Software Design Description
for the
Semantic Web Crawler

8 May 2012

Prepared for:

National University

Prepared by:

Michael Bastos
Davendra Patel
Manuel Covarrubias Jr.

Approvals

Title	Printed Name	Signature	Date
Client Representative	Dr. Appel		
Project Manager	Manual Covarrubias Jr.		
Project Leader	Davendra Patel		
Project Engineer	Michael Bastos		

Revision Changes

The revision change record will contain the revision number, date of revision, engineering change order (ECO) number, description of what was modified, added or deleted, and the individuals name responsible for the change.

Revision Change Record

Revision Number	Revision Date	ECO Number	Description of Change	Author
0	29-Mar-2012		Original	Manager/Leaders

Table of Contents

Table of Figures

1.0 Scope

1.1 System Overview

This System Design Document (SDD) specifies the design requirements necessary for the development of the Semantic Web Crawler (SWC) program. The SWC program is being developed to provide the user the capability to scan or *crawl* through Internet pages to create an index of the data that they are looking for. Creation of this capability with state of the art software will provide far greater flexibility and efficiency in user interface function development and supporting software development plus expanded opportunities to develop follow on capabilities to meet future needs and opportunities. The intended audience for this SDD consists of:

- Customer
- The SWC Development Team
- Senior Management

1.2 Glossary

This section provides the definitions of all items, abbreviations, and acronyms required to properly interpret the SDD.

1.2.1 Definitions

Cache	A location used to store data in order to have faster subsequent retrievals
Cloud Based Environment	Refers to applications and services offered over the Internet. Data Centers that offer services are referred to as the 'cloud'
Cloud S3	A simple storage service
De-couple	Two or more systems that are able to transact without being connected (coupled). The systems do not interact with each other and a decoupled system allows changes to be made to any one system without having an effect on any other system
Debug	The process of locating and fixing or bypassing bugs (errors) in a computer program code or the engineering of a hardware device
DynamoDB	A fully managed NoSQL database service that provides fast and predictable performance with seamless scalability
Encapsulate	The process of combining elements to create a new entity
Functional Model	A structured representation of the functions within the modeled

	system. Functions include activities, actions, processes, and operations
Hash	A function that maps large sets of data into smaller sets of data
Hash Value	A number generated from a string of text
Host Computer	A computer that is connected to a TCP/IP network, including the Internet, and has a unique IP address
NoSQL	A way of storing and retrieving data quickly, like a relational database except it isn't based on the mathematical relationship between tables as a traditional relational database does
Object Model	Description of an object-oriented architecture, which includes the details of the object structure, interfaces between objects, and other object-oriented features and functions
Polymorphism	A programming language's ability to process objects differently depending on their data type or class
Proxy	An online computer server that acts as an intermediary between an Internet user and their destination site
Sockets	Fundamental technology for programming software to communicate on TCP/IP networks. Sockets provide a bidirectional communication endpoint for sending and receiving data with another socket
SQL	A simple programming language used for accessing and managing data in relational databases such as SQL Server
URL	The global address of documents and other resources on the World Wide Web

1.2.2 Abbreviations and Acronyms

COTS	Commercial Off-The-Shelf
CSC	Computer Software Components
CSCI	Computer System Component Interfaces
CSU	Computer Software Units
CSV	Comma Separated Values
DNS	Domain Name Service
DSL	Digital Subscriber Line
ECO	Engineering Change Order
GB	Gigabytes

GHz	Gigahertz
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol with Secure Sockets Layer
I/O	Input/Output
IP	Internet Protocol
ISP	Internet Service Providers
IT	Information Technology
KLOC	Thousand Lines of Code
MTBF	Mean time between failure
MTTR	Mean time to repair
OOA	Object Oriented Architecture
OS	Operating System
RAM	Random Access Memory
regex	regular expression
SDD	System Design Document
SQL	Structured Query Language
SRS	Software Requirements Specification
SSL	Secure Sockets Layer
SWC	Semantic Web Crawler
TCP	Transmission Control Protocol
URL	Uniform Resource Locator

1.3 Document Overview

The sections of the SDD document describe the software designs of the SWC program.

Section 1 contains the system overview of the SWC program and lists the intended audience for the SDD document. The section also contains a glossary to provide definitions, abbreviations, and acronyms of the SDD document.

Section 2 contains a list of the references that were consulted for more in-depth information about some of the concepts in the SDD.

Section 3 provides an overview of the SWC software system, covering the general architecture, constraints, assumptions, and dependencies.

Section 4 is a detailed breakdown of the modules and classes that make up the system.
Sections 5 and 6 describe the data manipulated by the SWC; where it resides, in what format, and where it is used.

Section 7 links the modules and classes discussed in Section 4 to the corresponding requirements set forth in the SWC Software Requirements Specification (SRS).

2.0 References

Bass, Clements, Kazman, Software Architecture in Practice, 2nd Ed., April 2003

Bastos, M., Covarrubias, M., Patel, D., Semantic Web Crawler Operational Concept Document March 2012

Bastos, M., Covarrubias, M., Patel, D., Semantic Web Crawler Software Project Management Project Document March 2012

Bastos, M., Covarrubias, M., Patel, D., Semantic Web Crawler Software Requirements Specification Document April 2012

Pressman, Roger S., Project Engineering: *A Practitioner's Approach*, Seventh Edition, McGraw Hill, 2010.

Used for definitions. www.webopedia.com accessed 20 May 2012

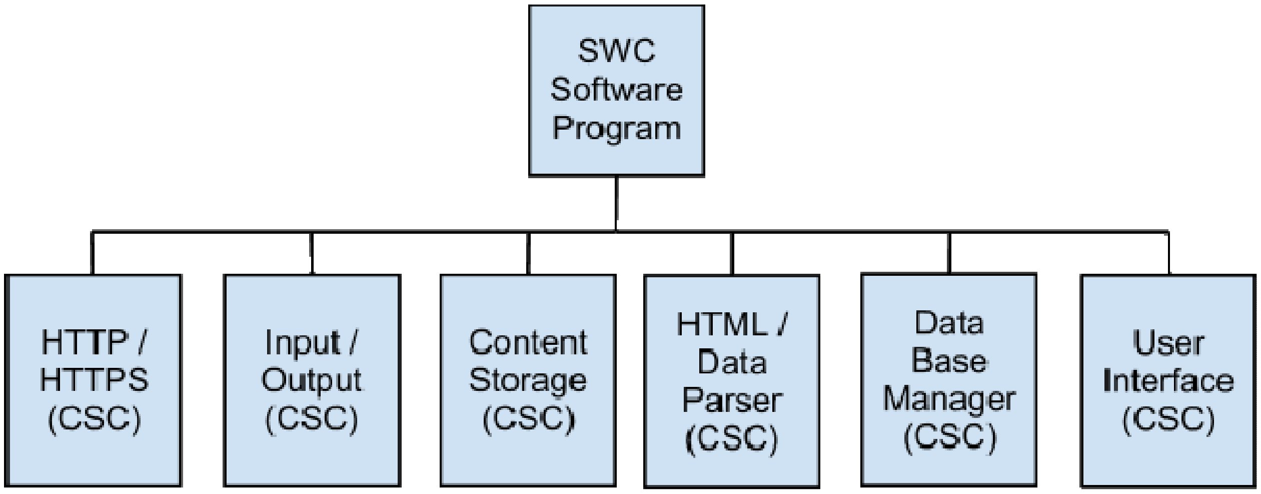
3.0 Computer System Component Interfaces (CSCI) Overview

The SWC program as the desktop Computer System Component Interfaces (CSCI). This section discusses the SWC-wide design decisions as to its behavior, from the user's point of view, in meeting requirements without discussing internal manipulation and selection and design of the software units that it consists of.

3.1 CSCI Architecture

Figure 3.1 shows the SWC program architecture.

FIGURE 3.1 SWC PROGRAM ARCHITECTURE



The SWC CSCI architecture is an Object Oriented Architecture (OOA) comprised of six Computer Software Components (CSC), each of which are implemented as separate application modules and contain subordinate Computer Software Units (CSU)s. The components encapsulate data and the operations that must be applied to manipulate the data. Communication and coordination between the components is accomplished via message passing. The individual CSCs and CSUs have separate Graphic User Interfaces (GUI)s that provide the interface between the user and the SWC program to implement activities and display reports. The following modules are available for assignment:

- Hypertext Transfer Protocol (HTTP)/Hypertext Transfer Protocol with Secure Sockets Layer (HTTPS) Crawler.
- Input/Output (I/O) Operations.
- Content Storage.
- Hypertext Markup Language (HTML) & Data Parser.
- Database Management.
- User Interface.

The following diagrams illustrate the navigational relationships between the different software components as displayed to the user on the GUIs.

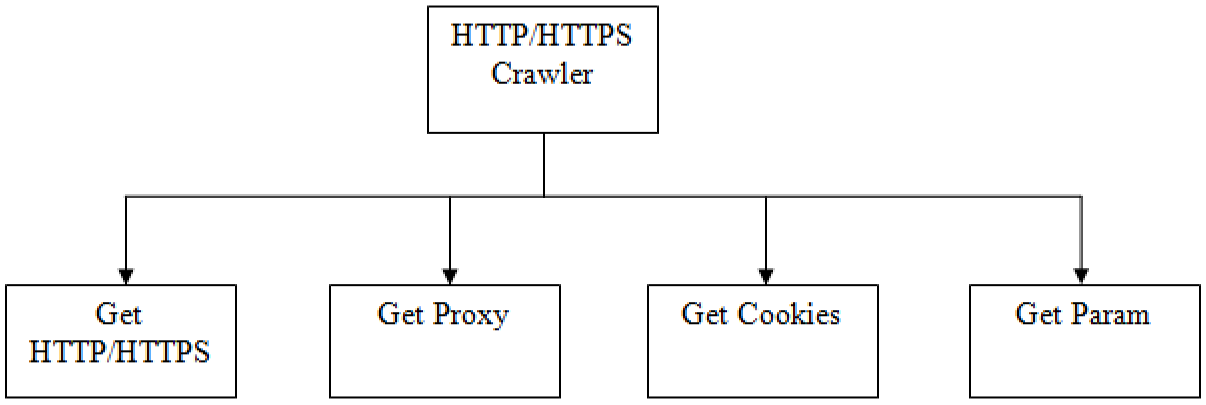
3.1.1 Functional Computer Software Components (CSC)

The functions of the CSC and CSU are summarized in the following subsections.

3.1.1.1 HTTP/HTTPS Crawler

This section is classified as the Crawler Object, its only job is interacting with the Internet and making sure that data is pulled in a quick and orderly manner. It will not try to interpret data but rather create the necessary sockets and systems necessary to merely grab it.

FIGURE 3.2 HTTP/HTTPS CRAWLER ARCHITECTURE



3.1.1.1.1 Get HTTP

This Class will basically create the necessary sockets to download a particular web page using the standard HTTP Get and Post Protocols. It will also allow for a Secure Sockets Layer (SSL) socket information so that we can download Secure HTTPS pages as well as handle the proper encryption and decryption of such pages into usable data.

3.1.1.1.2 Get Proxy

This Class will handle pushing all of the above data through the necessary ports in order to properly crawl a web page via a proxy of some kind; this process is required when a site begins to block the crawl process.

3.1.1.1.3 Get Cookies

This Class will handle anything session related such as saving a user state when visiting a web site or page throughout the crawl process. Sessions may even be saved in cache for reuse later.

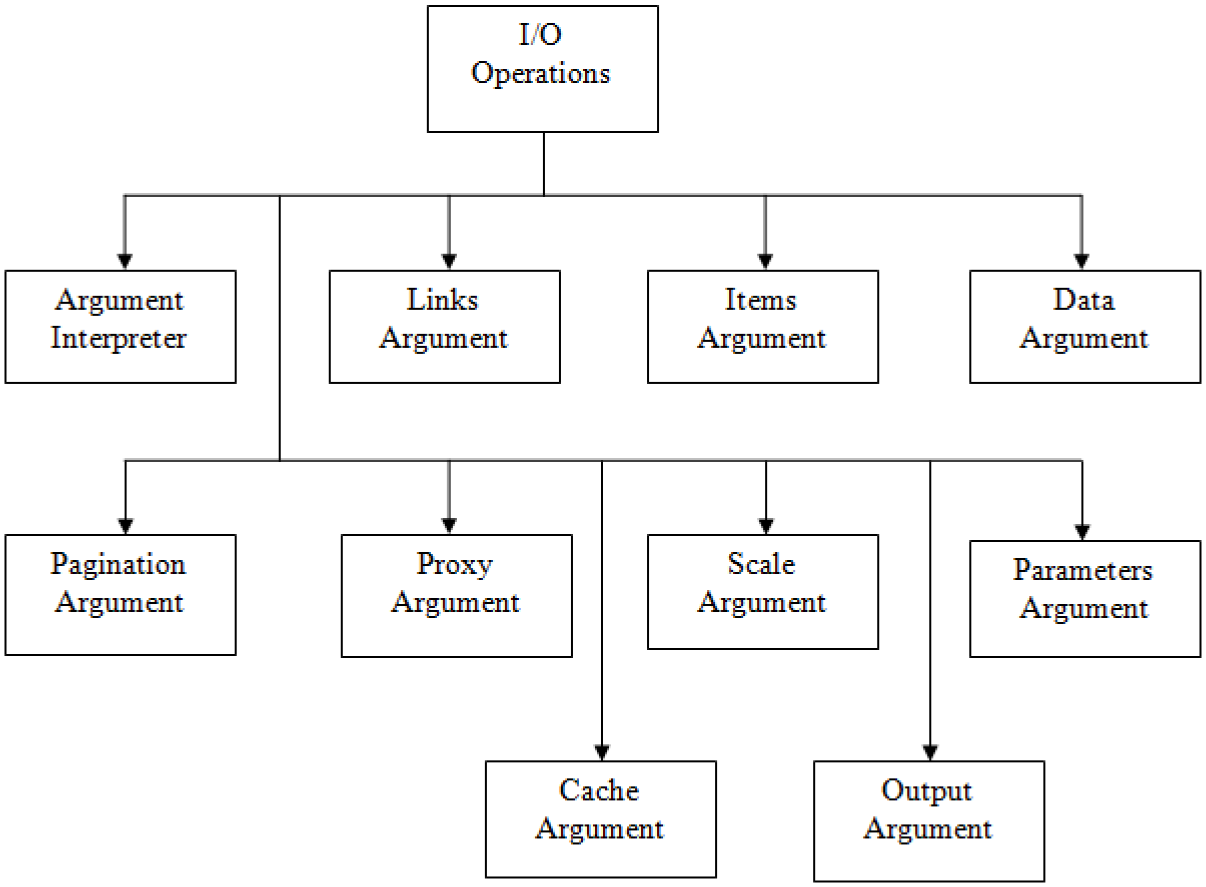
3.1.1.1.4 Get Post Param

This Class will be a bit more complicated, it will pull put together the necessary data given either through arguments or the parsing process recursively to allow for features like pagination or submit functions in the crawling process that use a post method of moving from page to page.

3.1.1.2 I/O Operations

This section is classified as the IO Object; its only job is taking in arguments and interpreting them allowing the application to understand what it needs to do in all of its other components.

FIGURE 3.3 I/O OPERATIONS ARCHITECTURE



3.1.1.2.1 Argument Interpreter

This Class will handle the most important job in the crawler, it will essentially take the interpreted arguments and give them life, it will call all the necessary Sub Classes in the Parser Class to handle not only the flow of parsing but the speed and efficiency allowing functions to become recursive when they need to be.

3.1.1.2.2 Links Argument

This Class will handle translating the arguments involved with the necessary links and what sites will be crawled by the crawler.

3.1.1.2.3 Items Argument

This Class will handle translating the arguments involved with when information is considered an item in a page with multiple data points that may need collecting. An item may have multiple data points and may require further filtering down of data in item recursively. Some form of Regular Expression will be required and given in this argument.

3.1.1.2.4 Data Argument

This Class will handle translating the arguments involved with the individual data points themselves as well as how and what name they are stored under within an item. Some form of Regular Expression will be required and given in this argument.

3.1.1.2.5 Pagination Argument

This Class will handle translating the arguments involved with the moving from one page to another and whether the movement is singular (in the case of collecting more data points for a specific item) or as a continuation of the item mass as a whole (in the case of moving to page 2 or 3 in order to collect more items).

3.1.1.2.6 Proxy Argument

This Class will handle translating the arguments involved with multiple proxer server and port information fed to the system in the event that a site tries to block the crawl, it will be smart enough to use multiple proxies if inputted to attempt a re-crawl again and again until it finds one that works.

3.1.1.2.7 Cache Argument

This Class will handle translating the arguments involved with where and how to cache the page information as well as it will allow us to determine the length of cache time so if a page is too old and needs to be re-downloaded, this argument will determine the length and age of the requirements.

3.1.1.2.8 Scale Argument

This Class will handle translating the arguments involved with allowing the crawl to scale across multiple instances; it will separate each link and its necessary arguments into those instances so that the system can quickly and effectively crawl multiple pages instantly across multiple servers.

3.1.1.2.9 Parameters Argument

This Class will handle translating the arguments involved with setting and collecting the parameters necessary to handle pagination through the Post process in the crawler download method. It will determine what parameters need to be collected and in what order they will be listed.

3.1.1.2.10 Output Argument

This Class will handle translating the arguments involved with the output of the collected data. Any formatting or database login information will be provided to the crawler at this point so that it can scale across multiple machines while saving to the database system all at the same time.

