12/11/2014

CIS 4911 Senior Project

Intelligence Inference Engine
Requirements Document

Group Members
Jose Acosta
Lazaro Herrera

Mentor
Eric Kobrin

Instructor
Masoud Sadjadi

Intelligence Inference Engine : iie-dev.cs.fiu.edu

Abstract

Most web searches currently depend on matching keyword or phrases in order to return results. Matching keyword and phrases works as long as you know what your are searching for and you only want information to that specific thing. In the modern world, when it comes to cyber security new exploits and attacks are constantly coming out. It is not feasible to keep track and know everything about these attacks because there are so many. Fortunately, most new attacks and exploits are usually related to old attacks and exploits. This is where the Semantic Web comes in. Data in Semantic Web is represented by relations, this makes it easy to search for things even without knowing exactly what you are looking for. The only problem with the Semantic Web is that there is a barrier of entry, knowing a specific querying language. This project aims to lower that barrier of entry.

Table of Contents

Date: 12/1/2014

References

Introduction

Humans are capable of using the Web to carry out tasks such as finding the German translation for "eight days", reserving a library book, and searching for the lowest price for a DVD. However, machines cannot accomplish all of these tasks without human direction because web pages are designed to be read by people, not machines. The semantic web is a vision of information that can be readily interpreted by machines, so machines can perform more of the tedious work involved in finding, combining, and acting upon information on the web.

Unfortunately the Semantic Web has a problem, it is designed so that it easily read by machines meaning that it is hard for a human user to directly read the data that is on the Semantic Web. This means that someone must query the data to find what they are looking for. This is a problem because it requires the person to know one of the Resource Description Framework(RDF) querying languages

Problem Definition.

The Semantic Web is about linked data. Linked Data is resource-based linking of information. The Semantic web is built of by large of linked data which is defined by the Resource Description Framework(RDF). Each RDF data point consist of three parts: a subject, a predicate, and an object. In essence, RDF give you little building blocks of data that can be connected to other building blocks of data in both directions. When you build complicated webs of connected information, you end up with really specific detailed structures of conceptual knowledge over which you can answer complex question programmatically.

Right now, if you went to Google and put a search query like “Everything related to exploits on SSL that affects Linux and similar systems ”, the results would mostly be articles that cover SSL, some that may cover Linux, and maybe some that cover exploits on SSL. That search result would be useless because it does not give you what you asked for, instead it just matched the keywords you put in, Google does not know what systems are similar to Linux by keywords alone, it does not know what how to put all that together to find specifically what your are looking for . This means that you have to know what you are looking for before you look for it. With Semantic Web you can use relations between different data points and tries to answer the query using relations.

Scope of system

The deliver ables for this project will be given in two week increments which is our current sprint cycle. This will line up with our in-class scheduled presentations. We believe that the features that should be developed should both be shown to the client and our fellow classmates for feedback. The technical limits will go as far as generating valid SPARQL from a web form and using that for both searches and submission of data to our triple-store.

Terminology - Definitions, acronyms, and abbreviations.

OSINT - Open-source intelligence

Cyber-attack - Any type of offensive maneuver employed by individuals or whole organizations that targets computer information systems, infrastructures, computer networks, and/or personal computer devices by various means of malicious acts usually originating from an anonymous source that either steals, alters, or destroys a specified target by hacking into a susceptible system [1]

Triple store - A triplestore is a purpose-built database for the storage and retrieval of triples through semantic queries. A triple is a data entity composed of subject-predicate-object. [2]

RDF - a family of World Wide Web Consortium (W3C) specifications originally designed as a metadata data model. It has come to be used as a general method for conceptual description or modeling of information that is implemented in web resources, using a variety of syntax notations and data serialization formats. [3]

Semantic Web - collaborative movement led by international standards body the World Wide Web Consortium (W3C). The standard promotes common data formats on the World Wide Web. By encouraging the inclusion of semantic content in web pages, the Semantic Web aims at converting the current web, dominated by unstructured and semi-structured documents into a "web of data". The Semantic Web stack builds on the W3C's Resource Description Framework (RDF). [4]

SPARQL - an RDF query language, that is, a semantic query language for databases, able to retrieve and manipulate data stored in Resource Description Framework format. [5]

Overview of document

This document will describe the current solution that is used to combat the problem and describes the limitations that the current system faces. This document also includes the project plan which details the breakdown of the work between the group members and what the responsibility of the group members are. Towards the end of the document, the use cases for the system will be described along with scenarios. The document will also contain all the various different UML diagrams that describe the system and how it functions and are used to make the use case, static, and dynamic models.

Current System

Currently, there exist many websites that catalog cyber-attacks, data breaches and vulnerabilities. Although all of these websites are critical to today's security researchers, missing any of them can leave important data behind as well as not knowing the correct term to search for can cause some of the important information to be left behind. Consider the possibility that an attack may be filed as targeting "banks" yet the security researchers might be searching for an attack that is targeting "financial institutions". Similar terms being used but a standard search engine does not take these into account.

Project Plan

For this project, the plan is to utilize agile development to deliver quick small sections of the overall system to a client beginning with a basic UI module, and enhancing it with some modular UI capabilities, evidence evaluation, query generation and submission, custom query capabilities and storage and a database exploration tool to be able to access the triple-store RDF database.

Project organization

Lazaro Herrera - Developer / UI Design / Testing

Jose Acosta - Developer / Database / Query Library Development

Eric Kobrin - Project Manager / Client

Work breakdown

For our Trello board, we have decided to use color tagging to both identify the assignments of tasks and the completion status of tasks

Colors and Definitions for Priority

Blue - Card created by students

Purple - Card created by client

Colors and Definitions for Completions (only one active at a time)

Green - This task has a completion of 25% or more

Yellow - This task has a completion of 50% or more

Orange - This task has a completion of 75% or more

Red - This task has a completion of 100%

The following epics currently exist on our board.

Some of these are receiving checklists to be converted into stories.

- Setup Development Environment (Mulgara)
- Setup Development Environment (Web Server + Bootstrap)
- Develop Primary Feature (Data Entry Web Form)
- Test Primary Feature (Data Entry Web Form)
- Develop Primary Feature (Database Explorer)
- Test Primary Feature (Database Explorer)
- Develop Primary Feature (Data Retrieval Web Form)
- Test Primary Feature (Data Retrieval Web Form)
- Develop Secondary Feature (Custom Reusable Queries)
- Test Secondary Feature (Custom Reusable Queries)
- Develop Secondary Feature (Confidence Ranking)

There were four major milestones that were followed for this project. The first was getting a basic UI. This milestone included getting all the pages in a mock up stage that were actual html pages. This was important because future updates could be done directly on the mock up eventually turning it into the actual UI. The next milestone was getting basic functionality working. This milestone was important because it was necessary to know whether the basic idea of the solution was a possible course of action. The next milestone was tweaking and upgrading the previous two, the UI and functionality. The final milestone is testing the code. This is very important to make sure that the software solution is in working order and that everything was implemented correctly.

Cost Estimate

Cost Matrix	Weeks Required	Jose - $30 hour / 12hr week	Lazaro - $30 hour / 12hr week	Total Cost
Data Entry Form Implementation	2	$720.00	$720.00	$12,672.00
Data Entry Form Testing	1	$360.00	$360.00	
Data Retrieval Form Implementation	2	$720.00	$720.00	
Data Retrieval Form Testing	1	$360.00	$360.00	
Database Explorer Implementation	3	$1,080.00	$1,080.00	
Database Explorer Testing	2	$720.00	$720.00	

Custom Queries Implementation	1	$360.00	$360.00	
Custom Queries Testing	0.6	$216.00	$216.00	
Confidence Ranking Implementation	3	$1,080.00	$1,080.00	
Confidence Ranking Testing	2	$720.00	$720.00	

Proposed System Requirements

This section of the document will cover what the proposed system will be able to do. It will describe what high level functionality the system should have and it will analyze those functions to determine requirements. This section will also cover all the scenarios that the system should be able to handle as well as their associated use cases. It will also cover the class structure of the system and will show the dynamic processes that the system will carry out.

Functional Requirements – describes high-level functionality

- A user must be able to submit data to be stored through some type of web form
- The data should be query-able directly using Sparql or Datalog
- A user must be able to set up predefined queries which are accessible by other users
- The system must be able to graphically display all the data that it currently stores.
- The system must be able to import data given a file with a specific format.
- The system must be able to export data to a file in a specific format.
- The system must be able to set a confidence interval for the data that is being collected.
- The data must be stored in one of the existing semantic web triple- or quad-stores such as Mulgara or Jena

For each functional requirement state the associated non-functional requirements, if any, for *Usability, Reliability, Performance,* and *Supportability*.

Analysis of System Requirements

Analysis models – contains the complete functional specification and is mainly for the designers and programmers. This section describes the diagrams in the Appendices B - D and validates the models against the use cases.

Scenarios

Scenario 1: Data Entry Success

Bob, a security analyst at company X, has gathered some data on attacks that have been happening to Company Y. Bob wants to share this data to see if further connections can be made. Bob goes to the web site that is being hosted at URL Z, and clicks on the button to contribute data. When Bob gets to the web form, he will fill out the various field that are required. Once Bob has filled out the form, he will click on the submit button. The data that Bob entered will then be sent to the Jena triple store and Bob will receive a confirmation telling him that the data has been entered.

Scenario 2: Data Entry Fail : Back End Down

Bob, a security analyst at company X, has gathered some data on attacks that have been happening to Company Y. Bob wants to share this data to see if further connections can be made. Bob goes to the web site that is being hosted at URL Z, and clicks on the button to contribute data. When Bob gets to the web form, he will fill out the various field that are required. Once Bob has filled out the form, he will click on the submit button. The web form will attempt to connect to the back end to enter the information Bob has provided, but it is down. Bob will remain on the current web form and receive an alert informing him that the back end service it currently down and to please try to enter the information again later.

Scenario 3: Data Entry Fail : Invalid Information

Bob, a security analyst at company X, has gathered some data on attacks that have been happening to Company Y. Bob wants to share this data to see if further connections can be made. Bob goes to the web site that is being hosted at URL Z, and clicks on the button to contribute data. When Bob gets to the web form, he will fill out the various field that are required. Once Bob has filled out the form, he will click on the submit button. Once the submit button has been pressed the information on the web form is checked to verify it is valid. It is found that one or more of the fields that Bob has filled out contain invalid information. Bob will remain at the current form. Bob can now fix the invalid fields and retry or he can exit the page and abandon entering data.

Scenario 4 : Data Entry : Import Data

Bob, a security analyst at company X, has gathered a lot of data on some recent attacks. Bob want to share his data but it is too much data to enter manually. Bob would go to the web site that is being hosted at URL Y, and clicks on the button to contribute data. On the page to contribute data, Bob would click on the button that says bulk import. Bob would then choose the file with the data he wants to contribute, and is in the correct format. The data will then be added to the server and the page will show how much data was inserted.

Scenario 5: Data Retrieval Success: Custom Query

Bob, a security analyst at company X, has noticed that recently there have been a lot of attacks in the field his client's company is in. Bob wants too see if his client's company is at risk of an attack. Bob goes to the web site that is being hosted at URL Z, and clicks on the button to query the data that has been collected. In the next form, Bob clicks on the button to enter his own custom sparql query. Bob will enter his query into the text box provided and then click on the submit button. The query that Bob has entered will then be sent to the Jena server and the results will sent back to be displayed on the web client Bob is on. Bob can now view the results of his query.

Scenario 6: Data Retrieval Success: Generated Query

Bob, a security analyst at company X, has noticed that recently there have been a lot of attacks in the field his client's company is in. Bob wants too see if his client's company is at risk of an attack. Bob goes to the web site that is being hosted at URL Z, and clicks on the button to query the data that has been collected. In the next form, Bob will be presented with several fields to fill out in order for the page to automatically generate a query. Once Bob has filled out the required fields, He will then click on the submit button. The web form will generate a query and send it to the back end Jena service. Once the results are retrieved, the page will show him the results of his query.

Scenario 7: Data Retrieval Failure : Back End Is Down

Bob, a security analyst at company X, has noticed that recently there have been a lot of attacks in the field his client's company is in. Bob wants too see if his client's company is at risk of an attack. Bob goes to the web site that is being hosted at URL Z, and clicks on the button to query the data that has been collected. Bob will fill out the necessary information for the type of query he wants to do. Once Bob has filled out the information, he will press submit. Since the back end is currently down, Bob will remain at the current page and he will receive a message informing him that the back end is currently down and he should retry at a later time. Bob can now wait and retry with his query or abandon his query.

