $mysql = ' (3959 * acos(cos(radians(" $dlat ')) * cos(radians(Latitude)) *
cos(radians(Longitude) * radians(' $dlon ')) + sin(radians(' $dlat ')) *
sin(radians(Latitude)))) AS distance FROM gmdbTable HAVING distance <= $dist '
AND (Kill_Species_Common_Name LIKE %%$newvariablename% OR Kill_Class LIKE
'%%$newvariablename% OR Country LIKE %%$newvariablename% OR State LIKE
'%%$newvariablename% OR City LIKE %%$newvariablename% OR
Kill_Species_Latin_Name LIKE %%$newvariablename%' OR Weather LIKE
'%%$newvariablename%' OR Surrounding LIKE %%$newvariablename% OR Edge LIKE
'%%$newvariablename%');
        $disterror = '0";
    }
    else
    {
        $disterror = "1";
    }
}
```

Fourth, check if distance, newvariablename, and search are not equal to 'Any'. Next, check the distance with a regular expression. If there is a match, disterror is set to 0; else disterror is set to 1. Using the Haversine function, newvariablename, and search values, the following is achieved:

```
else if(($search = Any') && ($dist = Any') && ($newvariablename = 'Any'))
{
    $distpat = '/^(?: [0-9]{1,4}[.][0-9]{1}||[0-9]{1,4})$/';
    if(preg_match($distpat $dist))
    {
        $msql = ' (3959 * acos(cos(radians(" $dlat ')) * cos(radians(Latitude)) *
cos(radians(Longitude) - radians(' $dlon ')) + sin(radians(" $dlat ')) *
sin(radians(Latitude)))) AS distance FROM gmdbTable HAVING distance <' $dist "
AND (Kill_Species_Common_Name LIKE '%$search%' OR Kill_Class LIKE
'%$search%' OR Country LIKE '%$search%' OR State LIKE '%$search%' OR City
LIKE '%$search%' OR Kill_Species_Latin_Name LIKE '%$search%' OR Weather
LIKE '%$search%' OR Surrounding LIKE '%$search%' OR Edge LIKE '%$search%')
AND (Kill_Species_Common_Name LIKE '%$newvariablename%' OR Kill_Class LIKE
'%$newvariablename%' OR Country LIKE '%$newvariablename%' OR State LIKE
'%$newvariablename%' OR City LIKE '%$newvariablename%' OR
Kill_Species_Latin_Name LIKE '%$newvariablename%' OR Weather LIKE
'%$newvariablename%' OR Surrounding LIKE '%$newvariablename%' OR Edge LIKE
'%$newvariablename%')";
        $disterror = '0';
    }
    else
    {
        $disterror = "1";
    }
}
```

Lastly, check if search and newvariablename are not equal to 'Any'. Using the search and newvariablename values, the following is achieved:

```
else if(($search = 'Any') && ($dist == Any') && ($newvariablename = 'Any'))
{
    $msql = FROM gmdbTable WHERE (Kill_Species_Common_Name LIKE
'%$search%' OR Kill_Class LIKE '%$search%' OR Country LIKE '%$search%' OR
State LIKE '%$search%' OR City LIKE '%$search%' OR Kill_Species_Latin_Name
```

```
LIKE %$isearch% OR Weather LIKE %$isearch% OR Surrounding LIKE  
'%$isearch% OR Edge LIKE %$isearch%' AND (Kill_Species_Common_Name LIKE  
'%$newvariablename%' OR Kill_Class LIKE %$newvariablename% OR Country LIKE  
'%$newvariablename%' OR State LIKE %$newvariablename% OR City LIKE  
'%$newvariablename%' OR Kill_Species_Latin_Name LIKE %$newvariablename%  
OR Weather LIKE %$newvariablename% OR Surrounding LIKE  
'%$newvariablename%' OR Edge LIKE %$newvariablename%');  
}
```

Appendix C: Description of the Mobile API

This appendix discusses how alternative mobile devices should communicate to the server for uploading reports.

The report should consist of all attributes in Appendix A. When submitting the report, the image parameter value is checked, so that the server can handle the image if it exists. The mobile device should connect to the `iupload.php` file which is located in the class directory. This method should be accomplished using a POST method which is established through an HTTP synchronous connection. The server will respond to the mobile device with a string consisting of “Invalid”, “Empty”, or [Image Name]. If image name or “Empty” is returned, the value will be used in the remainder of the report when uploading to the server. If “Invalid” is returned, the mobile application should notify the user for image correction. Valid image extensions are: JPG, JPEG, and PNG. Image size should not exceed 6MB.

After the image value has been successfully retrieved from the server, the remainder of the report is then processed. The mobile device should connect to the `isubmitdata.php` file which is located in the class directory. The string sent from the mobile application to the server, through the POST method established through an HTTP synchronous connection, should resemble the following:

```
submitdata=submit&cdate=[datevalue]&ctime=[timevalue]&latitude=[latitudevalue]&longitude=[longitudevalue]&segment=[segmentvalue]&mspeed=[mspeedvalue]&alanes=[alanesvalue]&lanesvis=[lanesvisvalue]&surround=[surroundvalue]&edges=[edgesvalue]&wildsigns=[wildsignsvalue]&stat=[statvalue]&identconf=[identconfvalue]&seenb=[seenbvalue]&cnyr=[cnyrvalue]&state=[statevalue]&city=[cityvalue]&sizes=[sizesvalue]&scn=[scnvalue]&sln=[slnvalue]&cls=[clsvalue]&wethr=[wethrvalue]&temp=[tempvalue]&photo=[photovalue]&cmmts=[cmmtsvalue]&tpoints=[tpointsvalue]
```

All parameter values should follow the same valid input as described in Chapter 4 and Appendix A. With the addition of transect points, the string should contain values of latitude and longitude separated by a comma. For example, a valid transect input is as follows:

```
startlatitude,startlongitude,waypointlatitude,waypointlongitude,...,endlatitude,endlongitude.
```

The server should respond to the mobile device with an XML data structure consisting of success or errors. If success is returned, the mobile application should handle the application accordingly. If errors are returned, the mobile application should notify the user through the application to correct any invalid input.

Success XML data structure example as follows:

```
<?xml version="1.0" encoding="utf-8"?>
  <recstatus>
    <insertsuccess data="Successfully Inserted" />
  </recstatus>
```

Error XML data structure example as follows:

```
<?xml version="1.0" encoding="utf-8"?>
  <recstatus>
    <dateerror data="Invalid Date" />
    <timeerror data="Invalid Time" />
    <laterror data="Invalid Latitude" />
    <lonerror data="Invalid Longitude" />
    <segerror data="Invalid Segment" />
    <mspeederror data="Invalid Max Speed" />
    <cntryerror data="Invalid Country" />
    <stateerror data="Invalid State" />
    <cityerror data="Invalid City" />
    <commonerror data=" Invalid Common Name" />
    <latinerror data=" Invalid Latin Name" />
    <classerror data=" Invalid Class" />
    <temperror data=" Invalid Temperature" />
    <photoerror data=" Invalid Image" />
    <cmmterror data=" Invalid Comments" />
    <inserterror data="Cannot Insert" />
    <submiterror data="Survey Invalid" />
  </recstatus>
```