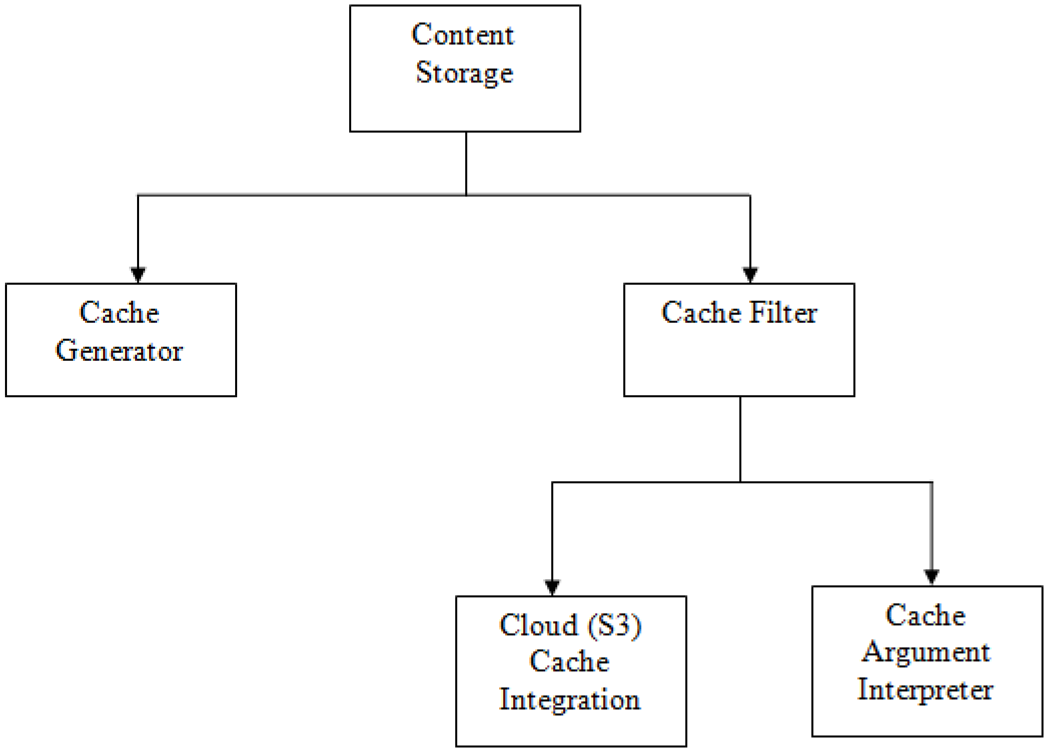
3.1.1.3 Content Storage

This section is classified as the Content Storage Object; its only job is taking the data that's downloaded from the web by the crawler and ensuring that it's properly up to date. It will save the data with a hash name of the web address the file was pulled from so that the system can quickly figure out and understand the data pulled.

3.1.1.3.1 Cache Generator

This Class will handle taking the data and hash values and making sure that they are properly saved and sorted in the cache folder. The generator's main job is to assess the age or relevance of the cached data as it relates to what needs to be parsed.

FIGURE 3.4 CONTENT STORAGE ARCHITECTURE



3.1.1.3.2 Cache Filter

This Class will handle pulling the hash data from the cache and loading it up to the necessary parser functions. The cache filter will also be able to look inside the cached information and tell the Cache Generator what it needs to determine cache relevancy.

3.1.1.3.3 Cloud (S3) Cache Integration

This Sub Class will handle integrating the cached environment with Cloud solutions such as S3 and DynamoDB; it will allow the quick and efficient processing of data across cloud systems for the purpose of scalability.

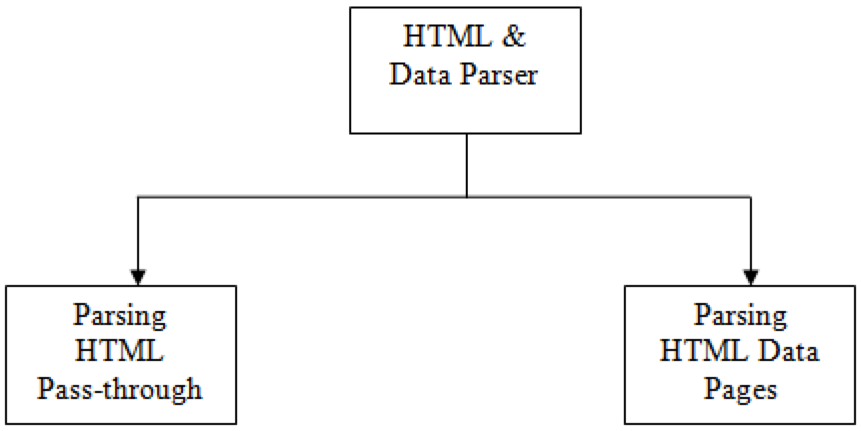
3.1.1.3.4 Cache Argument Interpreter

This Sub Class will handle the output from the Cache Argument and will separate it into its necessary components. It will call and setup what is needed in order for the caching to work.

3.1.1.4 HTML & Data Parser

This section is classified as the Parser Object; its only job is taking the data that's been downloaded and cached by the system so far and figuring out how to parse it for the necessary information using the arguments introduced through the IO class.

FIGURE 3.5 HTML & DATA PARSER ARCHITECTURE



3.1.1.4.1 Parsing HTML Pass-through

This Class will handle the simple move amongst pages as the argument is given, so if the data is two or three pages into a selection, this system will pick out the necessary Uniform Resource Locator (URL)'s necessary to move across multiple pages to grab the data the user is looking for.

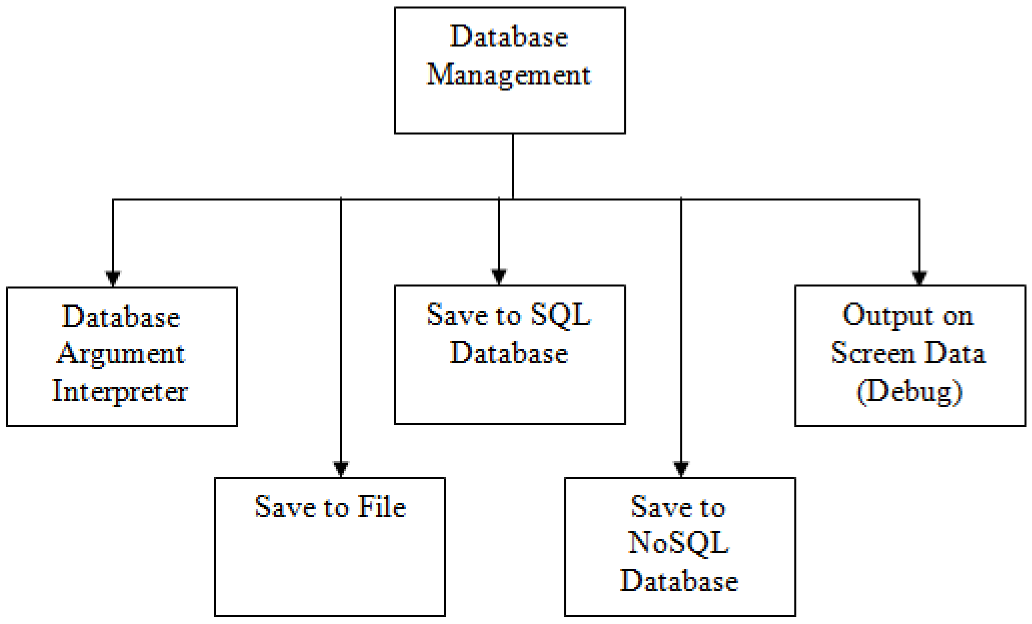
3.1.1.4.2 Parsing HTML Data Pages

This Class will handle the parsing of the actual items and data points throughout the page as well as handle the pagination arguments as they come. Using a For Loop it will do this job recursively going from line to line in a string and pulling what it needs as it needs it.

3.1.1.5 Database Management

This section is classified as the Database Object, its only job is to figure out how the parsed data will output, though it's called the database class it does not mean that the information will always be saved in database form. The user may want it to be saved as a flat file Comma Separated Values (CSV) to that same Cached Environment it downloaded the page to and pulled the data from.

FIGURE 3.6 DATABASE MANAGEMENT ARCHITECTUR



3.1.1.5.1 Database Argument Interpreter

This Class will handle the second most important job in the crawler, it will essentially take the interpreted arguments given in the database selection and go through the process and options the user selected to go through once the site itself has been crawled.

3.1.1.5.2 Save to File

This Class will handle the necessary functions with saving the data to file, in this case a CSV file separated by semi colons with the actual data inside parentheses.

3.1.1.5.3 Save to SQL Database

This Class will handle the necessary Structured Query Language (SQL) functions to save the data to a SQL database, once a connection to the database has been established by the Database Argument Interpreter.

3.1.1.5.4 Save to NoSQL Database

This Class will handle the necessary functions involved with setting up or creating a Table in a NoSQL database. It will then do all the necessary functions to transition the data over to those tables as necessary.

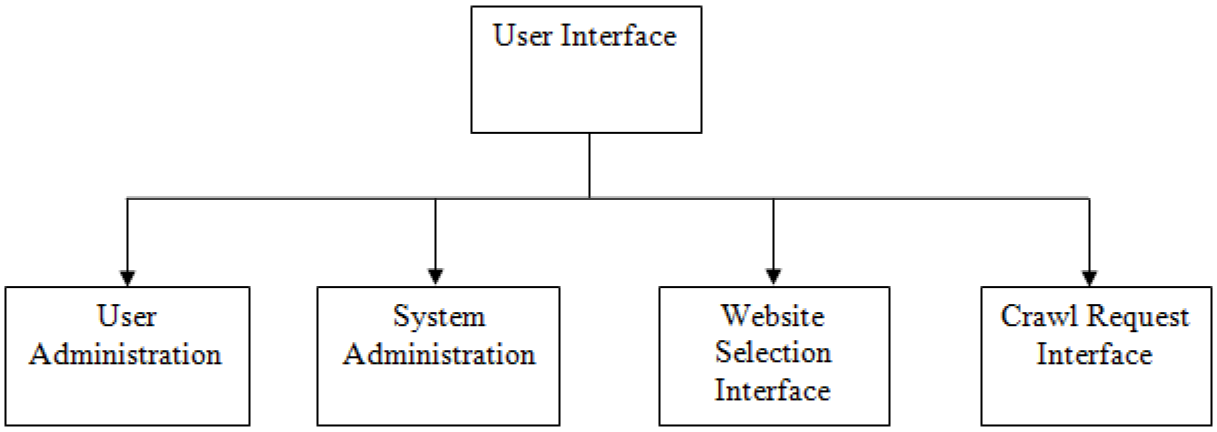
3.1.1.5.5 Output on Screen Data (Debug)

This Class will handle the Debug on screen data dumb required to see if your arguments worked without necessarily needing to save the data to a file or database. This allows the developer building both the crawler core as well as the User Interface to see what the output for a set of commands are going to be before they implement them.

3.1.1.6 User Interface

This section is classified as the User Interface Object, it's one of the more complicated systems as it has to first understand what the user is doing and interpret that information into a system that the IO class can understand.

FIGURE 3.7 USER INTERFACE ARCHITECTURE



3.1.1.6.1 User Administration

This Class will handle all of the aspects involved with what a user sees and how they interact with the system.

3.1.1.6.2 System Administration

This Class will handle all of the aspects to how the System Administrator sees and interacts with the system as a whole.

3.1.1.6.3 Website Selection Interface

This Class will handle what graphical interactions the user needs to utilize in order to pull out and select the very specific data they are looking at in a web page, the concept here is that the user sees what appears to be a browser and possible data points are highlighted and given as an option for selection.

3.1.1.6.4 Crawl Request Interface

This Class will handle taking the selections that the user makes in the Website Selection Interface and translate that into actionable command line arguments that the SWC Core needs in order to start the downloading process. It will also take the necessary database or output selection the user makes in order to handle the output arguments.

3.2 Non Functional Computer Software Components

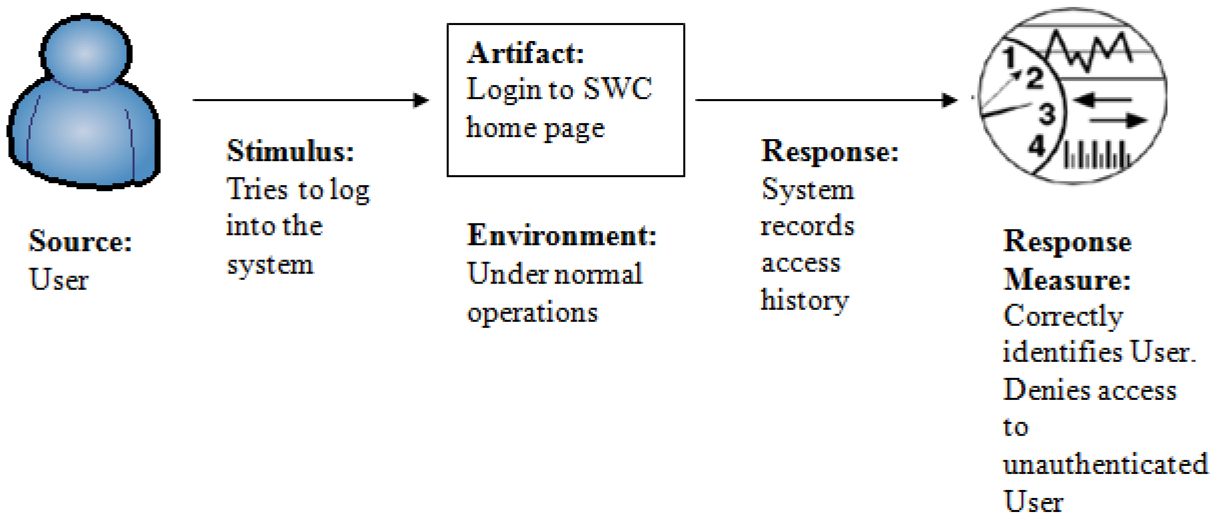
3.2.1 Security

The SWC program shall utilize a User Name Login and password. This shall be associated with the user Email to assure uniqueness and to guard against hackers using information technology (IT) threats.

3.2.1.1 Scenarios

Within the security quality attribute scenario there are two components of concern. The first component is regarding the login procedure and the second component relates to the data storage. When the user tries to log into the system, the system has to identify the user and records the event in its access history. After the successful identification and authentication process, the system shall correctly assign the user to one of two user classes: System Administrator and Users. The users of the System Administrator have a greater privilege that allows total access to every module of the SWC system. The User is only allowed to access and view their own information. The system shall deny the user access to the system, if their User Name is not successfully authenticated.

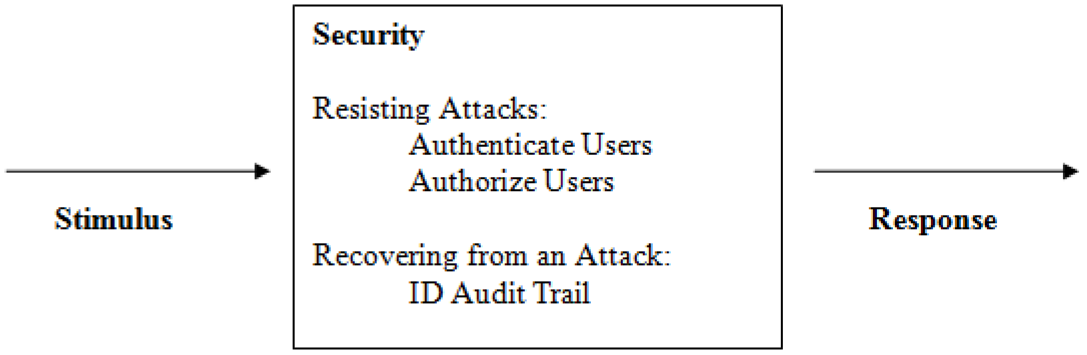
FIGURE 3.8 SECURITY QUALITY ATTRIBUTE SCENARIOS


Source: User
Stimulus: Tries to log into the system
Artifact: Login to SWC home page
Environment: Under normal operations
Response: System records access history
Response Measure: Correctly identifies User. Denies access to unauthenticated User
1
2
3
4

3.2.1.2 Tactics

In the scenarios illustrated in the section above, the appropriate tactics to apply to the Security quality attribute have already been alluded to. Resisting attacks by utilizing the proper security technology to authenticate and authorize users upon login is one of the primary tactics that should be considered. The other primary tactic should focus on the recovery of the system after a security breach is detected. The enforcement of constant identification and logging of activities by all the users to create a complete audit trail is the only way to enable this sort of tactic.

FIGURE 3.9 SECURITY QUALITY TACTICS



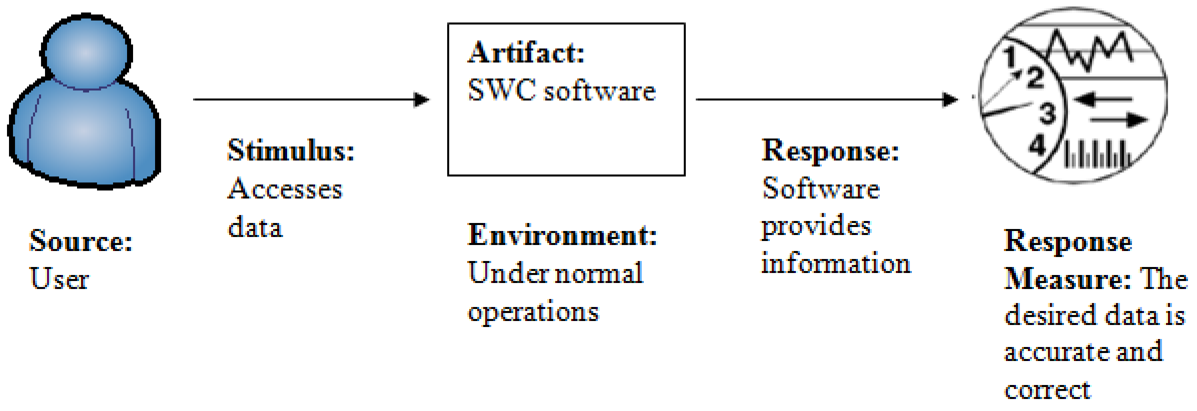
3.2.2 Correctness

In order to provide the desired level of value to the customer, the SWC software must operate correctly. The correctness of the software shall be evaluated by determining the defects per thousand lines of code (KLOC) reported by the user following the program release for general use. The target of correctness for the SWC program is < 1% and to ensure this goal is met, the product shall meet a correctness of < 0.1% during testing.