Scenario 8: Data Retrieval Failure : No Results For Desired Query

Bob, a security analyst at company X, has noticed that recently there have been a lot of attacks in the field his client's company is in. Bob wants too see if his client's company is at risk of an attack. Bob goes to the web site that is being hosted at URL Z, and clicks on the button to query the data that has been collected. Bob will fill out the necessary information for the type of query he wants to do. Once Bob has filled out the information, he will press submit. The query is sent to the Jena back end but no results are generated from the query. Bob will remain at the current page and receive a table showing him that his query generated no results. Bob can now change his query and retry or he can abandon his query.

Scenario 9: Custom Query Saving Success

Bob, a security analyst at company X, has created a query that he thinks will retrieved information other people will want. Bob wants to share this query. Bob goes to the website that is being hosted at URL Z, and click on the button to query the data that has been collected. In the next form, Bob will click in the button that says "Custom Query". Once Bob is at the custom query form he will fill out the text box provided. After Bob has filled out the text box provided with the query that he wants to save, he will click on the check box that says "Save Query". Enabling the "Save Query" button will add two additional text boxes to the current form. Bob will fill out the new text boxes with a title for the query and for tags (separated by commas) to describe the query. Once the two new text boxes are filled, Bob will press the submit button. If the query was successful, Bob will be taken to a new page containing the results of the query and a message saying the query was saved along with the id for said query for quick access.

Scenario 10: Custom Query Saving Failure : Back End Services Down

Bob, a security analyst at company X, has created a query that he thinks will retrieved information other people will want. Bob wants to share this query. Bob goes to the website that is being hosted at URL Z, and click on the button to query the data that has been collected. In the next form, Bob will click in the button that says "Custom Query". Once Bob is at the custom query form he will fill out the text box provided. After Bob has filled out the text box provided with the query that he wants to save, he will click on the check box that says "Save Query". Enabling the "Save Query" button will add two additional text boxes to the current form. Bob will fill out the new text boxes with a title for the query and for tags (separated by commas) to describe the query. Once the two new text boxes are filled, Bob will press the submit button. Because the back end services are down the query cannot complete or save. Bob will receive a message informing him that the back end is currently down and the query could not be saved and to wait and retry later. Bob can now wait and retry at a later time or he can abandon saving his query.

Scenario 11: Custom Query Saving Failure : Query already exists

Bob, a security analyst at company X, has created a query that he thinks will retrieved information other people will want. Bob wants to share this query. Bob goes to the website that is being hosted at URL Z, and click on the button to query the data that has been collected. In the next form, Bob will click in the button that says "Custom Query". Once Bob is at the custom query form he will fill out the text box provided. After Bob has filled out the text box provided with the query that he wants to save, he will click on the check box that says "Save Query". Enabling the "Save Query" button will add two additional text boxes to the current form. Bob will fill out the new text boxes with a title for the query and for tags (separated by commas) to describe the query. Once the two new text boxes are filled, Bob will press the submit button. The query is attempted to be saved but it is found the query already exists. Bob will be taken to a new page containing the results of his query but will also receive a message informing him that the query already exists and give him the query's id.

Scenario 12: Looking Up Custom Queries Success

Bob, a security analyst at company X, wants to look up some data, but he is not entirely sure on what he specifically wants to look for. Bob would go to the website that is at URL Z. Once Bob is at the website, he will click on the button to query the data that has been collected. At the query form, Bob will now click on the "Search" button which will take him to a new form. At this new form, Bob can fill out a text box in which he add tags to define what he wants to search for. Bob will click on the submit button when he has put in all the tags he wants to search for. The current saved queries will be compared to the tags Bob has input and relevant queries will be sent back to Bob's client and be displayed for Bob to look at.

Scenario 13: Looking Up Custom Queries Failure: Back End Services Down

Bob, a security analyst at company X, wants to look up some data, but he is not entirely sure on what he specifically wants to look for. Bob would go to the website that is at URL Z. Once Bob is at the website, he will click on the button to query the data that has been collected. At the query form, Bob will now click on the "Search" button which will take him to a new form. At this new form, Bob can fill out a text box in which he add tags to define what he wants to search for. Bob will click on the submit button when he has put in all the tags he wants to search for. Because the back end services are down the search cannot be completed. Bob will remain at his current page and receive a message informing him that the back end services are current down and to try at a later time. Bob can now wait and retry his search or he can abandon it.

Scenario 14: Looking Up Custom Queries Failure: No Results

Bob, a security analyst at company X, wants to look up some data, but he is not entirely sure on what he specifically wants to look for. Bob would go to the website that is at URL Z. Once Bob is at the website, he will click on the button to query the data that has been collected. At the query form, Bob will now click on the "Search" button which will take him to a new form. At this new form, Bob can fill out a text box in which he add tags to define what he wants to search for. Bob will click on the submit button when he has put in all the tags he wants to search for. The back end services will attempt to find queries that match the parameters given by Bob but none are found. Bob will remain at his current page and receive a message informing him that his search parameters yielded no results. Bob can now modify his search and retry or he can abandon his search.

Scenario 15: Confidence Interval

Bob, a security analyst at company X, wants to look up some data, Bob would go to the website that is at URL Z. Once Bob is at the website, he will click on the button to query the data that has been collected. At the query form, Bob will now click on the "Search" button which will take him to a new form. At this new form, Bob can fill out a text box in which he add tags to define what he wants to search for. Bob will click on the submit button when he has put in all the tags he wants to search for. The current saved queries will be compared to the tags Bob has input and relevant queries will be sent back to Bob's client and be displayed for Bob to look at. Confidence Interval will give Bob the results sorted by the evidence that was collected as reference when the data was initially input.

Scenario 16: RDF Data Explorer

Bob, a security analyst at company X, is curious about all the data that has been entered into the system but does not want to know any specific thing about the data. Bob would navigate to the RDF Data Explorer page. Once Bob is at the Database explorer page, a graph will be created that shows all the data that is currently in the database. From here Bob can examine the graph.

Scenario 17: RDF Data Explorer Failure: No Data

Bob, a security analyst at company X, is curious about all the data that has been entered into the system but does not want to know any specific thing about the data. Bob would navigate to the RDF Data Explorer page. Once Bob is at the Database explorer page, a graph will be created that shows all the data that is currently in the database. If there is no data then the graph will display two null point and a line between them representing that there is currently no data in the database.

Scenario 18 : Export Data

Bob, a security analyst at company X, has been using the system and has found the data provided to be useful. Bob decides he wants all the data that the database has in order to look over it. Bob will head to the Search Now page and click on the "Bulk Data Export" button. Bob will download a file containing all the data that is currently in the database.

Use case model

There are 7 use cases in this system (refer to Appendix B). The first use case is the Store Data use case. This use case covers the requirement to allow users to be able to store data into the system. Scenarios 1 to 3 are represented in this use case. The next use case is the Import Data use case. This use case covers the requirement to allow users to bulk import data into the database. This use case is an extension of the Store Data use case because it stores data through an additional method. This use case covers scenario 4. The next use case is the Query Data use case. This use case covers the requirement to allow uses to query the database and it covers scenario 6 to 8. The next use case is the Export Data use case. This use case fulfills the requirement to allow users to export data from the database. It is an extension of the Query Data use case as it queries for all the data then exports it. Scenario 18 is covered by this use case. The next use case is Custom Query, this use case also extends the Query Data use case as it also queries the database. This use case satisfies the requirement to allow users to provide their own queries to be executed and to search other submitted queries and covers scenarios 5,9,12,13, and 14. The following use case is the Saving Custom Query. This use case provides the functionality for the requirement to save custom queries. This use case covers scenarios 10 and 11. The final use case is the Data Visualizer. This meets the requirement of being able to see a visualization of all the data currently in the database. This use case covers scenarios 16 and 17.

Static model

Our deployment diagram is composed of a deployment server running Easy DEV PHP which is a standard WAMP stack optimized for developer speed. We utilize it's MySQL, PHP and Apache along with Apache Fuseki/Jena as our endpoint for triple-store storage. Apache serves the web pages, PHP handles MySQL/Jena interaction, MySQL stores custom queries and custom query tags and Jena stores relational information.

In order for the system to generate queries from the text that user provided a javascript library that created a query generator of a specified type. There are 2 type of query generators in the library that each create a specific type of SPARQL query. These 2 query generator are both extensions of an abstract generator class, as both generators are have the same functionality apart from the query generated. The abstract class is an implementation of an interface that specifies all the functionality needed to create SPARQL queries. The query generators use two classes in order to store the query information, Prefix and Expression. Expressions are the core of the query and they store the building blocks of the query. Expressions consist of a Predicate and two rdfObject. Predicate and rdfObject are both building blocks of Expressions. Prefixes are required by both Generators and Predicate as they are used to store definitions that are needed to create valid SPARQL queries.

There are 4 specific UI classes that extend User Interface (Search, Submit, Database Exploration and Custom search). Each of them extend the basic user interface handlers and implement their specific features for each of their pages. These are all embedded in the HTML files that they are part of with their javascript code.

Dynamic model

The first dynamic model is the Modular Field addition sequence diagram. This model shows how a click travels from the user to the search.html context to the search.js context and it causes jQuery to generate the extra modular fields and display them.

The second dynamic model is the Modular Field removal sequence diagram. This model shows how a click travels from the user to the search.html context to the search.js context and it causes jQuery to select the last added modular fields and remove them from view.

The third dynamic model is the Database Search sequence diagram. It shows how the data is handled after the submit and the BuildGeneratorFunction adds prefixes, expressions and then the generator context is passed to the jQuery JS which triggers the final query build. This also shows how JS AJAX is utilized in order to execute these queries.

The fourth dynamic model is the Database Storage sequence diagram. It shows how the data is handled after the submit and the BuildGeneratorFunction adds prefixes, expressions and then the generator context is passed to the jQuery JS which triggers the final query build. This also shows how JS AJAX is utilized in order to execute these queries.

The fifth dynamic model is the Search Custom Query sequence diagram. This shows how we execute a standard SPARQL over AJAX through the help of a PHP file.

The sixth dynamic model is the Search For A Custom Query sequence diagram. This shows our interactions with MySQL for tag searching and for retrieving the queries that match our target tags.

The seventh dynamic model is Storing of a Custom Query sequence diagram. In this diagram we illustrate the storage of our queries in the MySQL Database along with the tags required to retrieve the query later on.

The eight dynamic model is the Database Explorer sequence diagram. This diagram displays the Fuseki connection and the re-encoding operations required to turn data from Fuseki into data that D3JS will accept.

Glossary

OSINT - Open-source intelligence

Cyber-attack - Any type of offensive maneuver employed by individuals or whole organizations that targets computer information systems, infrastructures, computer networks, and/or personal computer devices by various means of malicious acts usually originating from an anonymous source that either steals, alters, or destroys a specified target by hacking into a susceptible system [1]

Triple store - A triplestore is a purpose-built database for the storage and retrieval of triples through semantic queries. A triple is a data entity composed of subject-predicate-object. [2]

RDF - a family of World Wide Web Consortium (W3C) specifications originally designed as a metadata data model. It has come to be used as a general method for conceptual description or modeling of information that is implemented in web resources, using a variety of syntax notations and data serialization formats. [3]

Semantic Web - collaborative movement led by international standards body the World Wide Web Consortium (W3C). The standard promotes common data formats on the World Wide Web. By encouraging the inclusion of semantic content in web pages, the Semantic Web aims at converting the current web, dominated by unstructured and semi-structured documents into a "web of data". The Semantic Web stack builds on the W3C's Resource Description Framework (RDF). [4]

SPARQL - an RDF query language, that is, a semantic query language for databases, able to retrieve and manipulate data stored in Resource Description Framework format. [5]

Appendix

Appendix A - Complete use cases

Name:

Store Data

Participants:

User
Events:
1. From the homepage of the website, the user must click on the “Contribute Now” button.
2. The user will be taken to a form which must be filled out.
3. On the form the user may enter his/her name or leave it blank.
4. The user must fill out the prefixes textbox with the prefixes they will be using for their properties.
5. The user must enter the subject of the data he is about to enter in the “Subject of Lead” textbox
6. The user must been enter at least one property of the subject by selecting a property from the properties drop down selecting the appropriate prefix in the prefix drop down and entering the value of the property in the textbox next to it.
7. If the user wants to submit more than one property they may click on the “+” button to add more property segments . If they want to remove any extra property input segment they can click on the “-” button.
8. The user clicks on the “Submit Lead” button.
9. The user will be taken to the Data Entry form and a message will appear on the top of the page notifying the user if their data was accepted or not.

Alternative Events:
- At step 8, if the user clicks the button and any of the data is invalid then an exception will occur that notifies the user there is invalid data.
- At step 1, or step 8, if the user clicks on the button and the system is down an exception will occur which will tell the user the system is currently down.

Entry Conditions:
User has data that they want to submit
User must be at homepage of the system
Exit Condition:
Data has been stored in the triple store and the user receives acknowledgment of it.
Exceptions:
Data that was input was invalid
The system is down.

Name:
Query Data
Participants:
User
Events:
1. From the home page, the user must click on the “Search Now!” button which will take them to the submit query form.
2. In the submit query form the user must enter the prefixes they will be using in the prefix textbox.

3. The user must then fill out one input segment by selecting at least one property from the drop down box along with an appropriate prefix from the prefix drop down and must enter a value in the text box.
4. If the user wants to submit more than one property they may click on the “+” button to add more property segments . If they want to remove any extra property input segment they can click on the “-” button.
5. The user must then click on the “Search Database” button.
6. The user will be taken to a page that has results of the query they submitted, if any exist.

Alternative Events:

- Instead of having the system generate a query for them, a user may submit their own query by selecting the “Custom Search” button, which will take them to the custom search form.
- At step 5, if the user clicks the button and any of the data is invalid then an exception will occur that notifies the user there is invalid data.
- At step 6, if no results are generated from the query that the user has used then an exception will occur which tells the user there are no results.
- At step 1, or step 5, if the system is down and the user goes through one of these steps then an exception will occur which will notify the user that the system is currently down

Entry Conditions:

User must be at homepage of the system

Exit Condition:

User has successfully entered a query and has gotten a list of results.

Exceptions:

User enters invalid data.

There are no results for the query specified by the user.

The system is down.

Name:

Use a custom query

Participants:

User

Events:

1. From the data retrieval page, the user clicks on the “Custom Search” button. This will take the user to the custom search page.
2. At the custom query page, the user will enter a query that they have generated themselves into the textbox provided.
3. The user will click on the “Execute Query” button
4. If the query is valid and produces results then the user will be taken to a results page.

Alternative Events:

- At step 2, If the user wants to use a query that another user has saved instead of one they generated themselves, then the user will enter the appropriate tags for that query in the tags textbox then click on “Find Queries”.

- At step 3, if the user has entered an invalid query then an exception will occur notifying the user that their query is malformed.
- At step 1, or 3, if the user follows these steps and the system is down an exception will occur notifying the user that the system is currently down.
- At step 4, if the user's query provides no results then an exception will occur and the user will be notified that their query produced no results.

Entry Conditions:
The user must be at the data retrieval page
The user must have a query or the user must know the tags of a query that has been saved
Exit Condition:
User has entered a valid query and has gotten a list of results the query generated
Exceptions:
User entered an invalid query.
The query has returned no results.
The system is down.

Name:
Saving a custom query
Participants:
User
Events:
1. From the data retrieval page, the user clicks on the "Custom Search" button. This will take the user to the custom search page.
2. At the custom search page the user will enter the query that they want to store and enter the tags that will be used to identify that query in the tags textbox.
3. The user will click on the "Add Query" button.
4. The user will receive a message telling them whether the query they submitted was saved.

Alternative Events:
- At step 1, or step 3, if the system is down then an exception will occur and the user will be notified that the system is currently down.

Entry Conditions:
The user must be at the data retrieval page
The user must have a query and the tags he wants to save that query under
Exit Condition:
The query has been saved into the system.
Exceptions:
The system is down.

Name:
Import Data
Participants:
User
Events:

1. From the data entry page, the user will click on the “Bulk Import Data” button.
2. When the button is clicked, a new window will open asking the user to select the file they want to import.
3. The user selects the file that they want to import.
4. The file will be processed entry by entry and valid entries will be added to the database.

Alternative Events:

- At step 3, if the user selects a file that is invalid the web page will refuse the file and do nothing.
- At step 4, if there is an invalid entry then the page will tell the user that an entry has failed to be inserted.

Entry Conditions:
The user must be at the data retrieval page.
The user must have a file that they want to import in the correct xml format.
Exit Condition:
The data has been imported into the system.
Exceptions:
The system is down.
File in unsupported format.

Name:
Bulk Data Export
Participants:
User
Events:

1. At the data retrieval page, the user must click on the “Bulk Export Database” button.
2. Once the button has been pressed the user will download a file containing all the data that is currently in the database.

Alternative Events:

- N/A

Entry Conditions:
The user must be at the data retrieval page.
The user must be able to save a file to their local machine.
Exit Condition:
The user has downloaded a file with all the data in the database.
Exceptions:
The system is down.

Name:
Database Visualizer
Participants:
User
Events:

1. From the homepage, the user will click on the “Explore our databse” tab at the top of the page.
2. Once the user has pressed the tab, then the user will be moved to the visualizer page.

3. At the visualizer page, a graph will be constructed using all the data that is currently in the database and the page will display the graph.

Alternative Events:

- At step 3, if there is no data in the database then a graph showing two null points and a line connecting them will be displayed.

Entry Conditions:

The user must be on the homepage of the system.

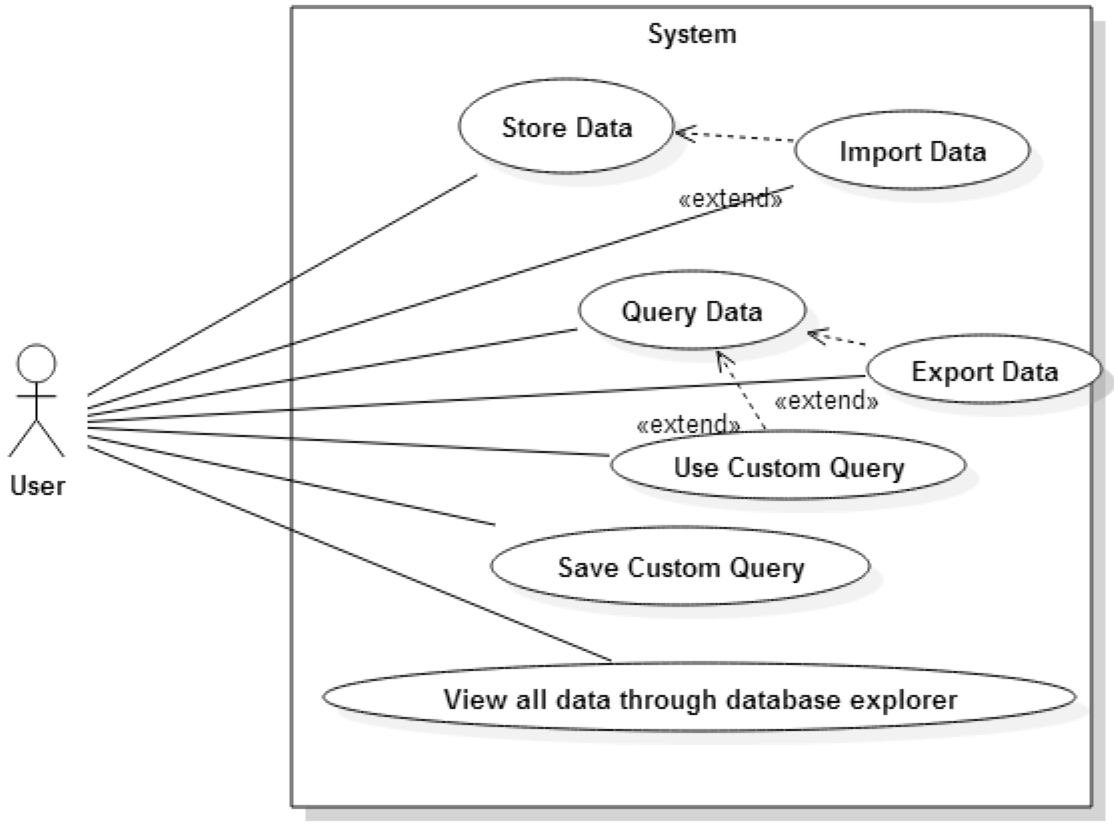
Exit Condition:

The user will be able to see a graph with all the data in the database.

Exceptions:

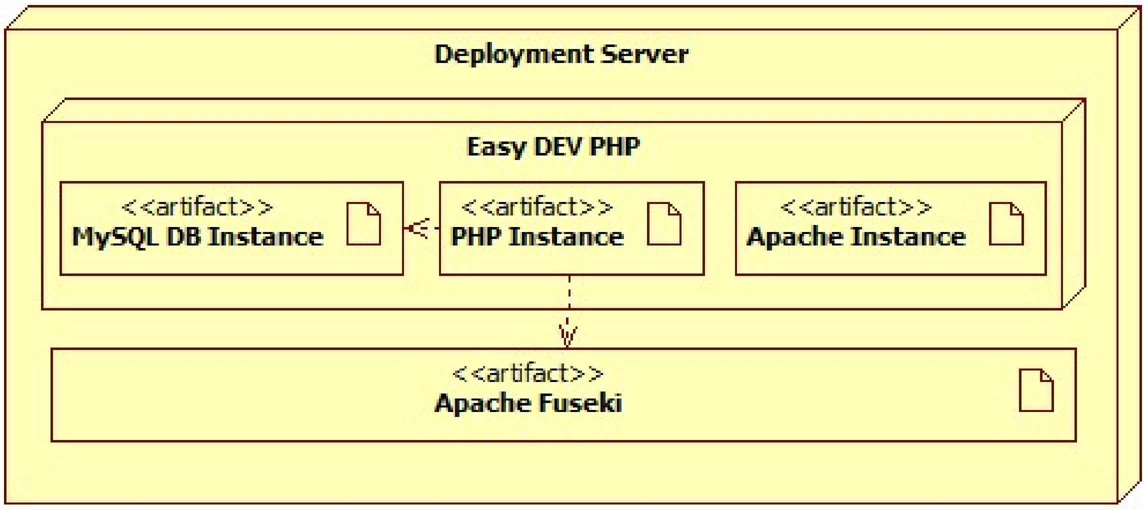
The system is down.