3.2.2.1 Scenarios

This quality attribute is typically concerned with the extent to which the SWC software conforms to its requirements. As the software shall be required to provide accurate and correct information for the users to operate the features effectively, correctness can be considered as one of the most important attributes. The users of the software are the source of the data and they provide a stimulus to the system by attempting to access the data. Under normal conditions the stimulus is provided and the response is expected to happen by providing the information that is being requested. The information shall then be evaluated for correctness by determining its accuracy.

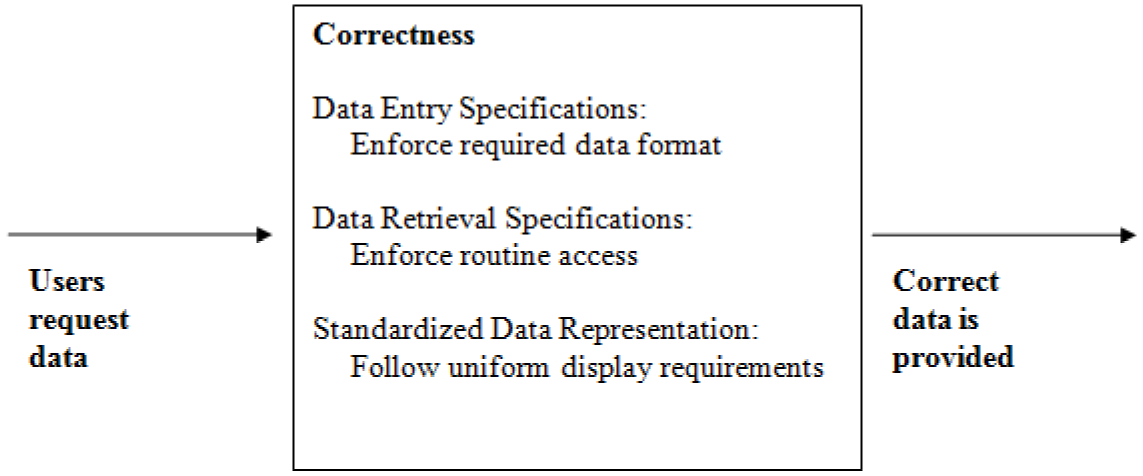
FIGURE 3.10 CORRECTNESS QUALITY ATTRIBUTE SCENARIOS


Source: User
Stimulus: Accesses data
Artifact: SWC software
Environment: Under normal operations
Response: Software provides information
Response Measure: The desired data is accurate and correct

3.2.2.2 Tactics

Data entry specifications, which consider the enforcement of required data formats are being followed upon data entry, are an appropriate tactic to apply to the Correctness quality attribute. The enforcement of data retrieval specification upon obtaining data from the database is another factor that should be considered. The uniform representation of data when it is displayed throughout the system and for the user is the final tactic that must be considered.

FIGURE 3.11 CORRECTNESS QUALITY TACTICS


Users request data
Correctness
Data Entry Specifications:
Enforce required data format
Data Retrieval Specifications:
Enforce routine access
Standardized Data Representation:
Follow uniform display requirements
Correct data is provided

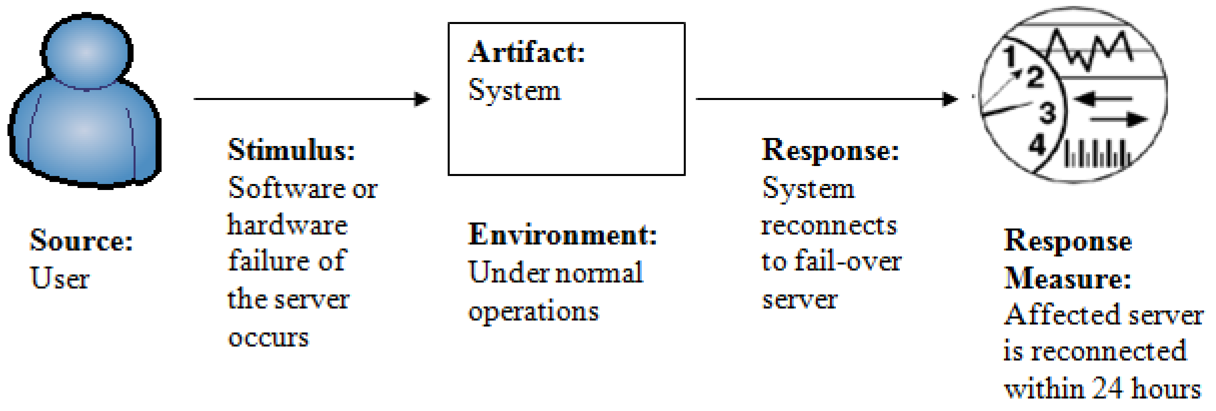
3.2.3 Maintainability

To allow for maximum code re-use and easy maintenance, the code base for the SWC program is modular in design. Common operations and access to data are encapsulated in classes that de-couple the interface from implementation. The user GUIs are also modular, which will allow for simple upgrades and custom re-branding as the clients may require.

3.2.3.1 Scenarios

When the software or hardware failure of a server occurs, such as the web server shuts down, the system should reconnect to the fail-over redundancy server to continue to provide services for all the users. Services from the Internet Service Providers (ISP) should be re-established and fixed on the affected server within 24 hours.

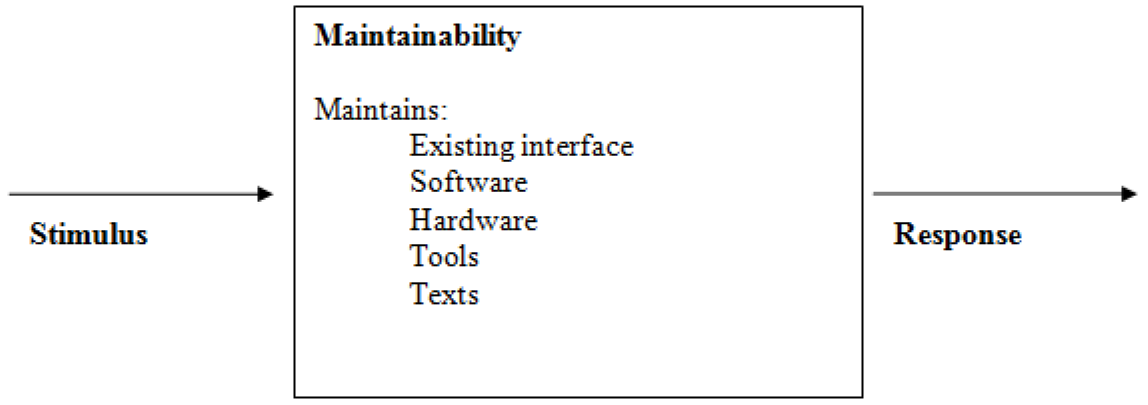
FIGURE 3.12 MAINTAINABILITY QUALITY ATTRIBUTE SCENARIOS


Source:
User
Stimulus:
Software or hardware failure of the server occurs
Artifact:
System
Environment:
Under normal operations
Response:
System reconnects to fail-over server
Response Measure:
Affected server is reconnected within 24 hours

3.2.3.2 Tactics

Maintaining the SWC software, hardware, tools, and the texts describing the aforementioned attributes are the appropriate tactics to apply to the Maintainability quality attribute. Maintaining the SWC software shall involve configuration management and change management. Maintaining the hardware shall focus on keeping the hardware functional and the operational maintenance agreements. The quality attribute for tools can either involve keeping the Integrated Development Environment for each build of the software, for the tools used for configuration management, or record keeping. The quality attribute for texts shall involve keeping the records of maintenance and the documentation that was involved with building the current application.

FIGURE 3.13 MAINTAINABILITY QUALITY TACTICS



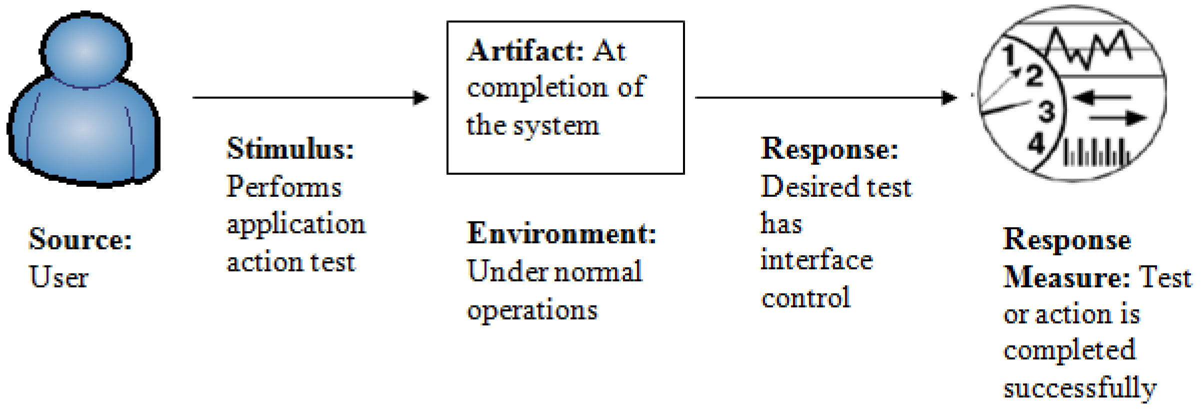
3.2.4 Testability

To determine a consistent and complete software product and to demonstrate the software fulfillment of each of the requirements, an in-depth testing program will be utilized. The user friendliness of the SWC program will be a key factor in testing. Extensive testing will be performed to demonstrate the usability of the SWC program.

3.2.4.1 Scenarios

This quality attribute is typically concerned with the relative effort in which the SWC software can be made to demonstrate its faults and/or provide the necessary interface controls to utilize the system. This scenario has the developers or testers are the sources that shall provide the stimulus of performing the application actions/tests on the system. Under the normal conditions and after the stimulus is provided, the response can be characterized as the system providing the necessary interface control to carry out the action or test. The method of measuring the response involves determining if the desired action or test is successfully carried out or completed.

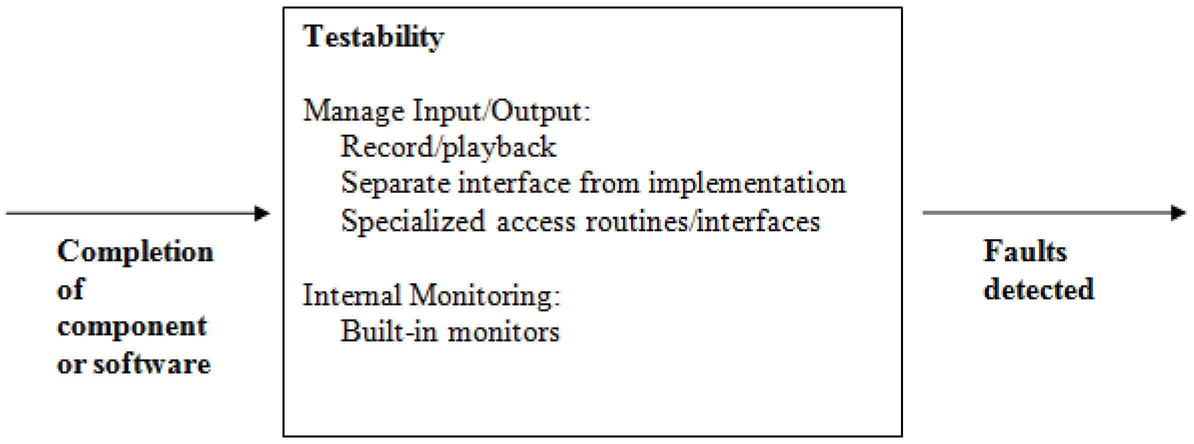
FIGURE 3.14 TESTABILITY QUALITY ATTRIBUTE SCENARIOS


Source: User
Stimulus: Performs application action test
Artifact: At completion of the system
Environment: Under normal operations
Response: Desired test has interface control
Response Measure: Test or action is completed successfully
1
2
3
4

3.2.4.2 Tactics

Managing Input/Output and *Internal Monitoring* are two major factors that must be considered In order to ensure the achievement of the Testability quality attribute within the system. These two factors are further detailed with their respective activities of record/playback, separate interfaces provided to the developers and testers, and specialized access routines that allow the bypassing of the application to effectively test it. Faults can be more efficiently detected by utilizing built-in monitors.

FIGURE 3.15 TESTABILITY QUALITY TACTICS


Completion of component or software
Testability
Manage Input/Output:
Record/playback
Separate interface from implementation
Specialized access routines/interfaces
Internal Monitoring:
Built-in monitors
Faults detected

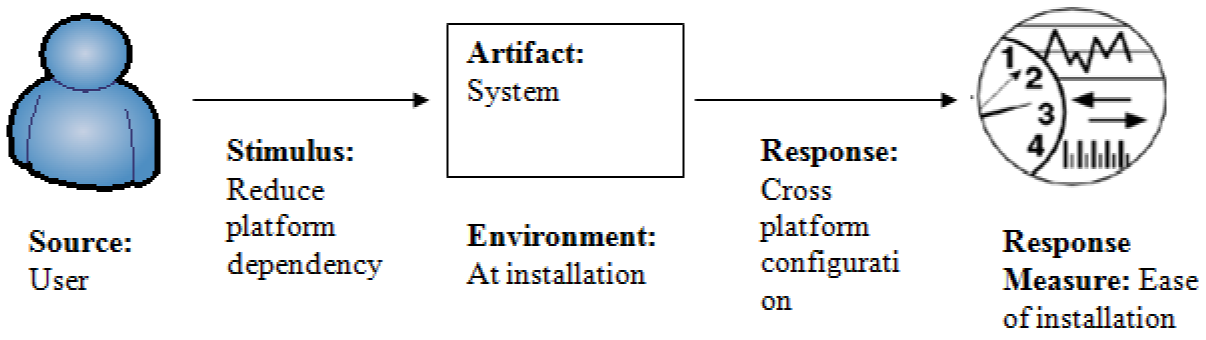
3.2.5 Portability

The SWC program is being developed within an object-oriented architecture that encapsulates the data and operations. The module approach will maximize the reuse of the software in different environments. The use of abstraction between the application logic and the system interfaces shall determine the portability. While the SWC program is being built only for the Linux Operating System and not multiple Operating Systems (OS)s, one of the main objectives during the enhanced phase will be to have the software operate in other Operating Systems.

3.2.5.1 Scenarios

This quality attribute is typically concerned with the relative effort to adapt the SWC software to different environments. This scenario has the platform dependency of the system tested by the System Administrator. In this case, the System Administrator is viewed from the implementation perspective. The desired response is to have the system be easily adaptable to cross platform configuration with the response measure being the ease of installation. The system involves the software and hardware that is to be used.

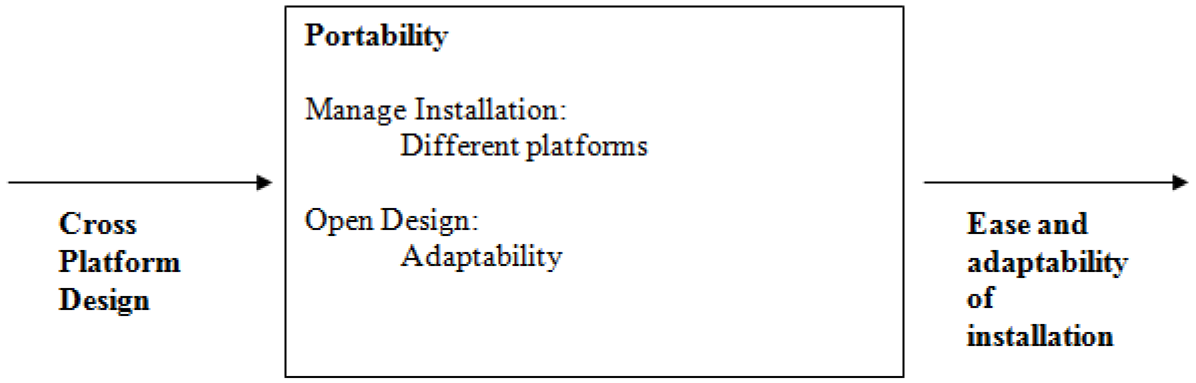
FIGURE 3.16 PORTABILITY QUALITY ATTRIBUTE SCENARIOS


Source: User
Stimulus: Reduce platform dependency
Artifact: System
Environment: At installation
Response: Cross platform configurati on
1
2
3
4
Response Measure: Ease of installation

3.2.5.2 Tactics

The SWC software shall be thought with cross platform design in mind in order for portability to be assured during design time. Another tactic is to have an open design that is adaptable to different environments. When distributing an installation, the SWC software shall run on different platforms. The goal is to have ease and adaptability of installation.

FIGURE 3.17 PORTABILITY QUALITY TACTICS



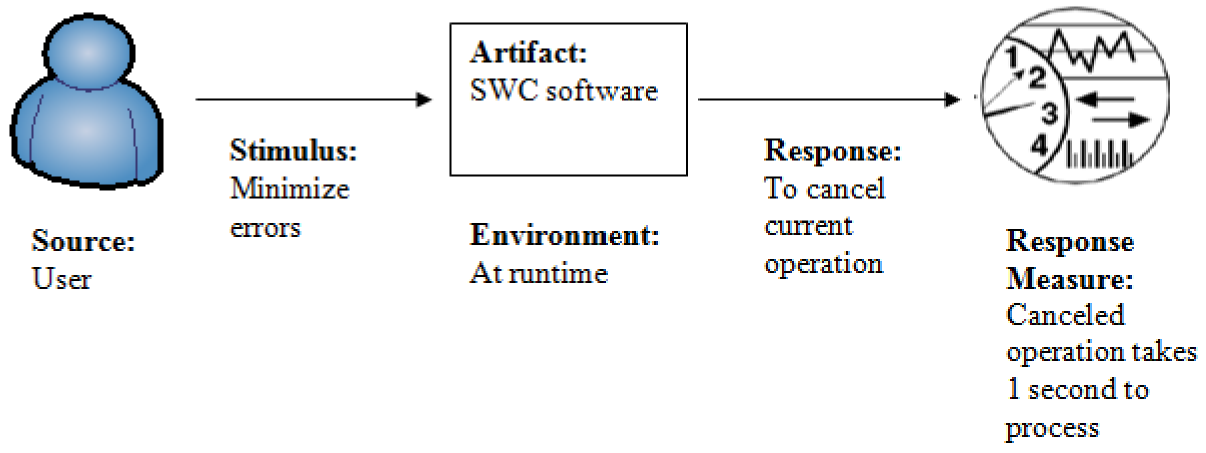
3.2.6 Usability

The primary factor for maximizing usability is to employ the iterative design, which progressively refines the design through evaluation from the early stages of design. The successive evaluation steps shall enable the software designers and software developers to incorporate user and client feedback until the system reaches an acceptable level of usability.