Appendix B - Use case diagram using UML

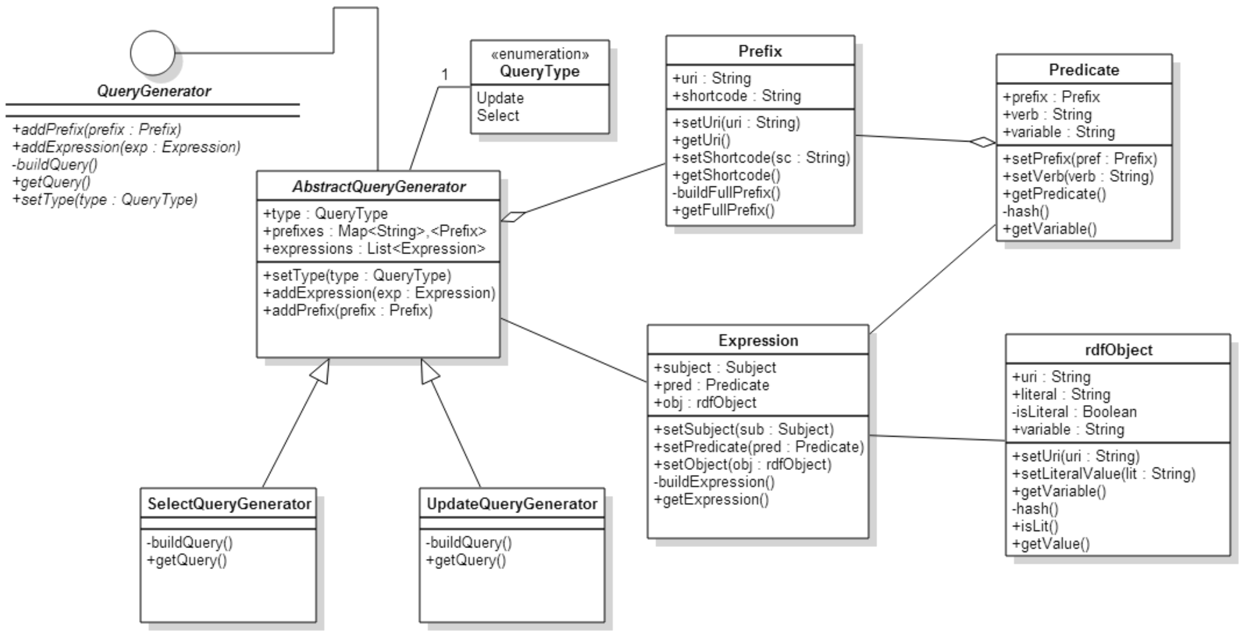


Appendix C - Static UML diagram

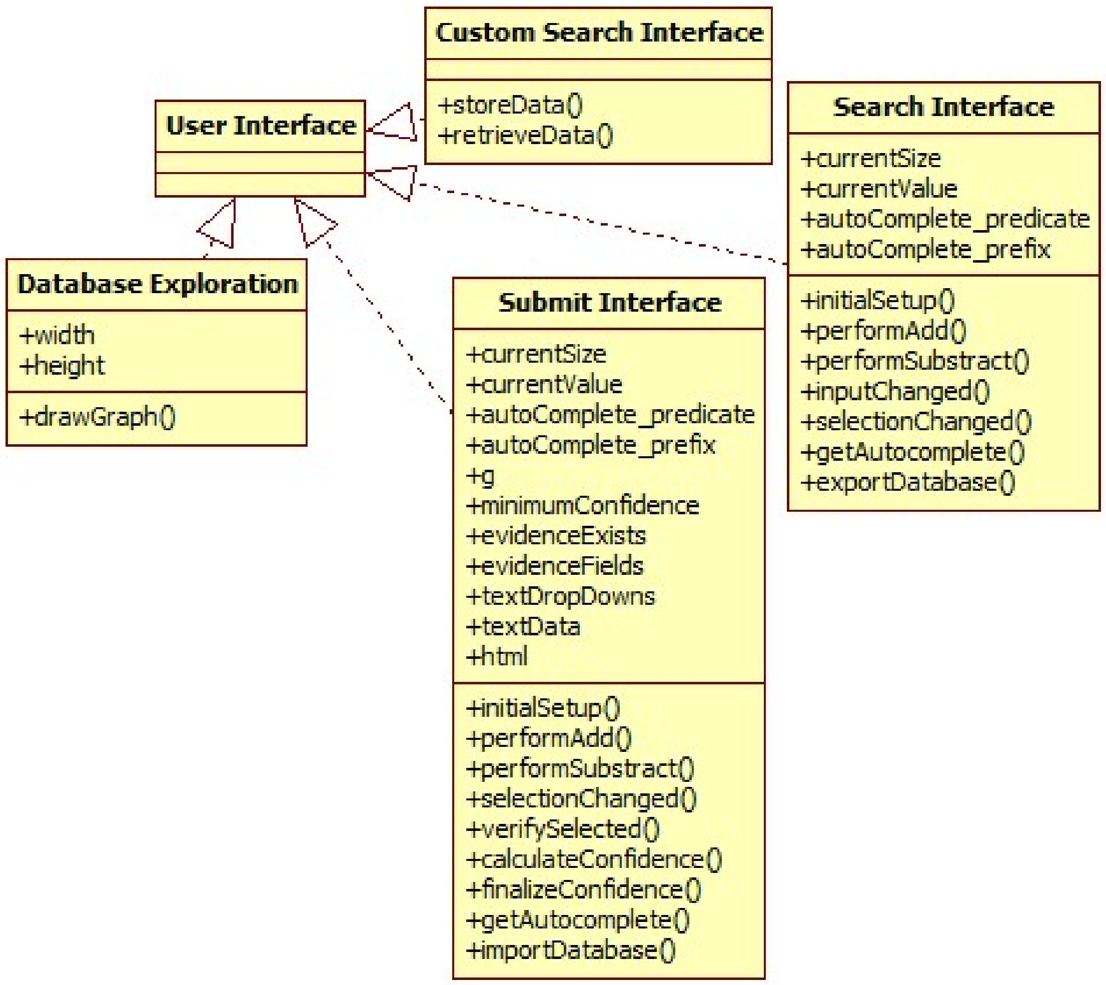
Deployment Diagram


Deployment Server
Easy DEV PHP
<<artifact>> MySQL DB Instance
<<artifact>> PHP Instance
<<artifact>> Apache Instance
<<artifact>> Apache Fuseki

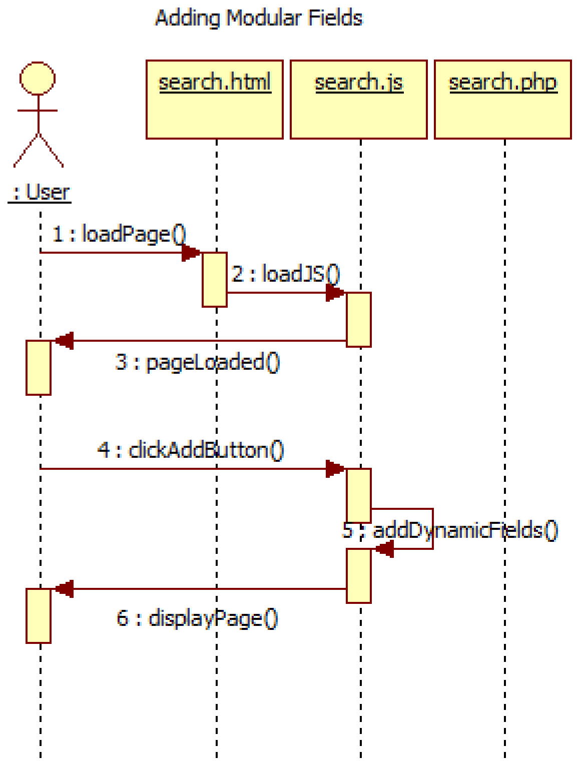
QueryJS library class diagram

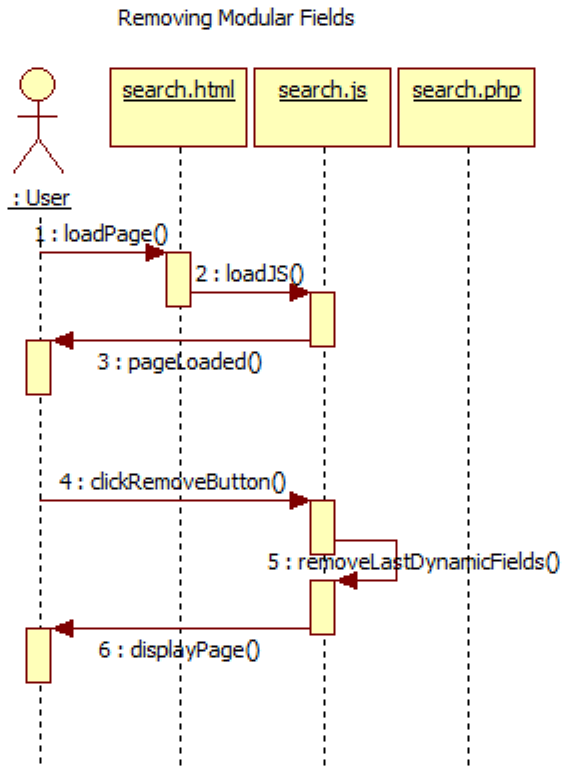


UI Class Diagrams

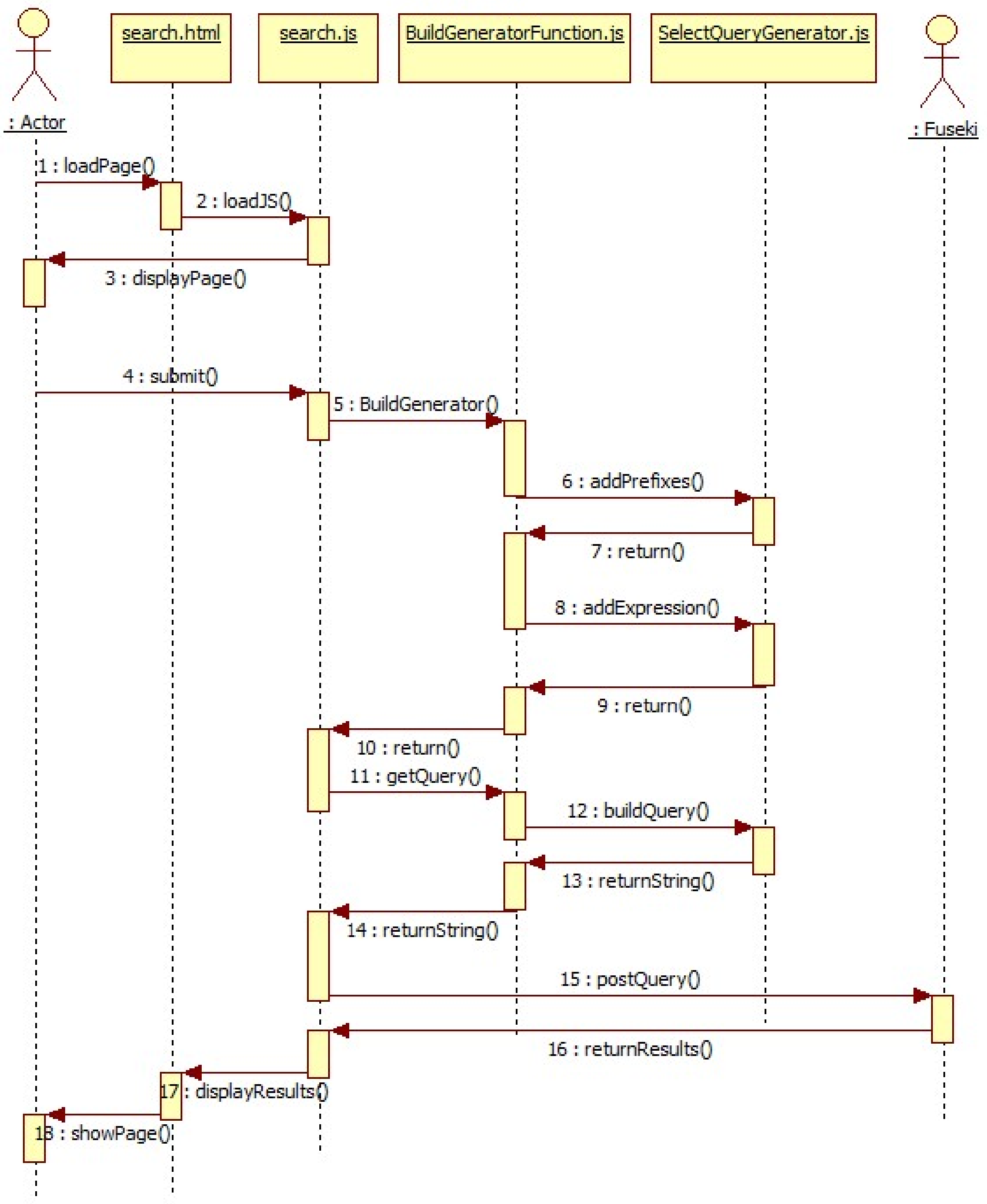


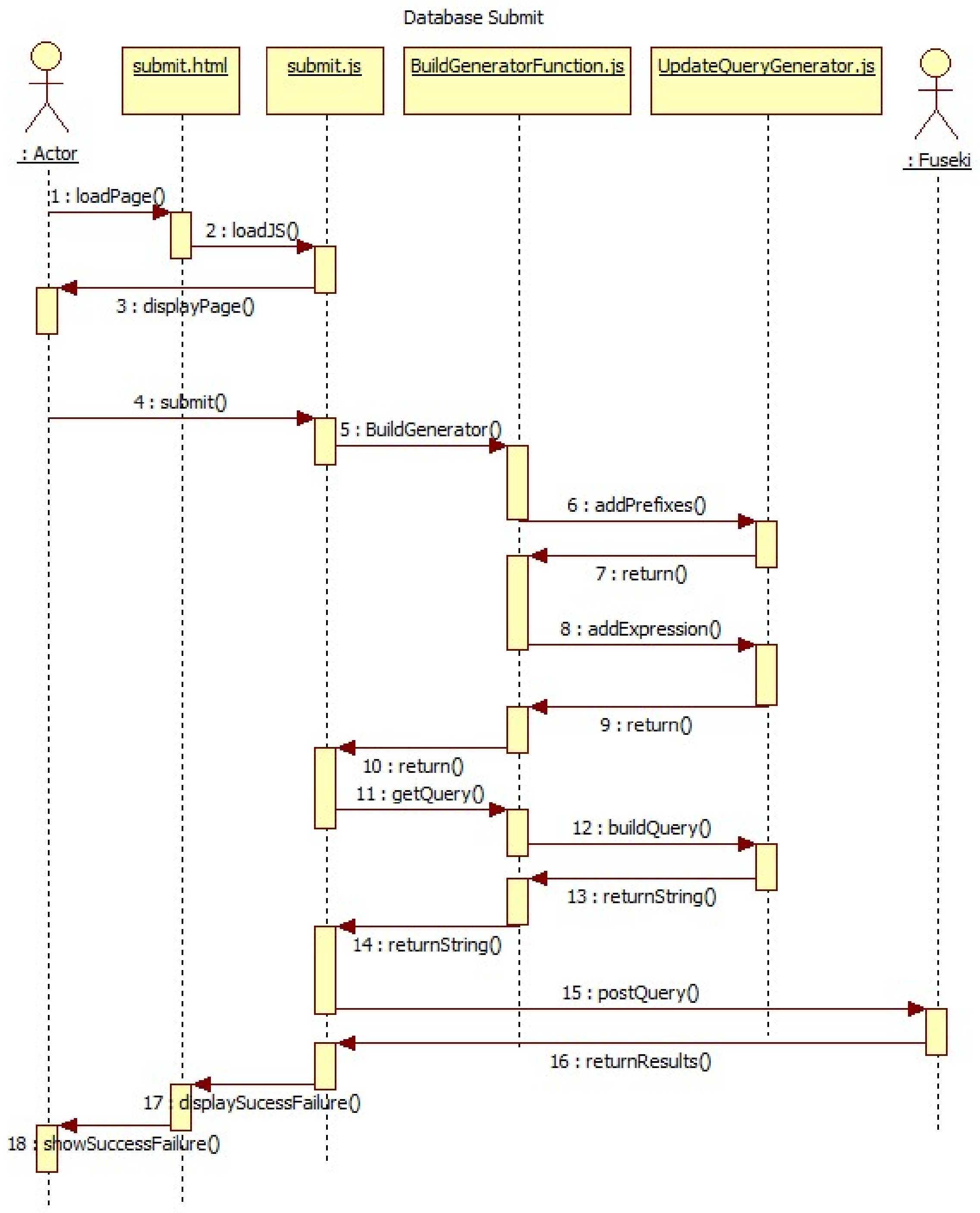
Appendix D - Dynamic UML diagrams



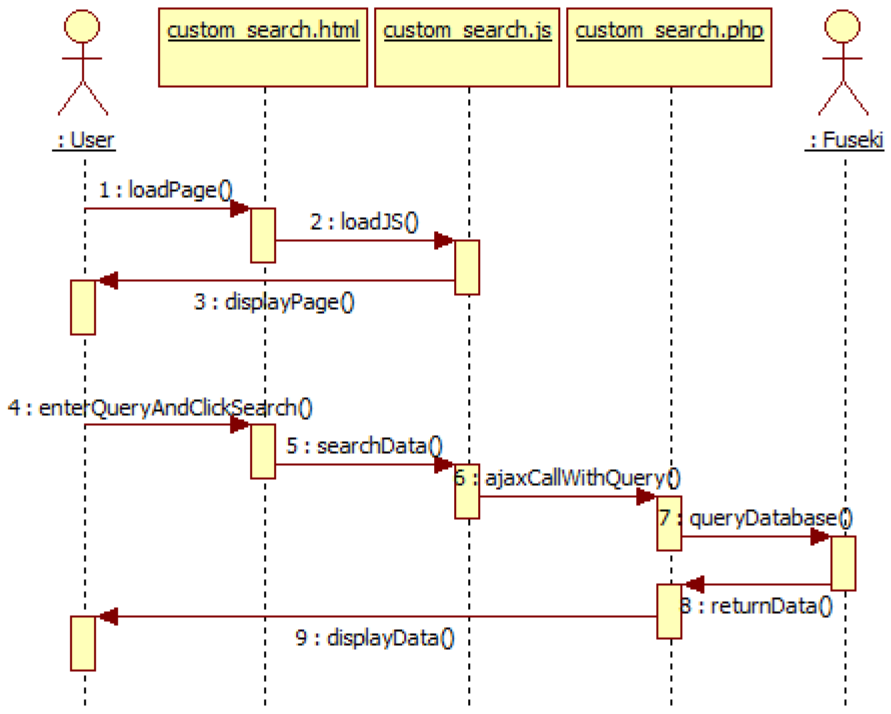


Database Search

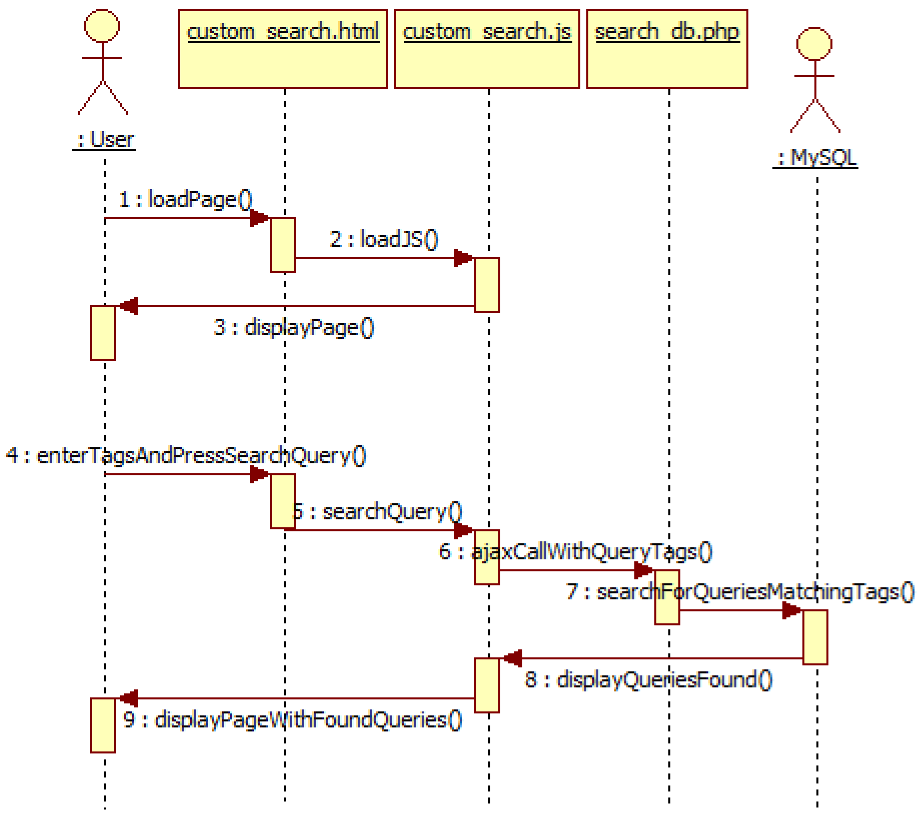



Database Submit
: Actor
submit.html
submit.js
BuildGeneratorFunction.js
UpdateQueryGenerator.js
: Fuseki
1 : loadPage()
2 : loadJS()
3 : displayPage()
4 : submit()
5 : BuildGenerator()
6 : addPrefixes()
7 : return()
8 : addExpression()
9 : return()
10 : return()
11 : getQuery()
12 : buildQuery()
13 : returnString()
14 : returnString()
15 : postQuery()
16 : returnResults()
17 : displaySucessFailure()
18 : showSuccessFailure()

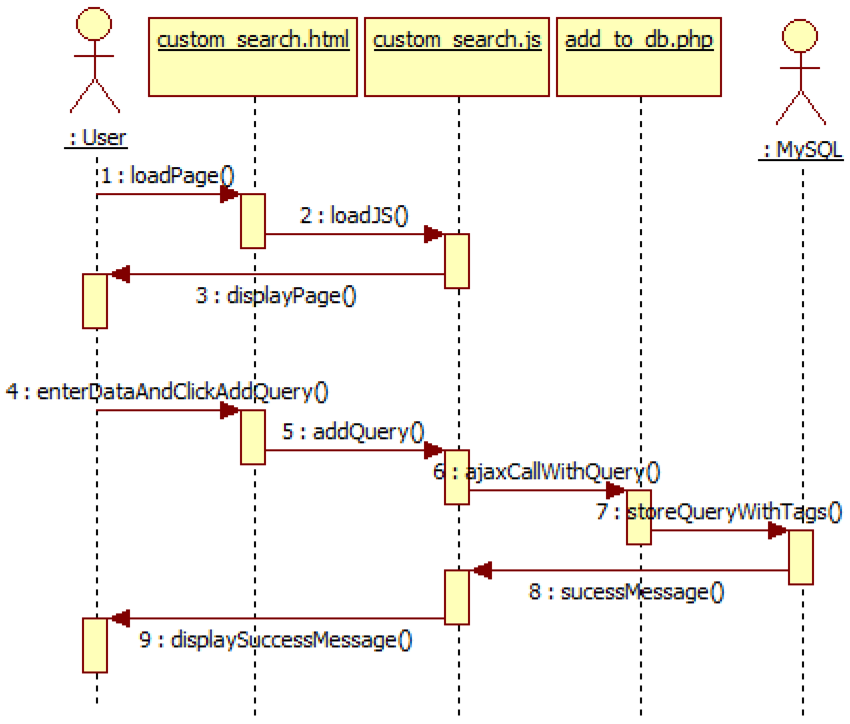
Search Custom Query



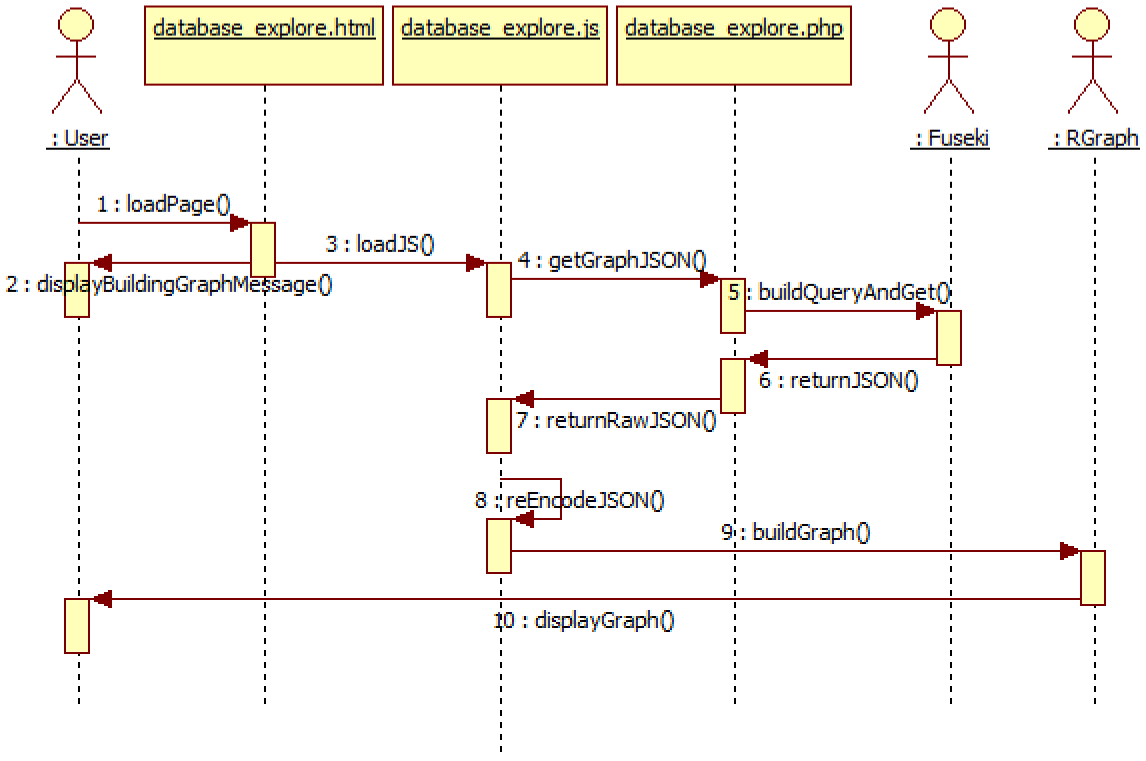
Search For A Custom Query


: User
custom_search.html
custom_search.js
search_db.php
: MySQL
1 : loadPage()
2 : loadJS()
3 : displayPage()
4 : enterTagsAndPressSearchQuery()
5 : searchQuery()
6 : ajaxCallWithQueryTags()
7 : searchForQueriesMatchingTags()
8 : displayQueriesFound()
9 : displayPageWithFoundQueries()