3.2.6.1 Scenarios

This quality attribute is typically concerned with the relative ease of using the SWC software, learning the software, and building confidence with the software. The end-users of the system are the source of this scenario; they provide the stimulus of minimizing errors to the system under normal conditions. In this case, the desired response involves the need of the end-user to cancel the current operation. The measure of this response includes the criteria that the canceled request of the operation occurs within 1 second.

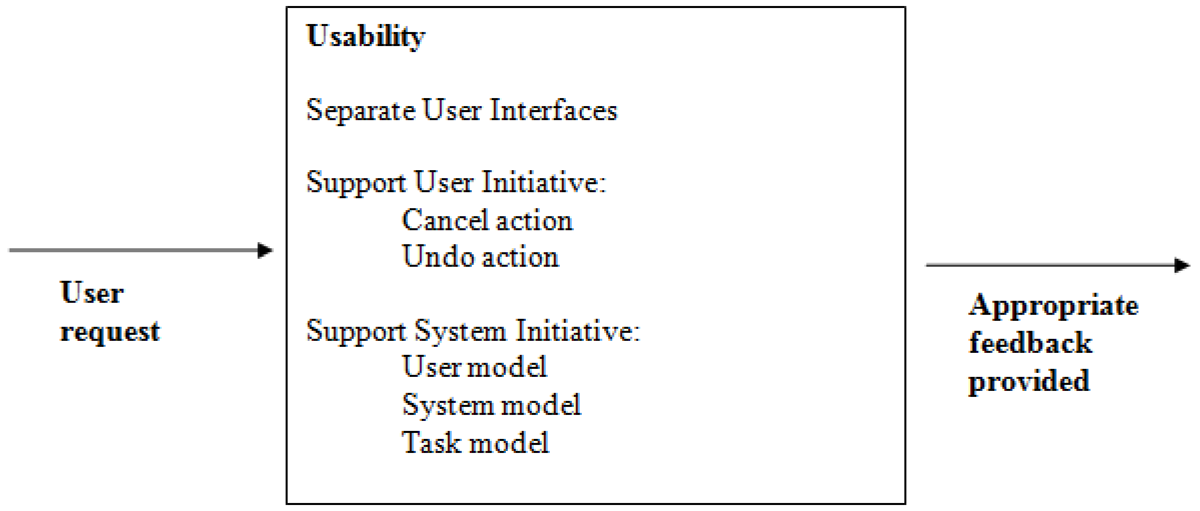
FIGURE 3.18 USABILITY QUALITY ATTRIBUTE SCENARIOS



3.2.6.2 Tactics

A separate user-interface for the user to utilize, supporting the user initiative, and supporting the system initiative are three major factors that must be considered in order to ensure the usability quality attribute. In supporting the user initiative, the system shall provide the user with the ability to cancel and undo actions at all times. In supporting the system initiative, the system must provide the user, system, and task models.

FIGURE 3.19 USABILITY QUALITY TACTICS



3.2.7 Modifiability

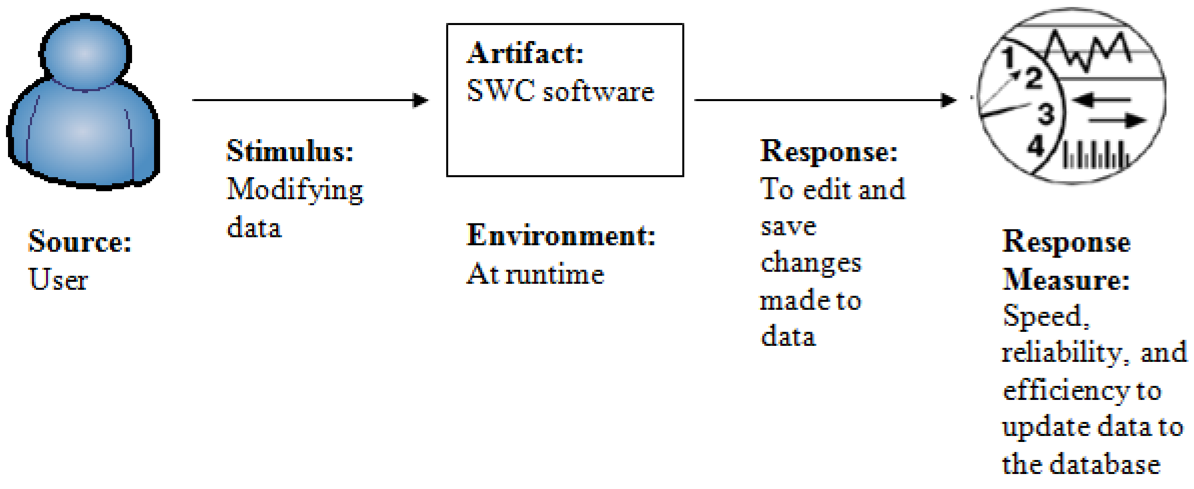
Modification of the SWC project shall correct faults and improve performance along multiple platforms. The modified software shall align with the customer priorities and staffing, allowing the modifications. The areas of importance are as follows:

- Adaptive – dealing with changes and adapting in the software environment
- Perfective – accommodating with new or changed user requirements which concern functional enhancements to the software
- Corrective – dealing with errors found and solutions

3.2.7.1 Scenarios

This quality attribute is the typical effort in which the SWC software can adapt changes to its software. In this scenario, the user needs to change the data that they interact with in the software. The operation shall occur at runtime. The desire response is to edit and save changes made to the data in a speedy, reliable, and efficient way so that the user can extract that data in the future.

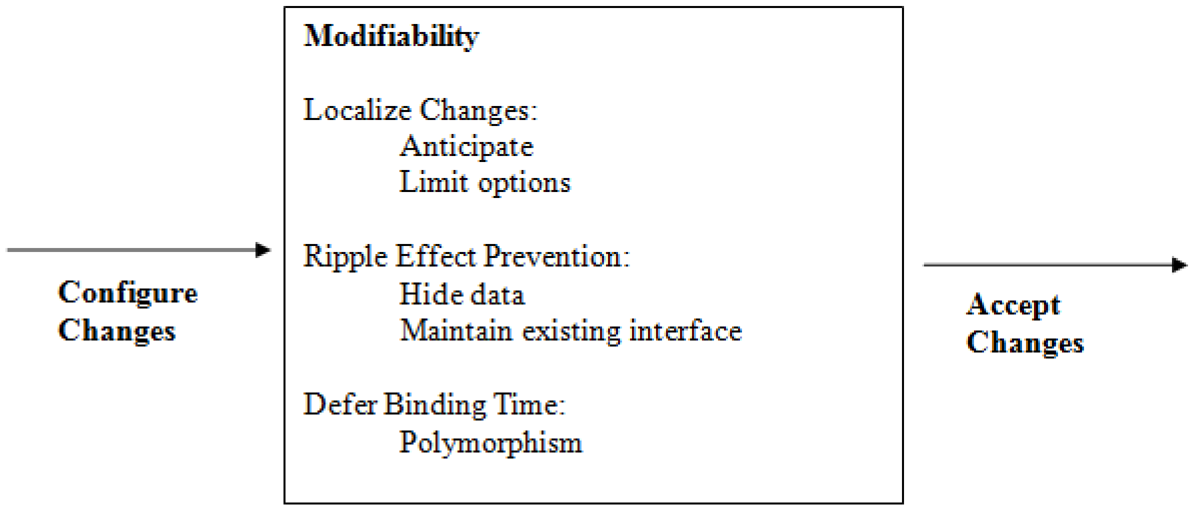
FIGURE 3.20 MODIFIABILITY QUALITY ATTRIBUTE SCENARIOS


Source: User
Stimulus: Modifying data
Artifact: SWC software
Environment: At runtime
Response: To edit and save changes made to data
Response Measure: Speed, reliability, and efficiency to update data to the database
1
2
3
4

3.2.7.2 Tactics

The SWC software should be able to localize and anticipate changes in order for the software to assure modifiability. Managing modifications will ensure that negative ripple effects do not occur. To make changes more easily taken by the software, polymorphism can be used. The ultimate goal of the SWC software is to have the system accept the changes requested with minimal disruptions.

FIGURE 3.21 MODIFIABILITY QUALITY TACTICS



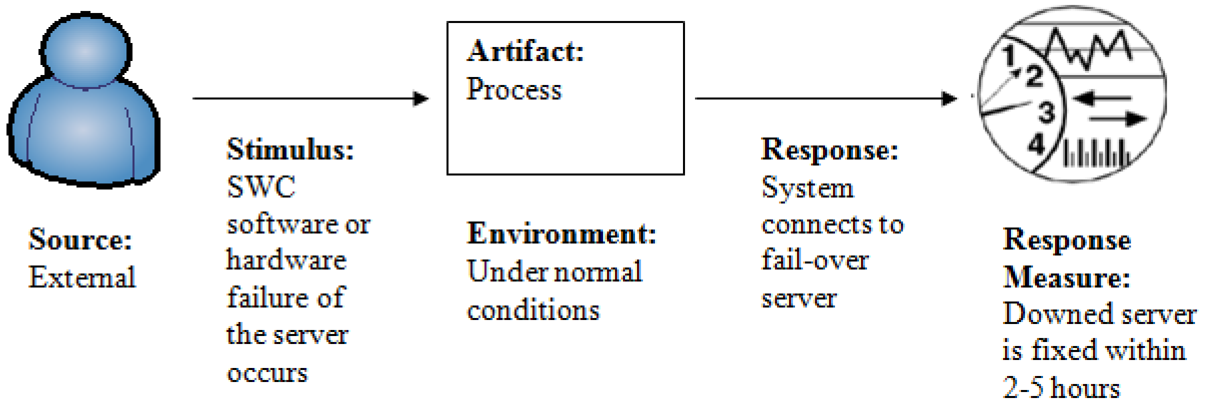
3.2.8 Availability

Availability is the proportion of time a system is in a functioning condition. MTBF (Mean time between failure) and MTTR (Mean time to repair) values are estimated for each component of the software system. This shall be based upon a system availability of 99.9% and a SWC program availability of 99.9%.

3.2.8.1 Scenarios

This quality attribute is typically concerned with the amount of time that the system is up and running correctly. This is the time between failure and the time needed to resume operation after failure. In this scenario, the SWC software or hardware failure occurs during normal operations. The system connects to a fail-over server that shall take over the responsibility of the downed server; to ensure that operations continue to be available with no downtime. The downed server is then restored by first fixing the SWC software or hardware problem within a 2 to 5 hour time frame.

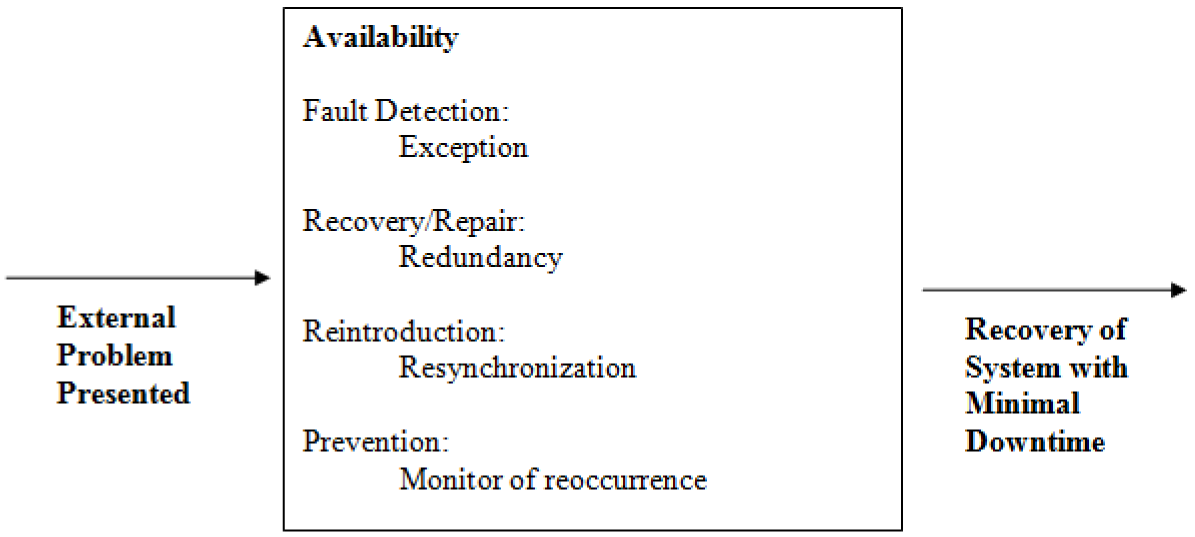
FIGURE 3.22 AVAILABILITY QUALITY ATTRIBUTE SCENARIOS


Source: External
Stimulus: SWC software or hardware failure of the server occurs
Artifact: Process
Environment: Under normal conditions
Response: System connects to fail-over server
Response Measure: Downed server is fixed within 2-5 hours
1
2
3
4

3.2.8.2 Tactics

With the introduction of an external problem, the system shall provide a fault detection, the ability to recover and accommodate for fast repairs, the easy resynchronization after rollover, and the monitoring of resources to prevent the occurrence of the same problems. When a problem is introduced, the main goal is the fast recovery of the system with minimal downtime.

FIGURE 3.23 AVAILABILITY QUALITY TACTICS



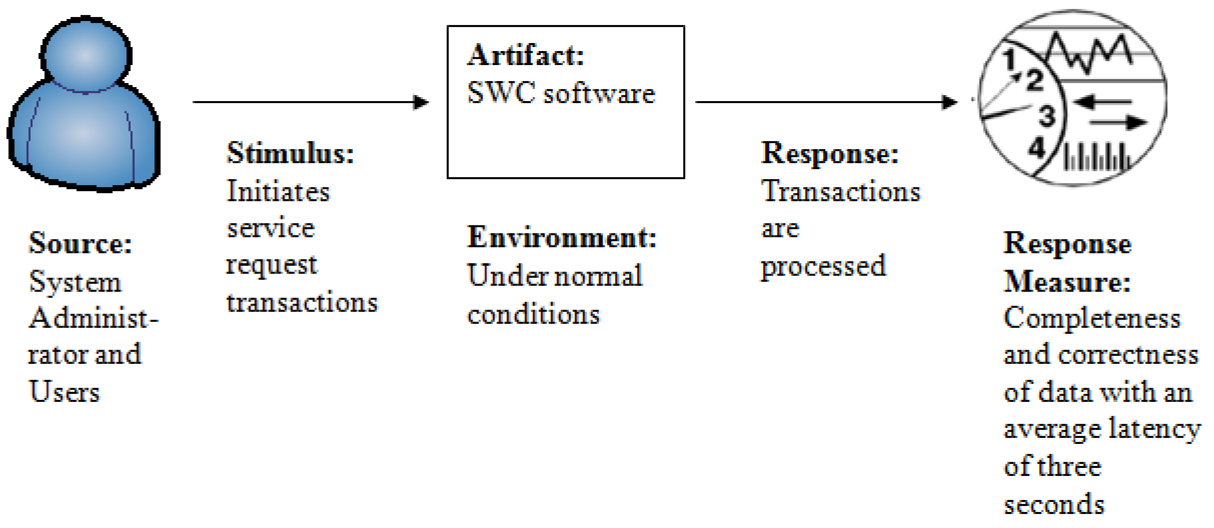
3.2.9 Performance

Performance modeling shall be performed utilizing the use case information as the input. For the SWC software system, a performance monitoring plan shall be developed. Performance software engineering applies a subset of activities related to performance monitoring, both for the performance test environment as well as for the production environment.

3.2.9.1 Scenarios

This quality attribute is typically concerned with the response time, utilization, and throughput behavior of the SWC software. In this scenario, the System Administrator and users provide sources of stimulus to the software. When the users request services from the system, sporadic events can occur. Under normal operations, the transactions shall be processed in the system and an acceptable response shall be provided to the users. Every process shall be completed with an average latency of three seconds and the data contained in the response shall be complete and accurate.

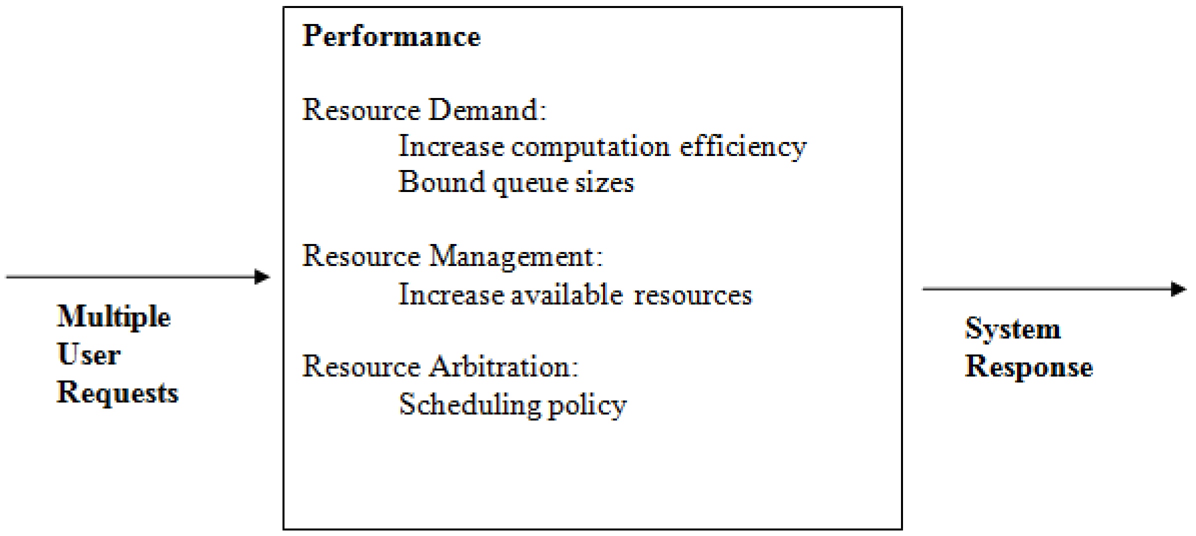
FIGURE 3.24 PERFORMANCE QUALITY ATTRIBUTE SCENARIOS


Source:
System Administ-rator and Users
Stimulus:
Initiates service request transactions
Artifact:
SWC software
Environment:
Under normal conditions
Response:
Transactions are processed
Response Measure:
Completeness and correctness of data with an average latency of three seconds
1
2
3
4

3.2.9.2 Tactics

Resource demand, which is primarily affected by the SWC software's computational efficiency and the size of the data queues, is the appropriate tactic to apply to the Performance quality attribute. Another consideration is Resource Management, to ensure that the necessary available resources are increased or fully utilized. Another tactic that can used to control the scheduling policies on the software to limit user loads at critical times is resource arbitration.