Storing Custom Query

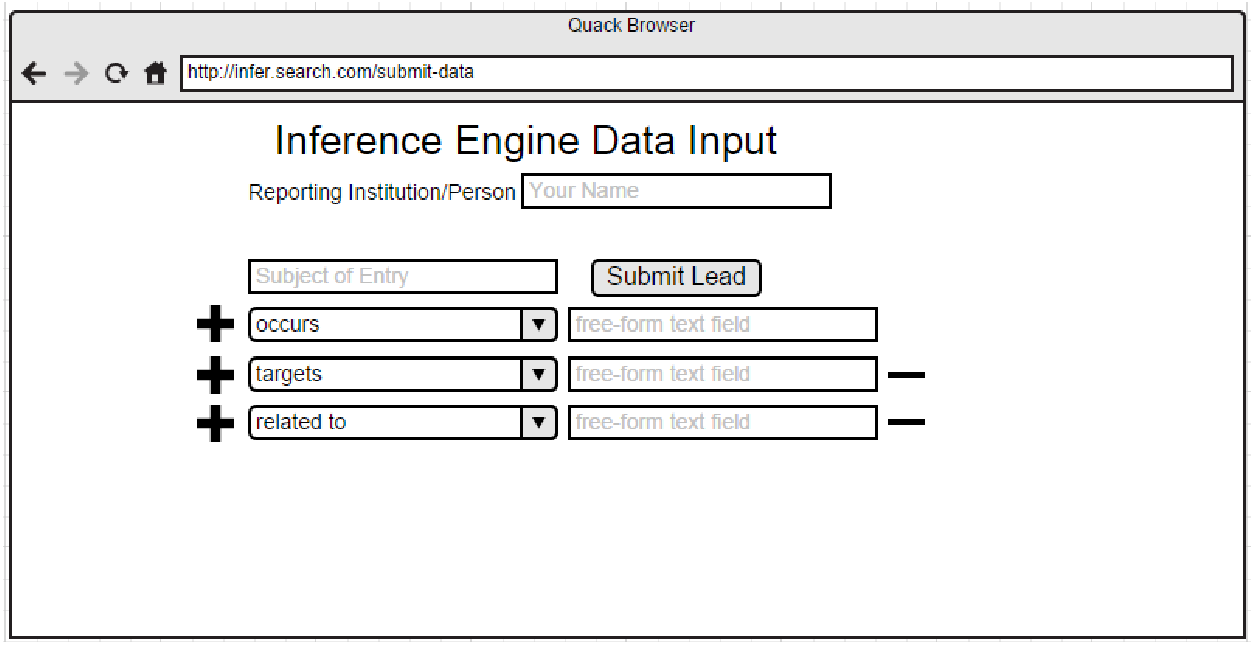


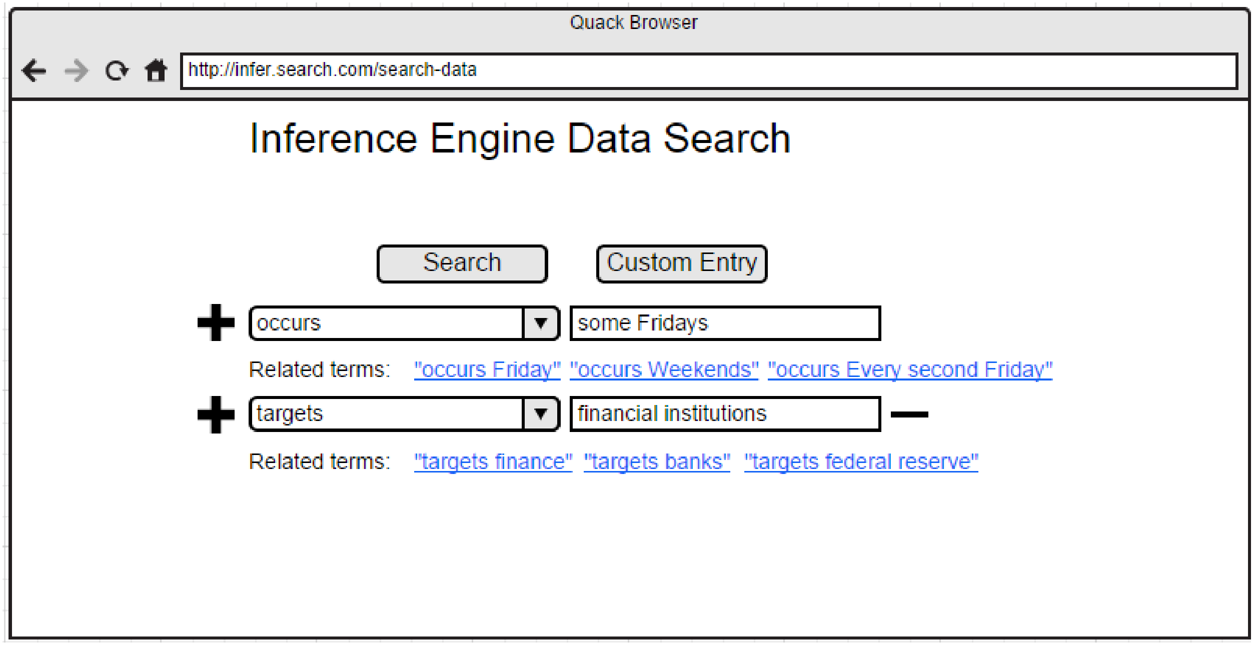
Database Explorer

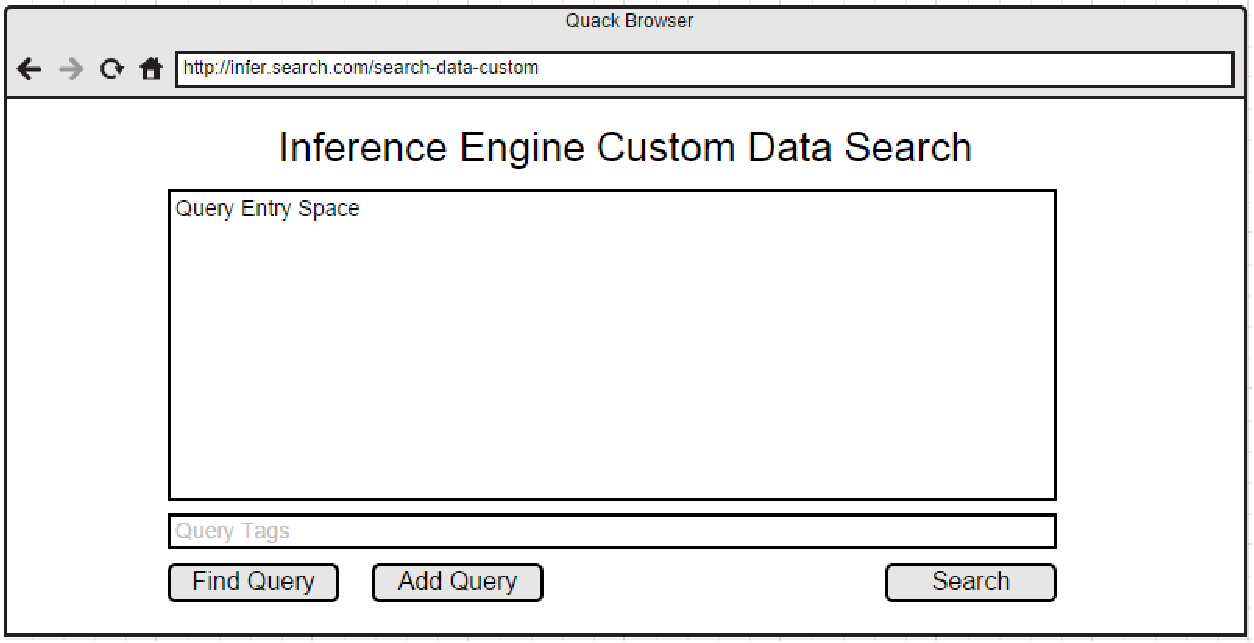


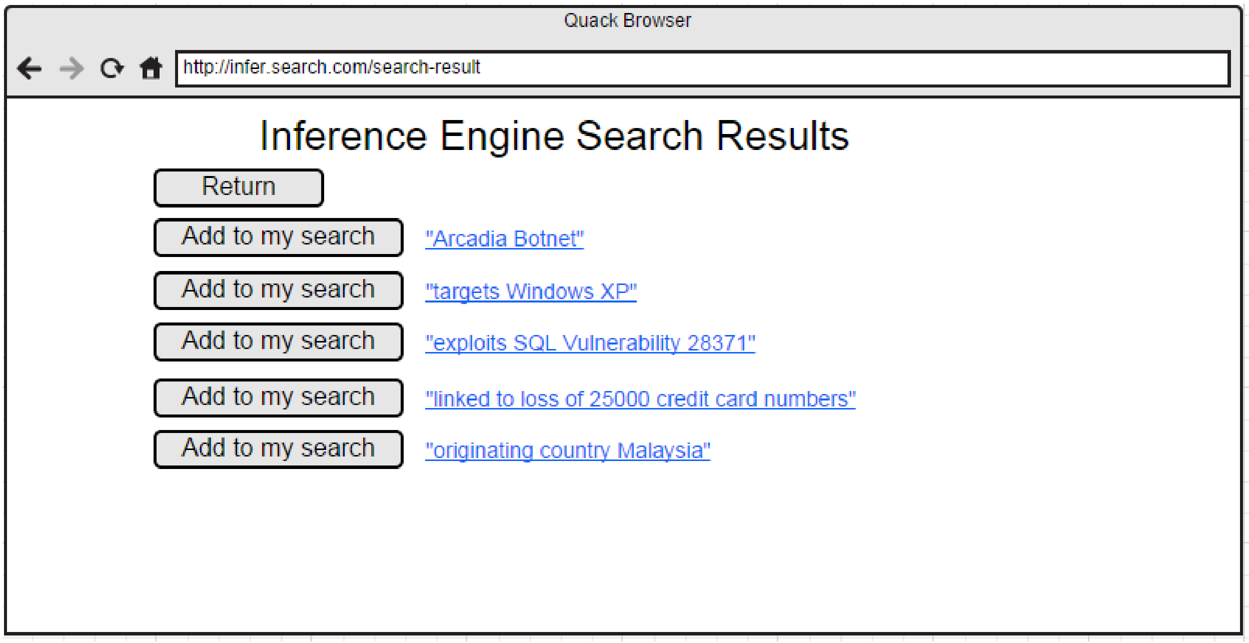
Appendix E - User Interface designs.