FIGURE 3.25 PERFORMANCE QUALITY TACTICS



3.3 General Constraints

This section lists the constraints that have an influence on the development of the SWC product and discusses the effects resulting from failure to comply with the constraints.

3.3.1 Architecture

The SWC program shall consist of a three-layered architecture, meaning that each layer accesses the layer directly below it. The three layers are the presentation tier, the middle tier, and the data tier.

3.3.1.1 Presentation Tier or GUI

- Provides the user with the GUIs.

3.3.1.2 Middle Tier

- Access the SWC program modules.
- Interfaces to the Commercial Off-The-Shelf (COTS) database.

3.3.1.3 Database Tier

- Contains the data access to the users.
- Provides the functions from the COTS database.

3.3.2 Time

The largest constraint for the development of the SWC product is time. The SWC Development Team's goal is to provide a functional prototype within two weeks of the final deliverance to allow for any alterations and improvements that are derived from testing and customer feedback. An aggressive pace will be used for a steadfast and rapid development and will leave little room for deviation from the initial plan. A strict schedule has been implemented and regimented into the development timeline.

3.3.3 Regulatory Policies

The regulatory policies impacting the SWC program development include the following:

- Internet operation policies as they apply to the web services and ISP. When installing web based capabilities, the provision of static Internet Protocols (IP) and aliases can be problematic.
- The limitations of the telephone company on exchanging data through the dial-up network.
- The protection of user and owner privacy.

3.3.4 Computer Requirements

The requirements of the host computer for the SWC program are determined by the application modules to be implemented. Applications over the Internet have a substantial impact on the processing and memory functions. A reliable connection is required when using the application and an unreliable connection can cause an unpredictable operation of the system, which can include the loss or corruption of data. The following subsections summarize the requirements of the host computer to maximize the SWC program performance.

3.3.4.1 Hardware Requirements

The hardware requirements of the host computer must provide the basic capabilities to support the installed OS, the external interfaces that are consistent with the OS, and the SWC software. The minimum hardware requirements are:

- 1 Gigahertz (GHz) processor, Pentium or equivalent.
- Linux Operating System.
- 2 Gigabytes (GB) of Random Access Memory (RAM).
- 5 GB of hard-disk space for storage.
- Monitor.
- Keyboard.
- Pointing device, mouse or equivalent.

3.3.4.2 Internet Requirements

A full-time Internet connection is required to access the SWC program. The Internet connection can be either a Digital Subscriber Line (DSL) or Cable service. The connection provides a gateway between the home (host computer) and the Internet. To accomplish this, any one of the following connection configurations is required:

- Static IP Address - a dedicated IP address that is assigned to the Internet account by the ISP.

- Dynamic Domain Name Service (DNS) - a service that allows the user to alias the assigned Dynamic IP address to a permanent hostname, to allow the computer to be more easily accessed from various locations on the Internet. This allows the user to access the SWC program via the web server over the Internet by entering the static host name instead of the currently assigned Dynamic IP address.

3.4 Assumptions and Dependencies

This section lists the assumptions and dependencies for the SWC Product.

3.4.1 Assumptions

The SWC Product's requirements outlined in this document are subject to the following assumptions:

- It will be assumed that the user will have an Internet connected web based browser to interface with the software program to initiate the semantic web crawl.
- It will be assumed that the user will have a limited amount of experience with the techniques used to implement the software functions. It is apparent that some users have more experience than others, but a minimal amount of experience is required to operate the SWC product.
- It will be assumed that the users have the necessary familiarity to use the database and GUIs.
- It will be assumed that the functions used will be running in the background away from the user.
- It will be assumed that the user will be able to utilize the forms and fields to enter the criteria for the semantic web crawl necessary for the program to initialize the instance.
- It will be assumed that the host computer will be capable of maintaining a reliable Transmission Control Protocol (TCP)/IP connection.
- It will be assumed that the host computer meet the minimum hardware and software requirements. Section 3.3.4 Computer Requirements of this document outlines the hardware and software requirements.

3.4.2 Dependencies

The SWC project is dependent upon the access of web based servers and the source code that is used to implement the required functions for the web crawl. By accessing the HTTP/HTTPS web page, the user's profile information shall be encrypted with SSL.

A third party ISP's network will host the application server. The primary responsibility of this level is to provide the necessary connectivity between the web's application and database. The application server also fulfills the user requests that the web application provides and retrieves and supplies the proper data from the database. Also provided by the ISP's network is the database and utilizes the storage capacity provided to the application. The database will also be continuously available and permit multiple user initiated connections concurrently.

4.0 Detailed Design

This section of the SDD supports the software coding effort by laying out the design of the SWC software design in a detailed manner.

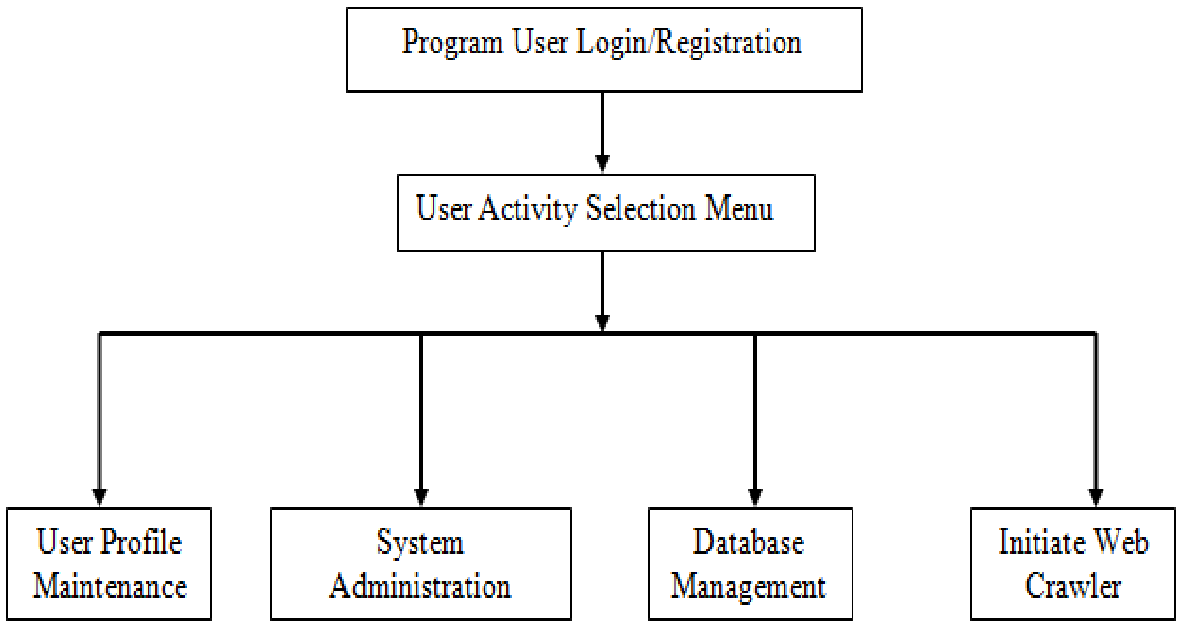
4.1 GUI Screen Display Samples

The SWC program shall be user friendly by providing the user simple to use GUIs. The operations provided to the user by the SWC program are:

- User Registration
- User Login
- User Activity Selection
- System Administration
- Database Management
- Initiate Web Crawler

Figure 4.1 displays the GUI manager based on the function distribution of the SWC program.

FIGURE 4.1 GUI MANAGER FUNCTION DISTRIBUTION



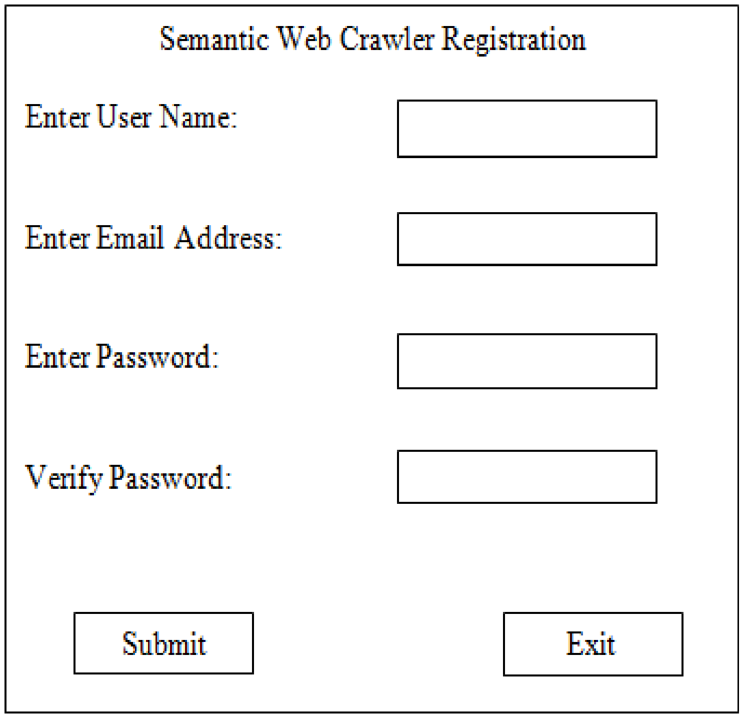
4.1.1 User Registration

The User Registration module shall allow the user to create a user profile for the SWC program. It ensures that the user creates a User Name, a password to provide access, and to provide an E-mail address for updates and information from the System Administrator.

4.1.1.1 User Registration Screens

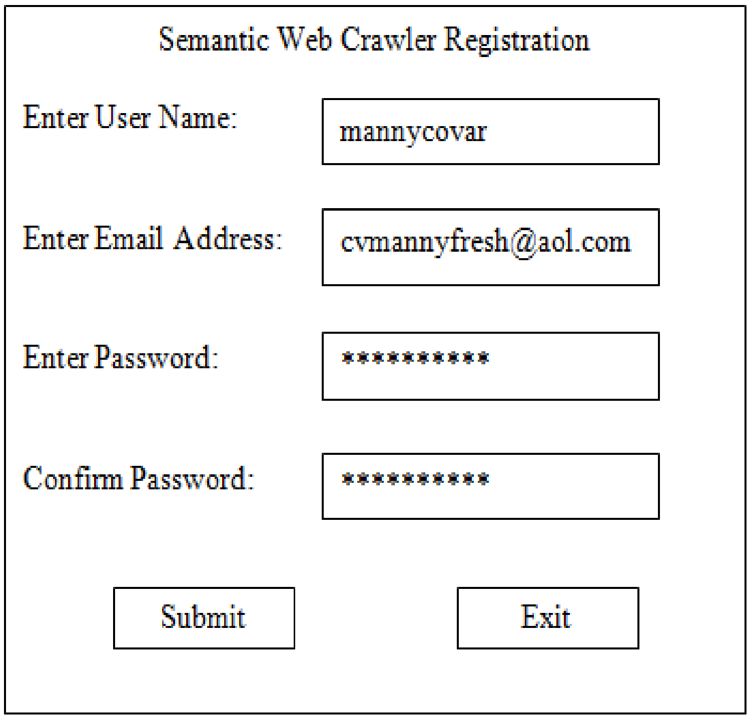
This section contains samples of the displays used for the User Registration. See Figures 4.2 and 4.3.

FIGURE 4.2 BLANK REGISTRATION GUI


Semantic Web Crawler Registration

Enter User Name:	
Enter Email Address:	
Enter Password:	
Verify Password:	

FIGURE 4.3 REGISTRATION GUI WITH USER INFORMATION


Semantic Web Crawler Registration

Enter User Name:	mannycovar
Enter Email Address:	cvmannyfresh@aol.com
Enter Password:	**********
Confirm Password:	**********

4.1.1.2 User Registration Chart Diagrams

This section contains the User Registration Class Diagram and Sequence Chart. See Figures 4.4 and 4.5.

FIGURE 4.4 USER REGISTRATION CLASS DIAGRAM

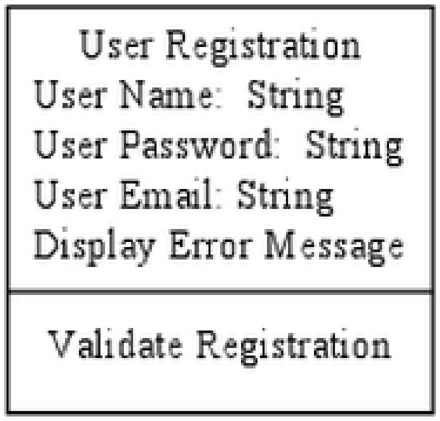
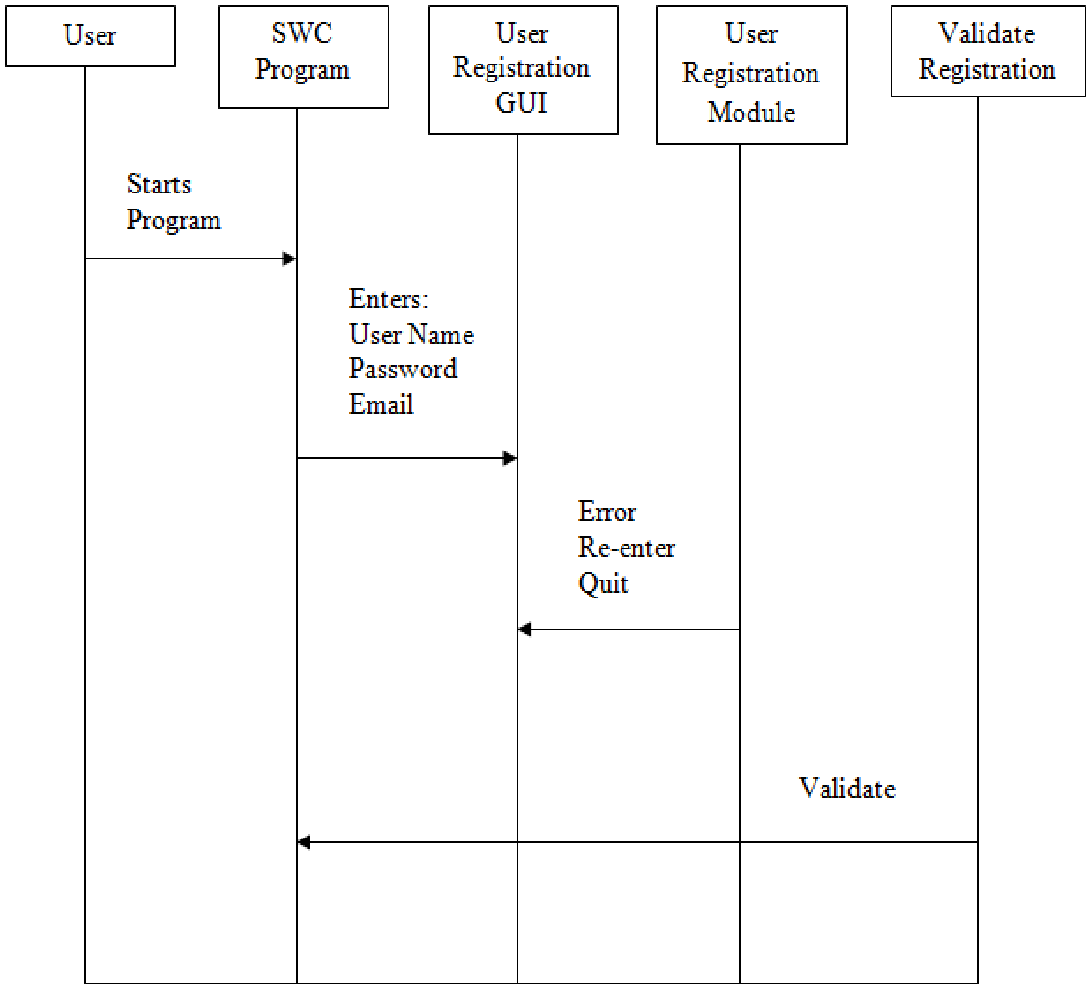


FIGURE 4.5 USER REGISTRATION SEQUENCE CHART



4.1.2 User Login

The User Login module shall provide access to the SWC program for authorized users and security for the system by ensuring only authorized users can access the system and designated sections of the applications as defined by their permissions. Based on the login User Name, the system will determine the permissions of the user by determining whether they are a system administrator or user and displays the User Activity Selection GUI with options for selection of the functionality to be provided to the user.

4.1.2.1 User Login Screens

This section contains samples of the displays used for the User Login. See Figures 4.6 and 4.7

FIGURE 4.6 BLANK LOGIN GUI

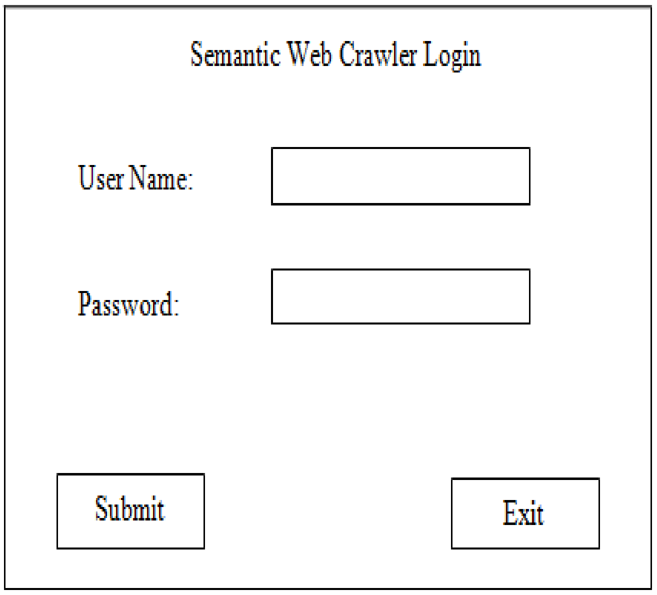
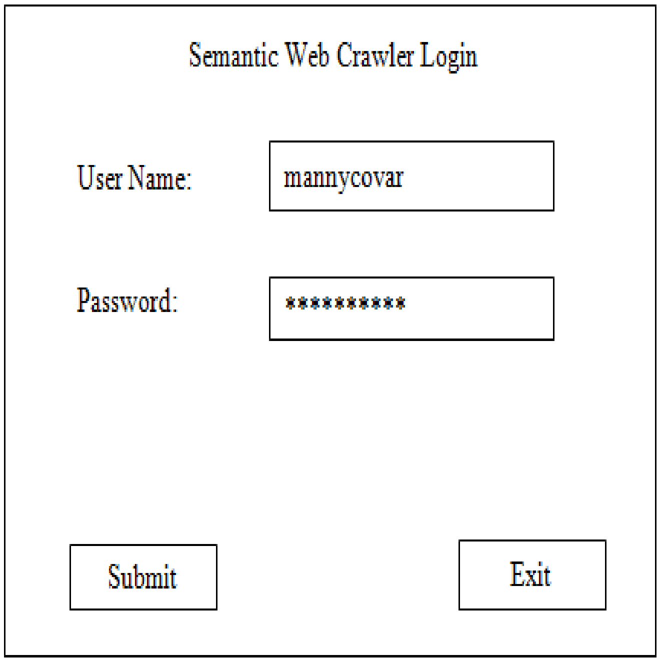


FIGURE 4.7 LOGIN GUI WITH USER INFORMATION



User Name: mannycovar

Password: **********

4.1.2.2 User Login Chart Diagrams

This section contains the User Login Class Diagram and Sequence Chart. See Figures 4.8 and 4.9.

FIGURE 4.8 USER LOGIN CLASS DIAGRAM

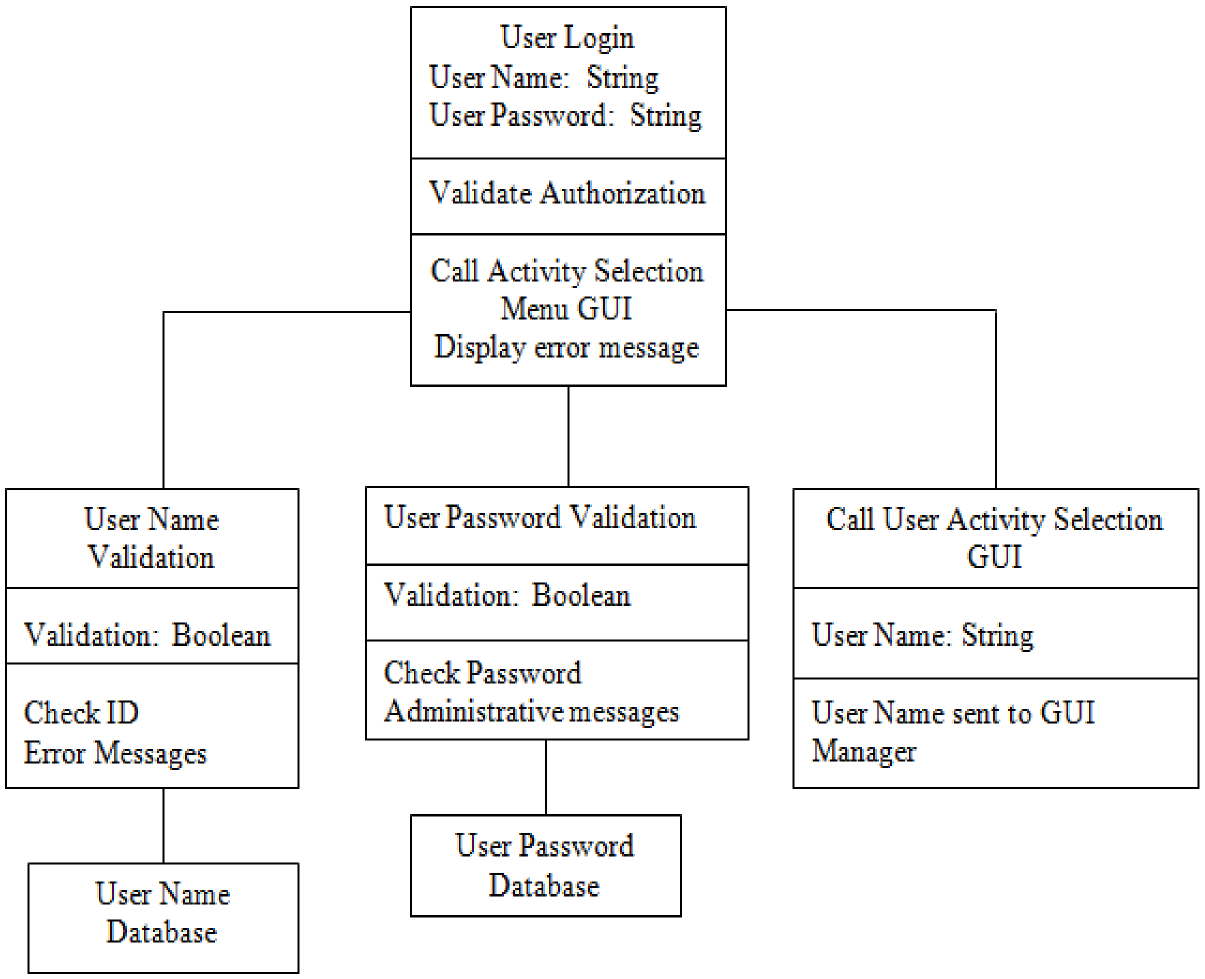
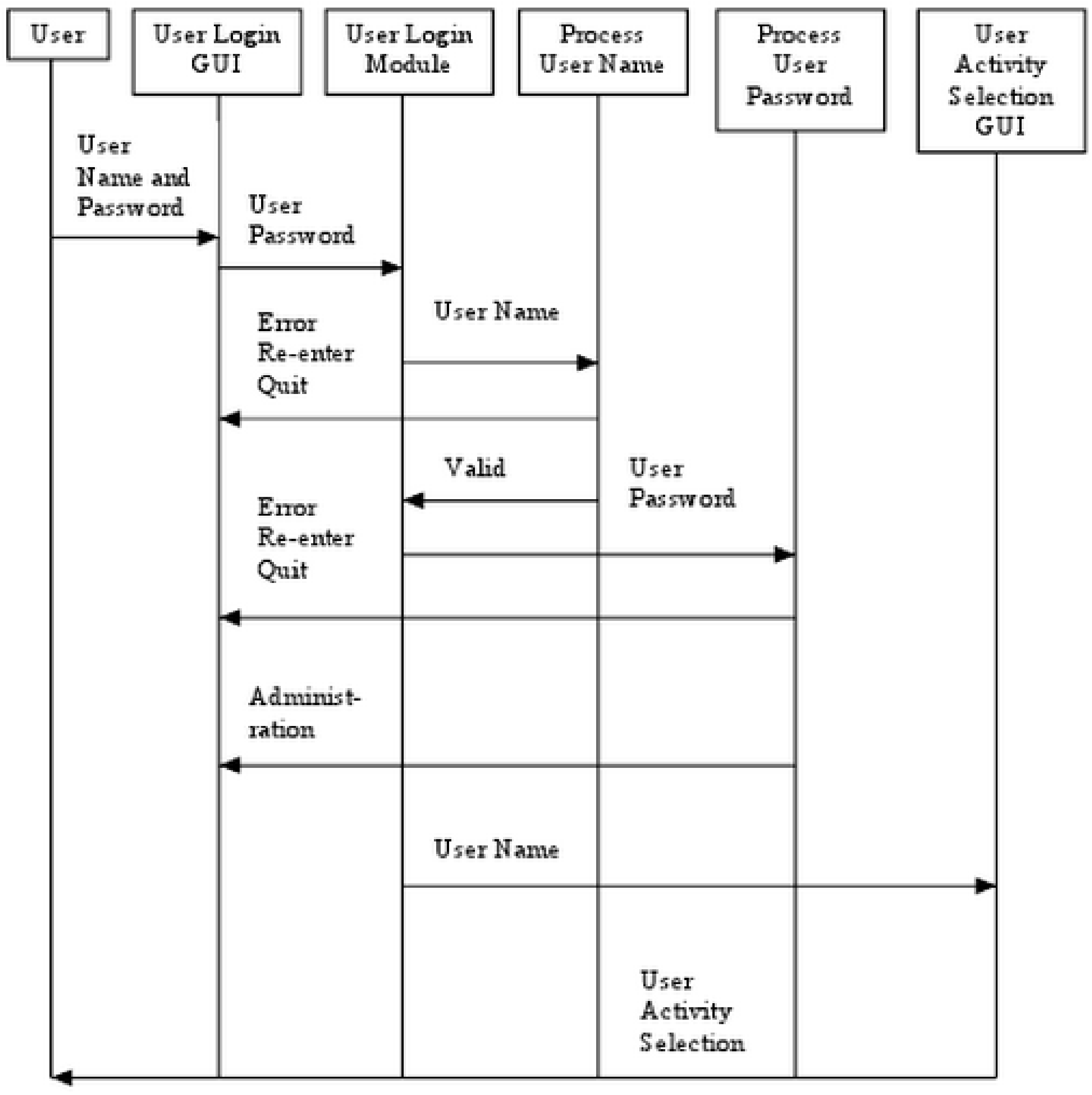


FIGURE 4.9 USER LOGIN SEQUENCE CHART



4.1.3 User Activity Selection

The User Activity Selection GUI shall be accessed by successfully logging into the SWC program.

The GUI manager shall accept the User Name and password from the Login function and returns the User Activity Selection GUI. The User Activity Selection GUI presented to the user will be one of the following classes based on their permissions in the security module:

- System Administrator
- User

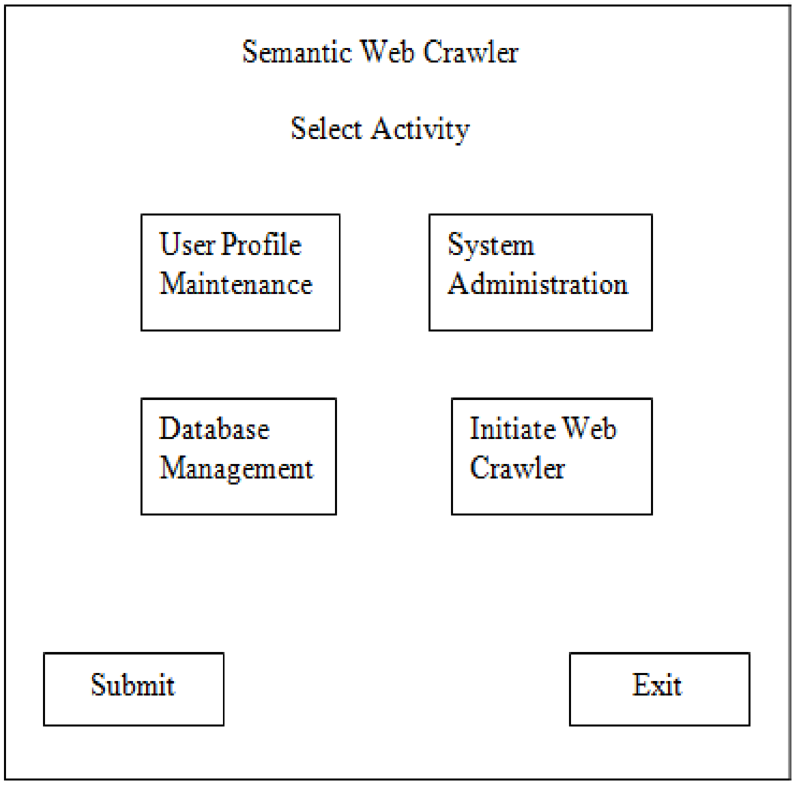
The User Activity Selection GUI shall display a selection of activities. The following activities will be available:

- User Profile Maintenance
- System Administration
- Database Management
- Initiate Web Crawler

4.1.3.1 User Activity Selection Screen

This section contains samples of the displays used for the User Selection Activity. See Figure 4.10.

FIGURE 4.10 USER ACTIVITY SELECTION MENU GUI



4.1.3.2 User Activity Selection Chart Diagrams

This section contains the User Activity Selection Class Diagram and Sequence Chart. See Figures 4.11 and 4.12.

FIGURE 4.11 USER ACTIVITY SELECTION CLASS DIAGRAM

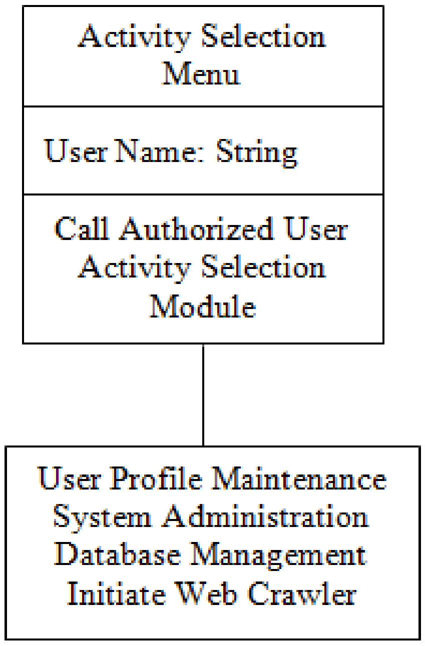
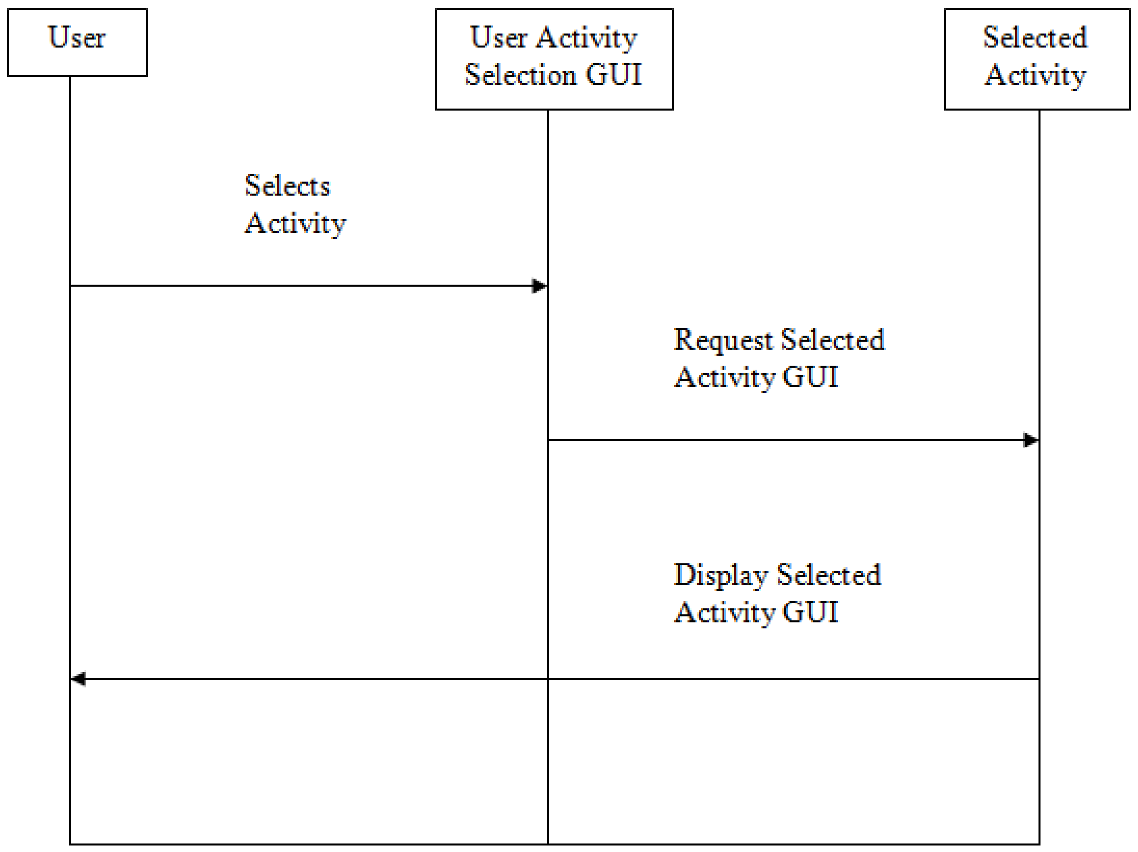


FIGURE 4.12 USER ACTIVITY SELECTION SEQUENCE CHART


User
User Activity Selection GUI
Selected Activity
Selects Activity
Request Selected Activity GUI
Display Selected Activity GUI