Mocks Ups:




Quack Browser
http://infer.search.com/search-data
Inference Engine Data Search
Search
Custom Entry
occurs
some Fridays
Related terms: "occurs Friday" "occurs Weekends" "occurs Every second Friday"
targets
financial institutions
Related terms: "targets finance" "targets banks" "targets federal reserve"


Quack Browser
http://infer.search.com/search-data-custom
Inference Engine Custom Data Search
Query Entry Space
Query Tags
Find Query
Add Query
Search


Quack Browser

http://infer.search.com/search-result

Inference Engine Search Results

Return

Add to my search "Arcadia Botnet"

Add to my search "targets Windows XP"

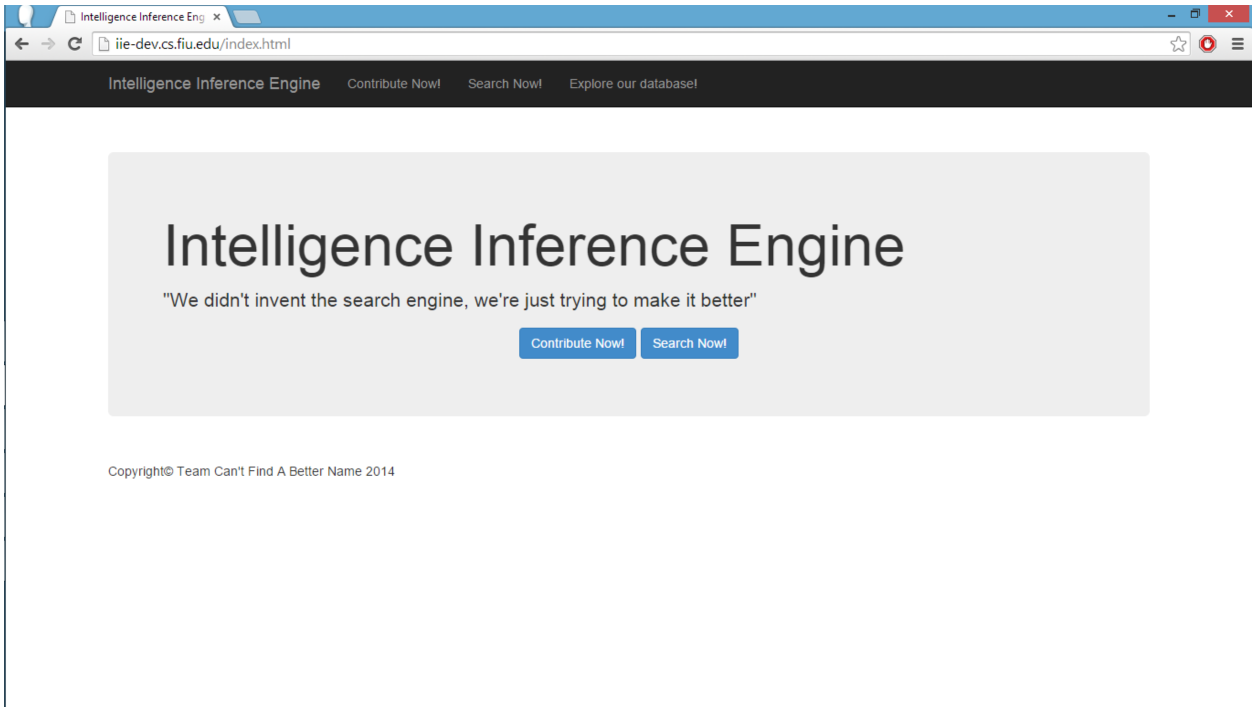
Add to my search "exploits SQL Vulnerability 28371"

Add to my search "linked to loss of 25000 credit card numbers"

Add to my search "originating country Malaysia"

Actual:

Home Page


Intelligence Inference Eng

iie-dev.cs.fiu.edu/index.html

Intelligence Inference Engine | Contribute Now! | Search Now! | Explore our database!

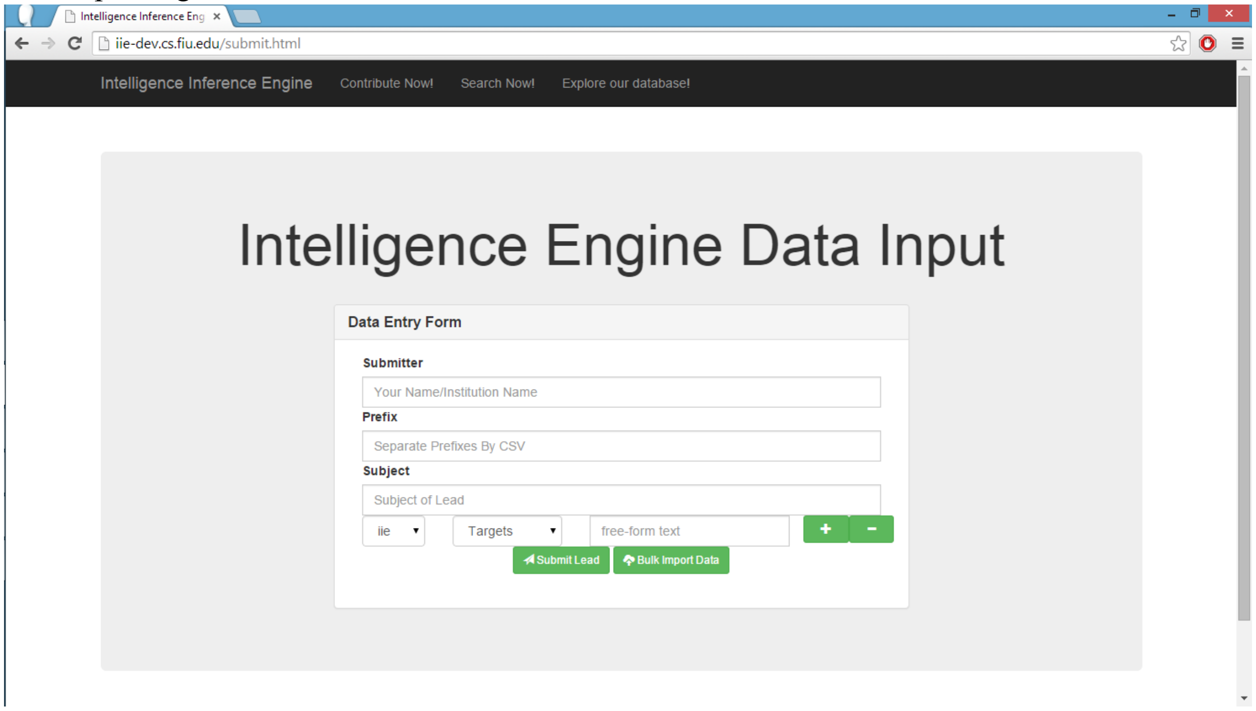
Intelligence Inference Engine

"We didn't invent the search engine, we're just trying to make it better"

Contribute Now! | Search Now!

Copyright© Team Can't Find A Better Name 2014

Data Input Page



Intelligence Inference Engine | Contribute Now! | Search Now! | Explore our database!

Intelligence Engine Data Input

Data Entry Form

Submitter: Your Name/Institution Name

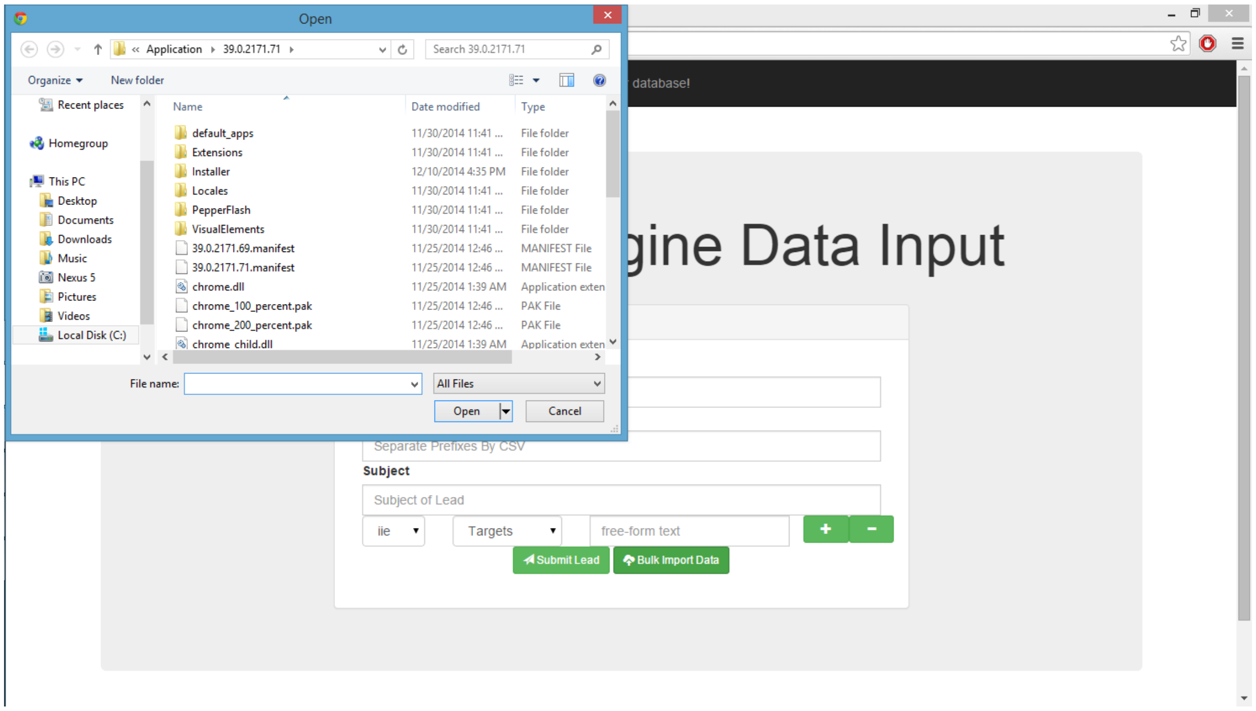
Prefix: Separate Prefixes By CSV

Subject: Subject of Lead

iie | Targets | free-form text | + | −

Submit Lead | Bulk Import Data

Data Import Dialog



Open

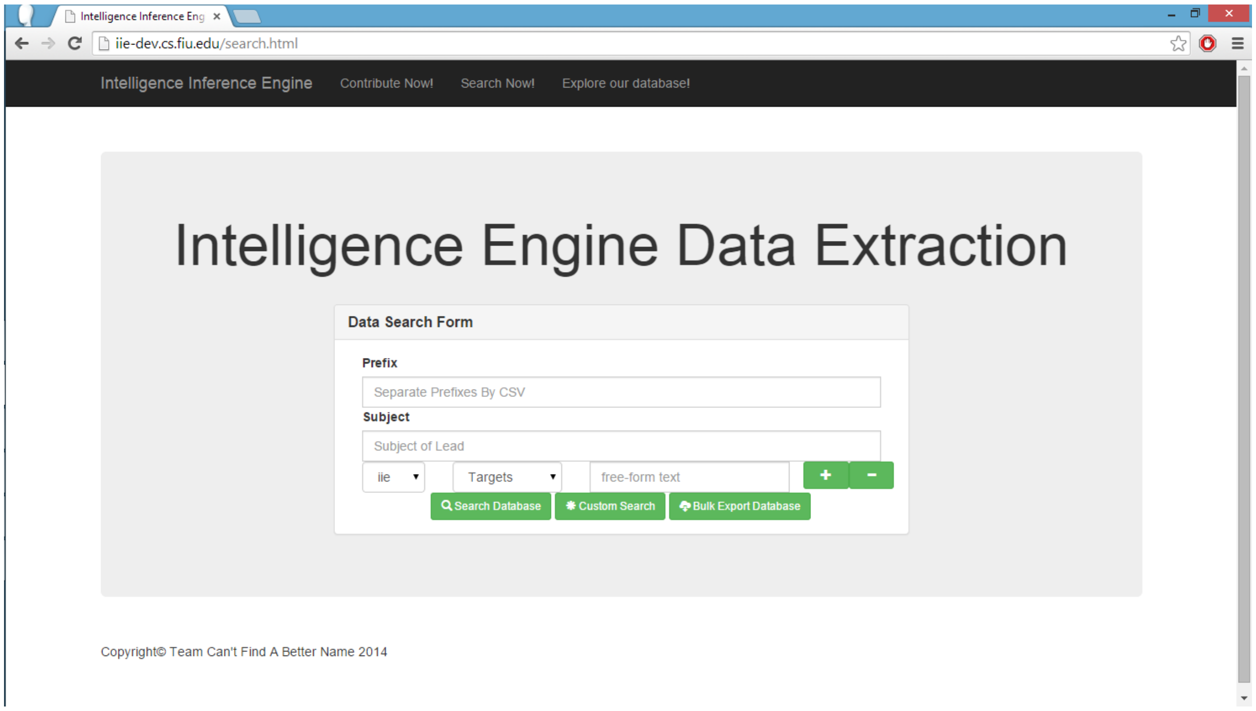
« Application › 39.0.2171.71 ›

Name	Date modified	Type
default_apps	11/30/2014 11:41 ...	File folder
Extensions	11/30/2014 11:41 ...	File folder
Installer	12/10/2014 4:35 PM	File folder
Locales	11/30/2014 11:41 ...	File folder
PepperFlash	11/30/2014 11:41 ...	File folder
VisualElements	11/30/2014 11:41 ...	File folder
39.0.2171.69.manifest	11/25/2014 12:46 ...	MANIFEST File
39.0.2171.71.manifest	11/25/2014 12:46 ...	MANIFEST File
chrome.dll	11/25/2014 1:39 AM	Application exten
chrome_100_percent.pak	11/25/2014 12:46 ...	PAK File
chrome_200_percent.pak	11/25/2014 12:46 ...	PAK File
chrome_child.dll	11/25/2014 1:39 AM	Application exten

File name: | All Files

Open | Cancel

Data Search Page


Intelligence Inference Eng
iie-dev.cs.fiu.edu/search.html
Intelligence Inference Engine
Contribute Now!
Search Now!
Explore our database!