4.1.4 User Profile Maintenance

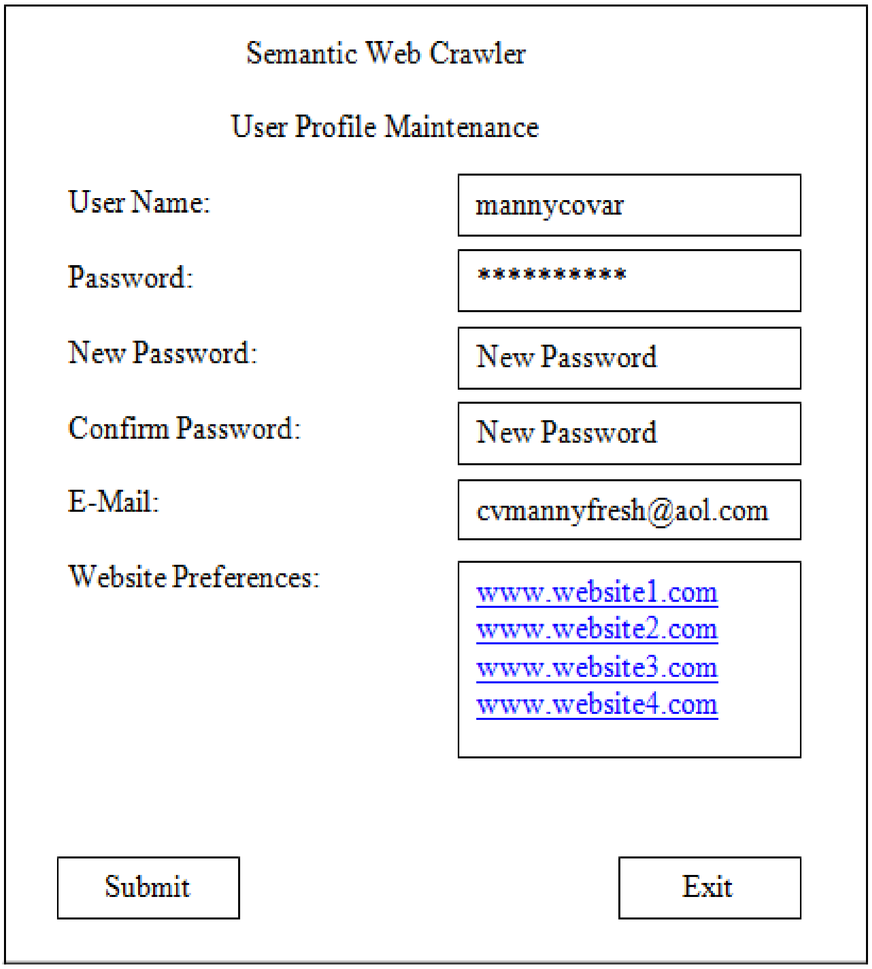
The User Profile Maintenance module is accessed through the SWC User Activity Selection GUI. The User Profile Maintenance module shall allow the user to update the selected data of their personal user profile. The selection of this module shall display the User Profile Maintenance GUI with the data currently held in the data database of the logged in user. The content of the fields is changed by selecting the field with the pointer or tabbing up/down and using the standard keyboard functions to edit. When editing is complete, the *Submit* button is used to forward the changes to the database. All of the following data options may be updated by the user:

- E-mail Address
- Password
- Website Preferences

4.1.4.1 User Profile Maintenance Screens

This section contains samples of the displays used for the System Administration. See Figure 4.13.

FIGURE 4.13 USER PROFILE MAINTENANCE GUI



Semantic Web Crawler

User Profile Maintenance

User Name:	mannycovar
Password:	**********
New Password:	New Password
Confirm Password:	New Password
E-Mail:	cvmannyfresh@aol.com
Website Preferences:	www.website1.com www.website2.com www.website3.com www.website4.com

4.1.4.2 User Profile Maintenance Chart Diagrams

This section contains the System Administration Class Diagram and Sequence Chart. See Figures 4.14 and 4.15.

FIGURE 4.14 USER PROFILE MAINTENANCE CLASS DIAGRAM

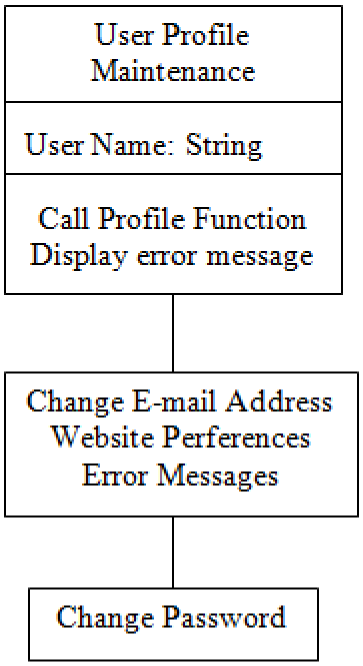
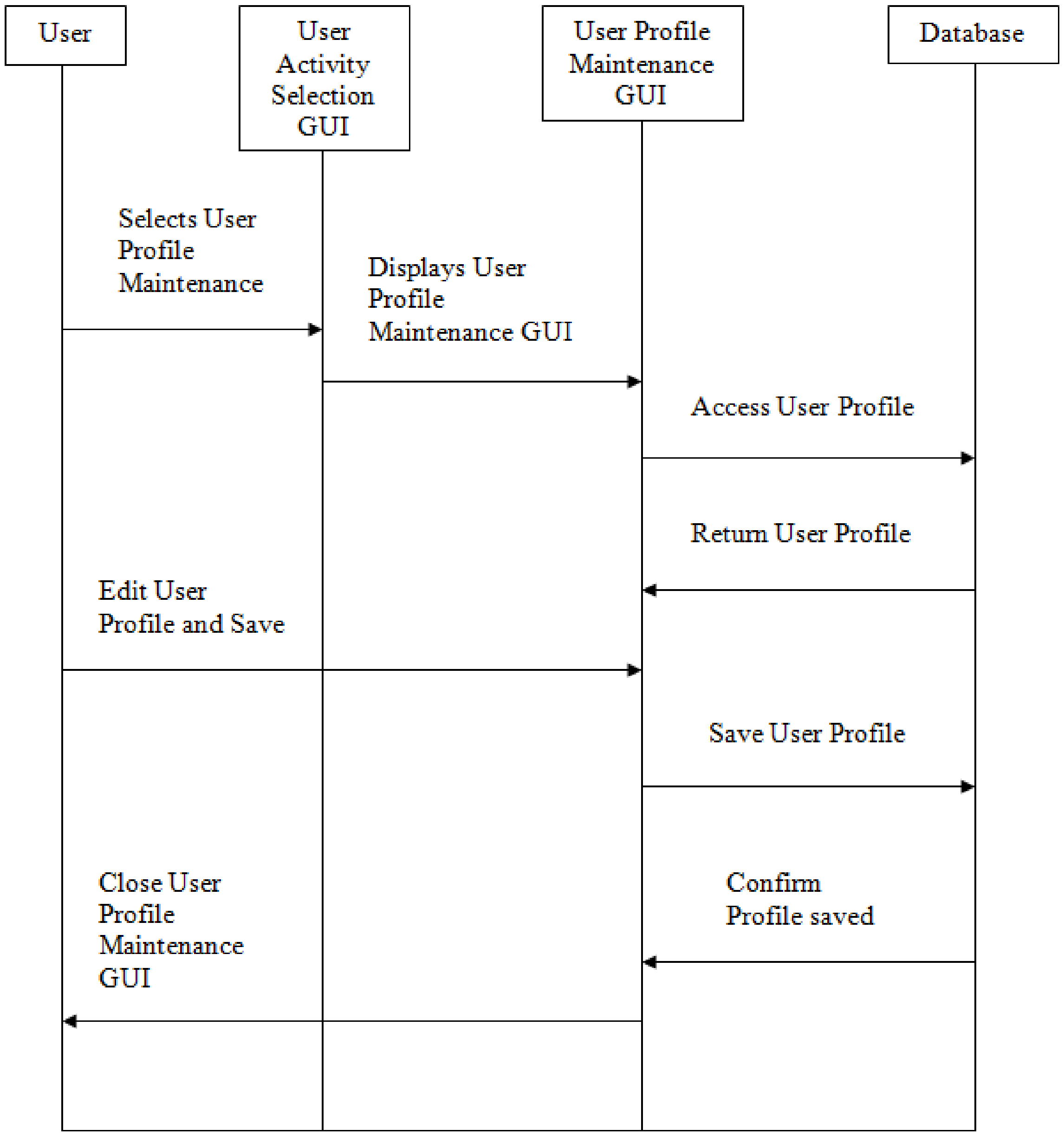


FIGURE 4.15 USER PROFILE MAINTENANCE SEQUENCE CHART


User
User Activity Selection GUI
User Profile Maintenance GUI
Database
Selects User Profile Maintenance
Displays User Profile Maintenance GUI
Access User Profile
Return User Profile
Edit User Profile and Save
Save User Profile
Confirm Profile saved
Close User Profile Maintenance GUI

4.1.5 System Administration

The System Administration module shall provide the user the capability to manage the user profiles by creating new user profiles and editing or deleting existing user profiles. The System Administration GUI shall be accessed by navigating the SWC User Activity Selection GUI. The System Administration GUI shall provide the user a menu with the following options:

- Create New User Profile
- Modify or Delete User Profile

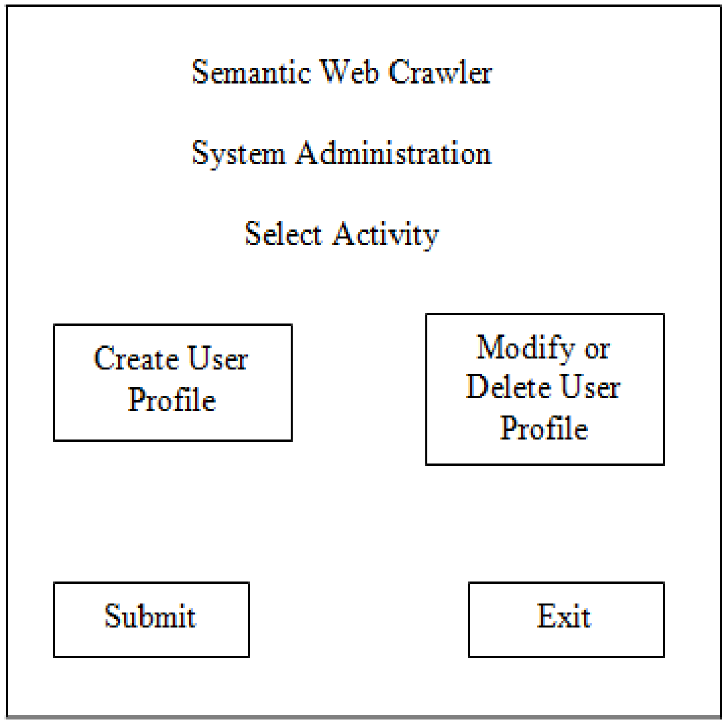
The Create New User Profile GUI shall display a blank user profile GUI. The user fills in the fields with data entry. When the data entry is complete, the user submits the data to the database with the *Submit* button. The module error checks the entered data and either updates the database or returns an error message. The erroneous data is highlighted. The *Exit* button returns the user to the System Administration Activity Selection GUI.

The Modify or Delete User Profile Activity requests the user to enter in the User Name of the profile to be modified or deleted. The selected profile with the data currently in the database is displayed when the User Name is submitted. The user can either modify the data and submit it with the *Submit* button or delete the profile with the *Delete* button. Prior to implementation, the module confirms the selected action. The module responds with a confirmation that the selected activity was completed. The module then queries the user if they would like to modify or delete an additional profile. Selecting the *No* button returns the user to the System Administration Activity Selection GUI. Selecting the *Yes* button allows the user to access another profile.

4.1.5.1 System Administration Screens

This section contains samples of the displays used for the System Administration, Create New User Profile, and Modify or Delete User Profile. See Figures 4.16, 4.17, and 4.18.

FIGURE 4.16 SYSTEM ADMINISTRATION GUI



4.17 CREATE NEW USER PROFILE GUI

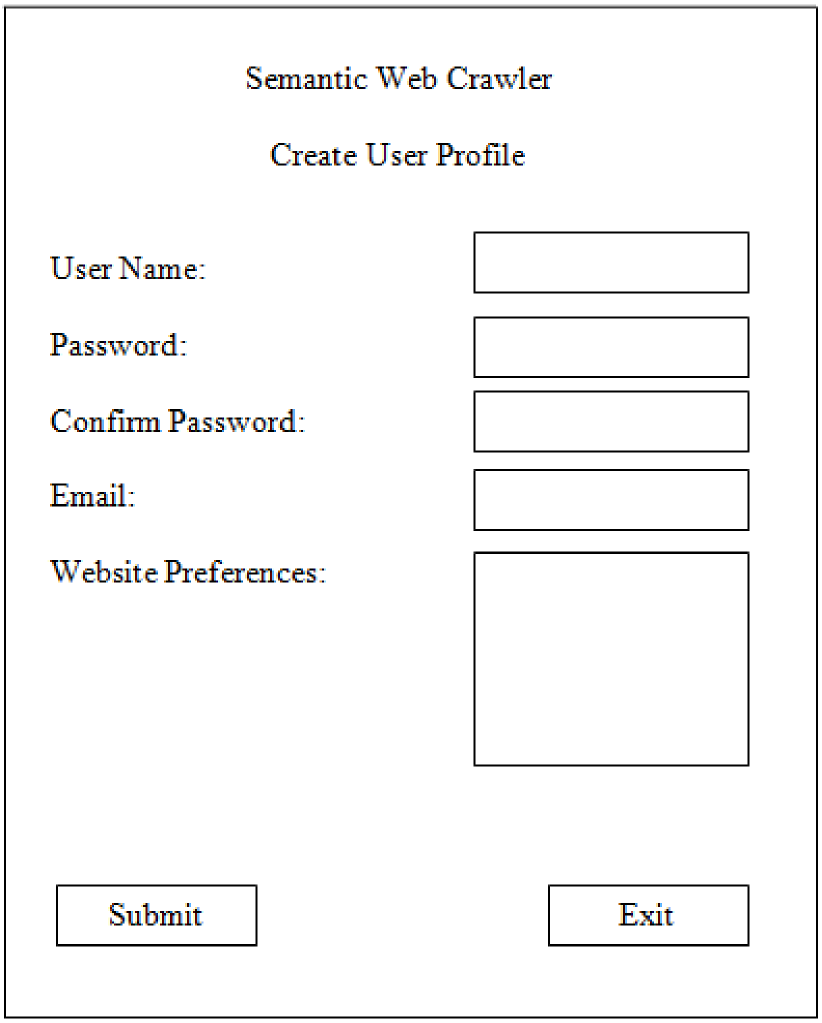
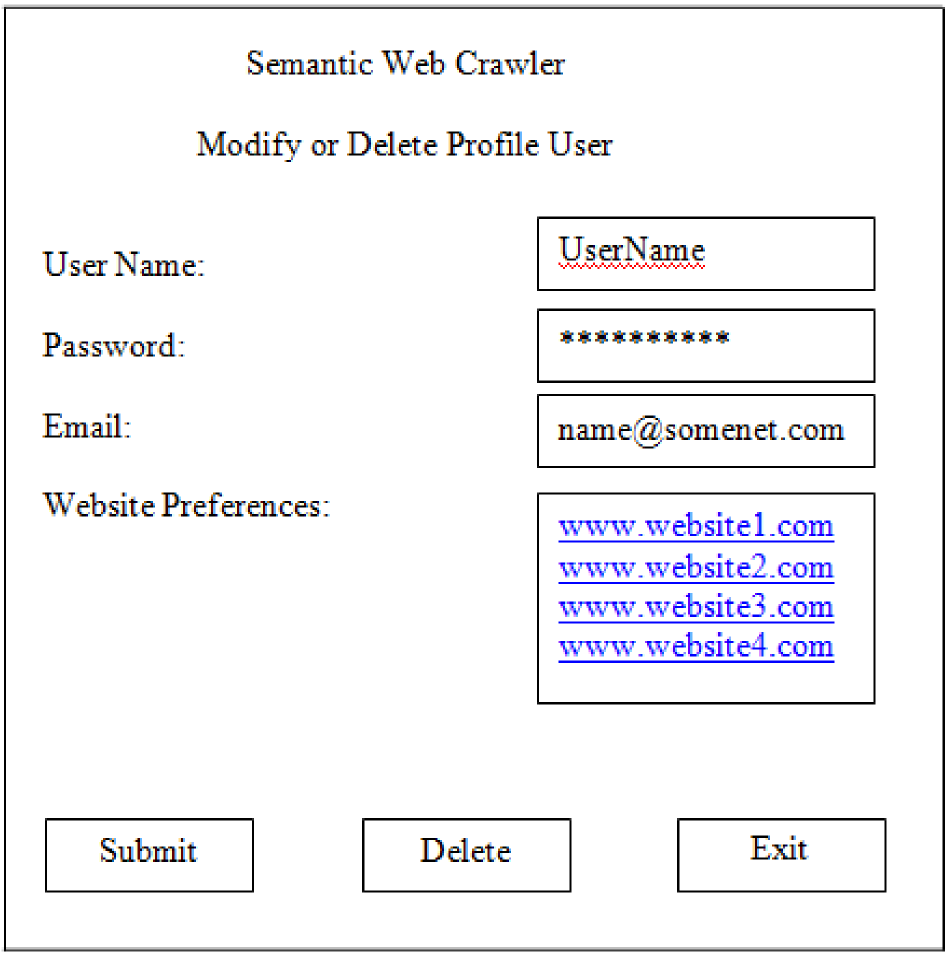


FIGURE 4.18 MODIFY OR DELETE USER PROFILE GUI


Semantic Web Crawler

Modify or Delete Profile User

User Name:	UserName
Password:	**********
Email:	name@somenet.com
Website Preferences:	www.website1.com www.website2.com www.website3.com www.website4.com

4.1.5.2 System Administration Chart Diagrams

This section contains the Class Diagrams and Sequence Charts for the System Administration, Create New User Profile, and Modify or Delete User Profile. See Figures 4.19 thru 4.24.