Intelligence Engine Data Extraction

Data Search Form

Prefix

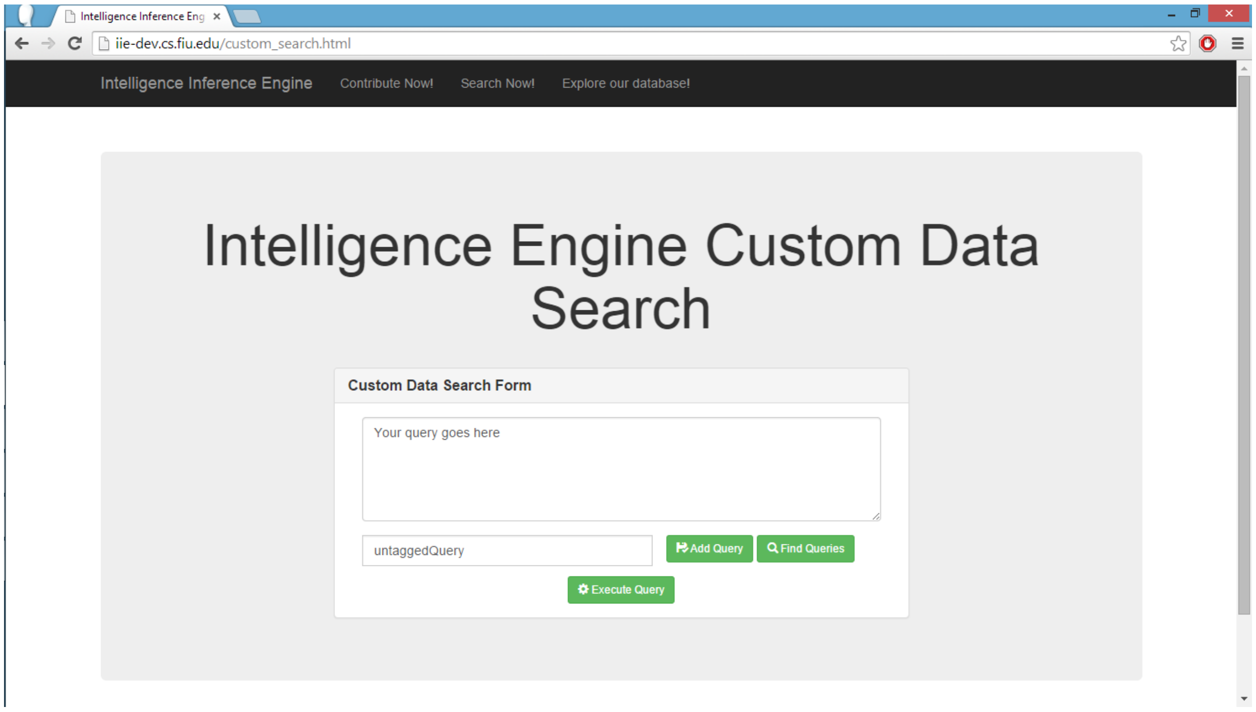
Separate Prefixes By CSV

Subject

Subject of Lead

Copyright© Team Can't Find A Better Name 2014

Custom Search Page


Intelligence Inference Eng
iie-dev.cs.fiu.edu/custom_search.html
Intelligence Inference Engine
Contribute Now!
Search Now!
Explore our database!

Intelligence Engine Custom Data Search

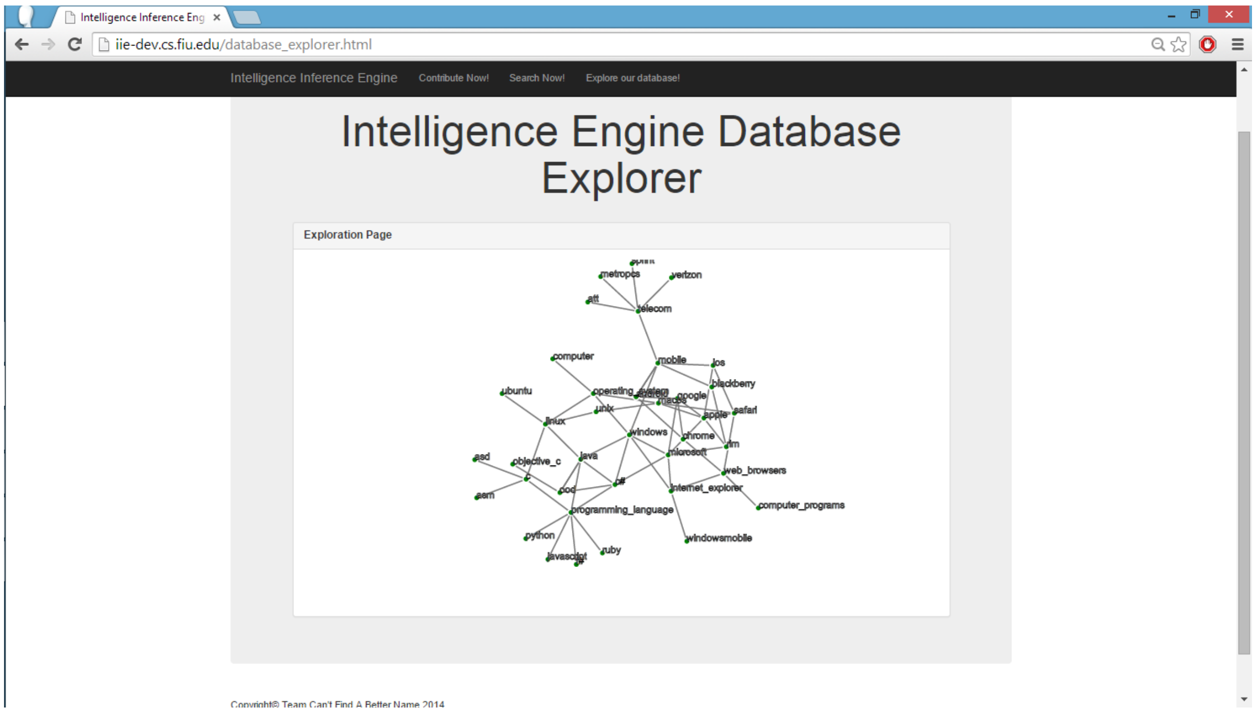
Custom Data Search Form

Your query goes here

untaggedQuery

Add Query
Find Queries
Execute Query

Data Exploration Page



Intelligence Engine Database Explorer

Copyright© Team Can't Find A Better Name 2014

Appendix F - Diary of meeting and tasks.

Date: 9/3/2014

Meeting Begins: 6:03PM

Meeting Ends: 6:45PM

Medium of Communication: Skype

Present Members: Eric, Jose, Lazaro

6:03PM - 6:15PM Eric (Formal Introductions, Intro to Inference Engine)

6:15PM - 6:30PM Use Case Elicitation

6:30PM - 6:33PM Architecture concerns and discussion

6:33PM - 6:33PM Meeting with client formally disbands

6:33PM - 6:45PM Jose and Lazaro discuss required documentation, recap about requirements.

Assignments:

Lazaro and Jose will install Mulgara and attempt to execute basic queries, we'll also be looking at bootstrap-based approach to generating simple queries from a web form.

Lazaro and Jose will also begin writing the required initial drafts for our Feasibility Study, Requirements Document and Project Plan.

Date: 9/6/2014

Meeting Begins: 11:30AM

Meeting Ends: 6:00PM

Medium of Communication: Physical Meeting

Present Members: Jose, Lazaro

11:30AM - 1:30PM: Trello Board Upgrades and Discussion

1:30PM - 4:30PM: Feasibility Study Documentation

4:30PM - 5:30PM: Requirements Documentation

Date: 9/21/2014

Meeting Begin: 10:00 AM

Meeting Ends: 3:00 PM

Medium: Physical Meeting

Present members: Jose, Lazaro

In this meeting Lazaro started work on implementing the UI of the website. The initial UIs were done. He also installed Mulgara and testing it to make sure it worked. Jose begin looking for a library to do the REST calls that the UI needed to make in order to make queries. He found a library and started testing it to make sure that it had everything that was needed for the project

Date: 10/4/2014

Begin : 11:00 AM

End : 4:00 PM

Medium : Physical Meeting

Present Members : Jose, Lazaro

In this meeting Lazaro worked mainly on getting the confidence ranking to show up on the forms when evidence was added. Apart from that, he also included a lot of bug fixes for code that was previously submitted. Jose worked on the PHP scripts that are going to generate SPARQL queries based on what users entered in the forms. He also worked on some bug fixes for the scripts.

Date: 10/19/2014

Begin : 11:00 AM

End : 4:00 PM

Medium : Physical Meeting

Present Members : Jose, Lazaro

In this meeting Lazaro began working on the custom query subsystems of the system. He also continued to work on the previous forms. Jose continued testing and developing the PHP scripts so that they generated valid SPARQL queries and correctly connected to the server.

Date: 10-28-2014

Meeting Begins: 4:30PM

Meeting Ends: 5:15PM

Medium of Communication: Skype

Present Members: Jose, Eric

4:30 - 4:40 Discussed what parts of the system I was showing today

4:40 - 4:50 Talked about the data entry form and how predicates worked

4:50 - 4:55 Discussed how predicates currently worked and changes to be made

4:55 - 5:05 Talked about the data search form and some specific queries

5:10 - 5:15 Talked about some good ontologies to use

Date: 10-31-2014

Meeting Begins: 11:30

Meeting Ends: 1:30

Medium of Communication: Physical Meeting

Present Members: Jose, Eric, Lazaro

11:30 - 11:50 Tour of Eric's workplace

11:50 - 12:20 Lazaro showed Eric some of the features of the system and asked for comments

12:20 - 12:30 Eric talked to us about how he would like to be implemented

12:30 - 12:50 We discussed about some good ontologies to use and which default ontologies we should use

12:50 - 12:55 Jose asked about the web crawler and if Eric has any recommendations

12:55 - 1:10 Eric explained what he wanted the crawler to do as we thought it had a different function

1:10 - 1:30 We got some minor features that Eric wanted us to implement and also got comments on the current system

Date:11/1/2014

Begin : 11:00 AM

End : 4:00 PM

Medium : Physical Meeting

Present Members : Jose, Lazaro

In this meeting, we discussed the need to change our core technology. We decided it was necessary to replace Mulgara with something that supported everything that was necessary for the project. Lazaro continued work on the front end forms. Jose continued work on the scripts and modified them for the new back end triple store that was going to be used.

Date: 11/17/2014

Begin : 11:00 AM

End : 4:00 PM

Medium : Physical Meeting

Present Members : Jose, Lazaro

Jose modified the PHP scripts so that they returned XML and prepared the PHP scripts to be retired. He also began work on the javascript library that were going to replace the PHP script for query generation. Lazaro worked on getting auto-complete for the different text fields working. He also added the base code import and export.

Date: 12/1/2014

Begin : 11:00 AM

End : 4:00 PM

Medium : Physical Meeting

Present Members : Jose, Lazaro

Lazaro began the testing of the front end part of the system. Jose added some of the javascript files for query generation.

Date: 12/6/2014

Begin : 11:00 AM

End : 6:00 PM

Medium : Physical Meeting

Both Jose and Lazaro finalized their code for their respective parts and began integrating the parts together. Both also fixed any bugs that came up from integration in their respective parts. Both continued to test their parts and tested the integration.

References

[1] - S. Karnouskos: *Stuxnet Worm Impact on Industrial Cyber-Physical System Security.* In:*37th Annual Conference of the IEEE Industrial Electronics Society (IECON 2011), Melbourne, Australia*, 7-10 Nov 2011. Retrieved 20 Apr 2014.

[2] - Jack Rusher, Semantic Web Advanced Development for Europe (SWAD-Europe), Workshop on Semantic Web Storage and Retrieval - Position Papers

[3] - http://www.w3.org/TR/PR-rdf-syntax/ "Resource Description Framework (RDF) Model and Syntax Specification

[4] - Berners-Lee, Tim; James Hendler; Ora Lassila (May 17, 2001). "The Semantic Web". *Scientific American Magazine*

[5] - Hebeler, John; Fisher, Matthew; Blace, Ryan; Perez-Lopez, Andrew (2009). *Semantic Web Programming*. Indianapolis, Indiana: John Wiley & Sons. p. 406. ISBN 978-0-470-41801-7