FIGURE 4.19 SYSTEM ADMINISTRATION CLASS DIAGRAM

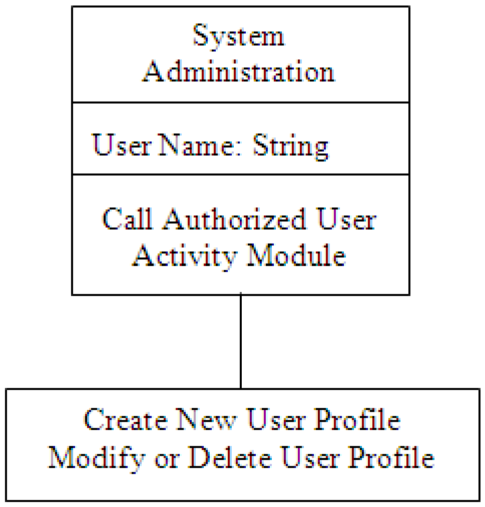

System Administration
User Name: String
Call Authorized User Activity Module
Create New User Profile
Modify or Delete User Profile

FIGURE 4.20 SYSTEM ADMINISTRATION SEQUENCE CHART

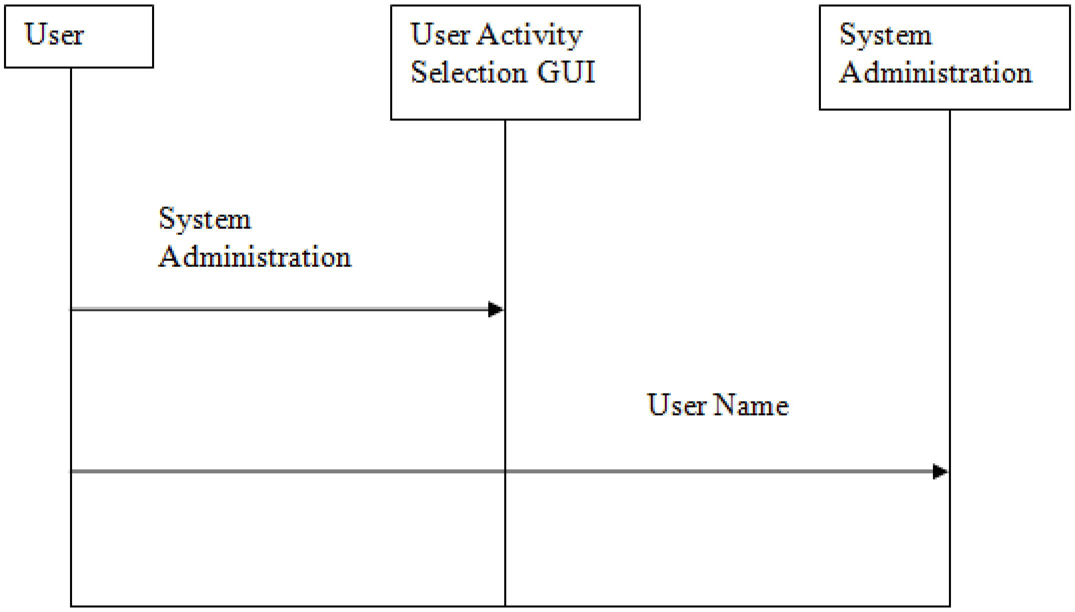


FIGURE 4.21 CREATE NEW USER PROFILE CLASS DIAGRAM

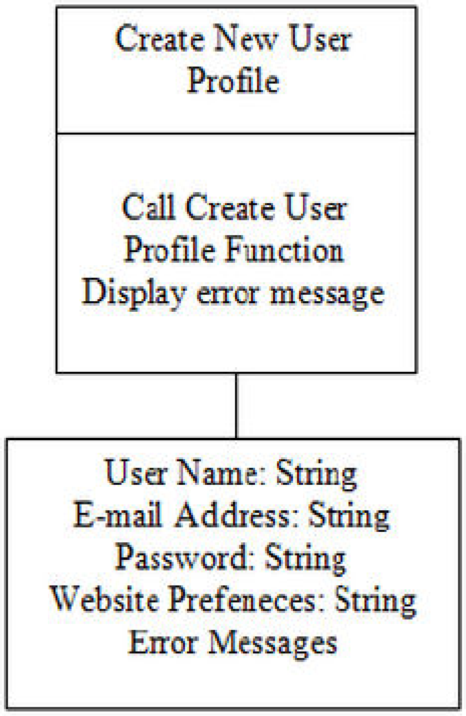


FIGURE 4.22 CREATE NEW USER PROFILE SEQUENCE CHART

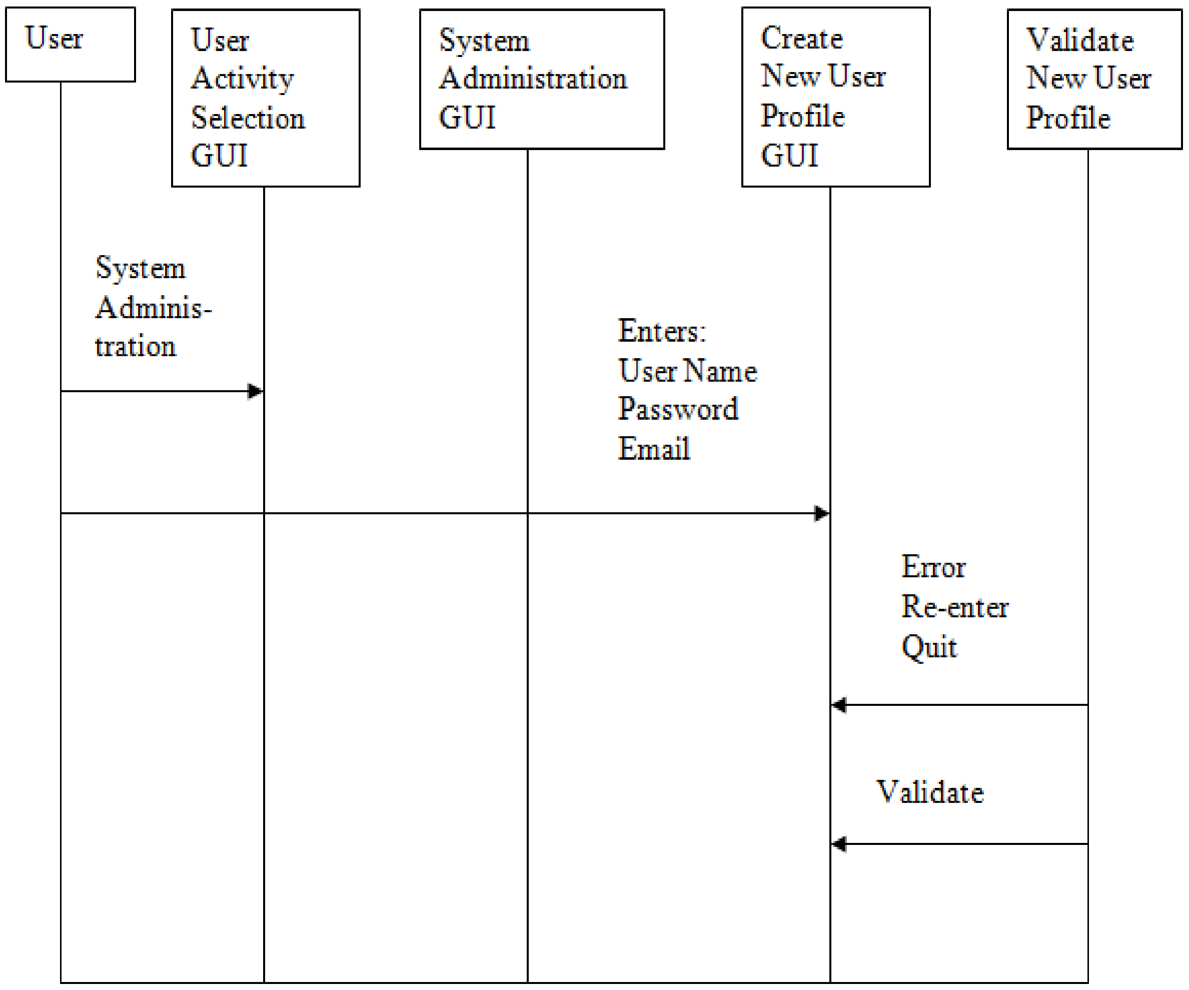

User
User Activity Selection GUI
System Administration GUI
Create New User Profile GUI
Validate New User Profile
System Adminis-tration
Enters: User Name Password Email
Error Re-enter Quit
Validate

FIGURE 4.23 MODIFY OR DELETE USER PROFILE CLASS DIAGRAM

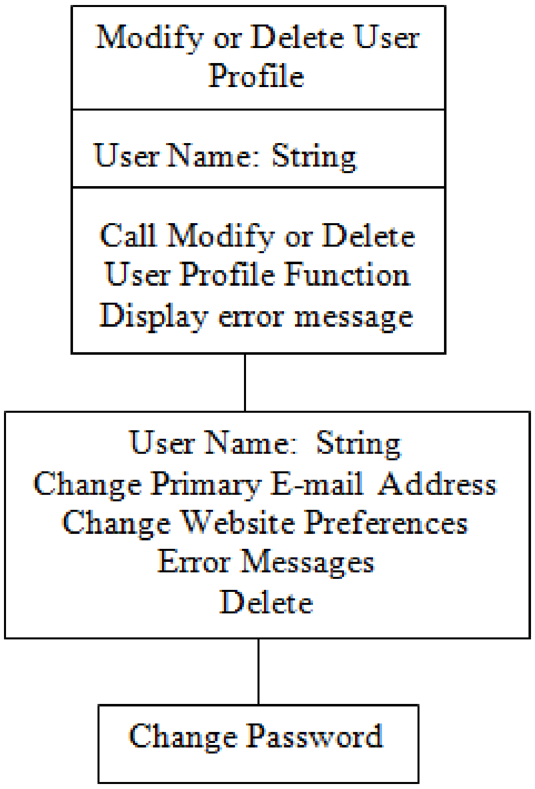
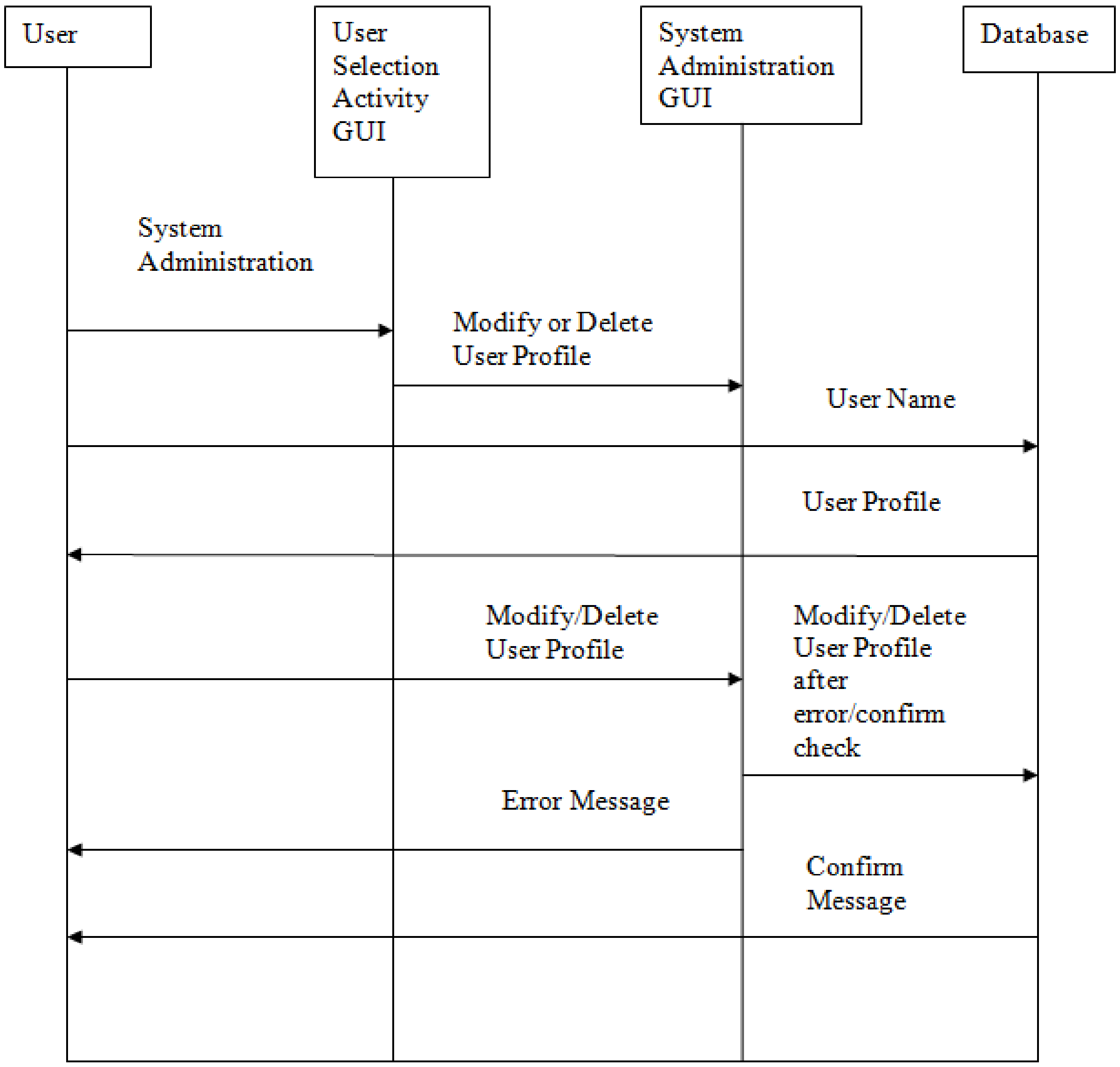

Modify or Delete User Profile
User Name: String
Call Modify or Delete User Profile Function
Display error message
User Name: String
Change Primary E-mail Address
Change Website Preferences
Error Messages
Delete
Change Password

FIGURE 4.24 MODIFY OR DELETE USER PROFILE SEQUENCE CHART


User
User Selection Activity GUI
System Administration GUI
Database
System Administration
Modify or Delete User Profile
User Name
User Profile
Modify/Delete User Profile
Modify/Delete User Profile after error/confirm check
Error Message
Confirm Message

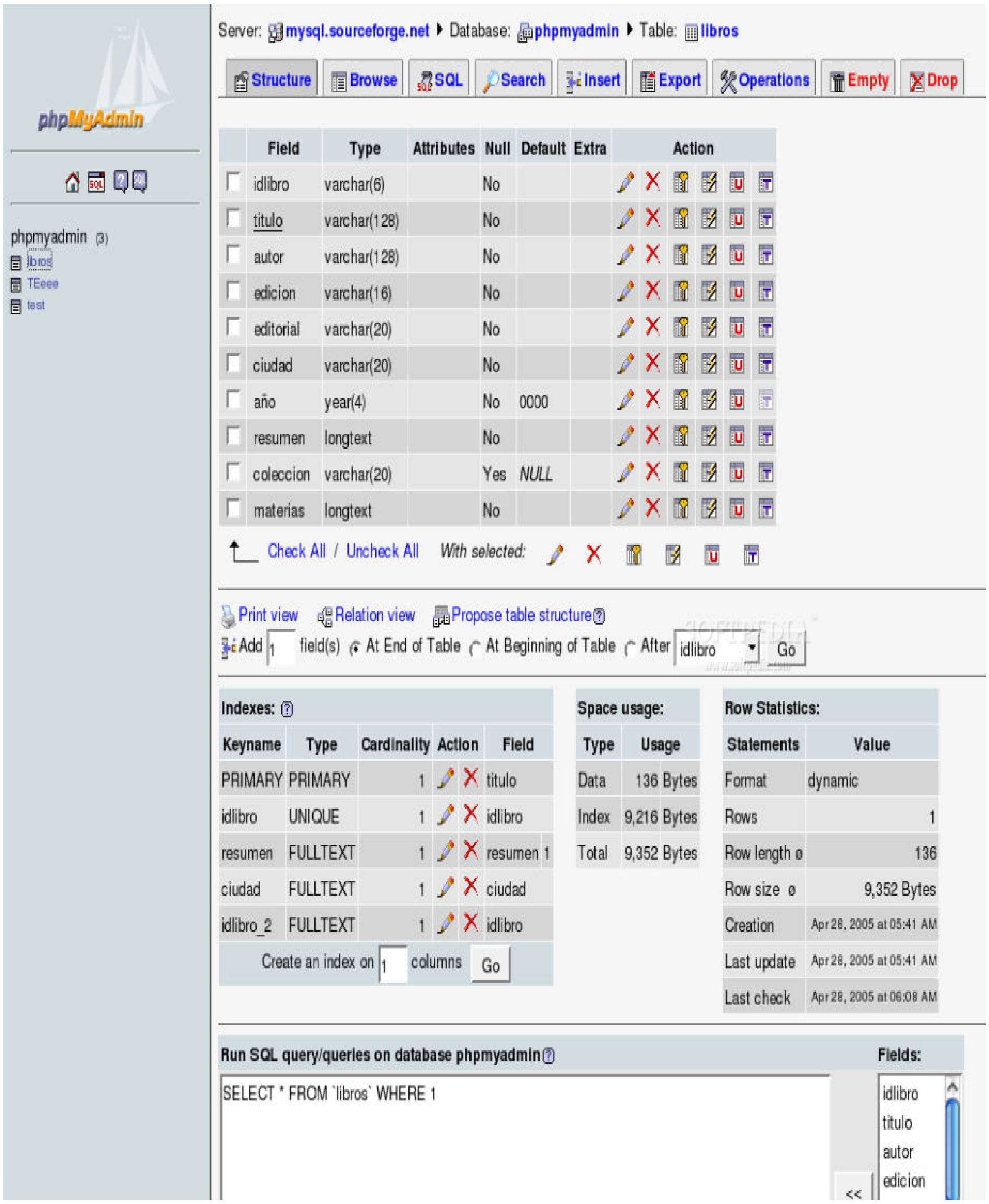
4.1.6 Database Management

The Database Management module shall provide access to the COTS Database product to exchange data with the application. The SWC Database shall be called with the COTS Database application and shall be presented to the user for use. All of the capabilities provided by the COTS product shall be provided to the user. The user can store the current task in the PC memory during or upon completion of the activity. When the work is complete, the user closes the application. The system shall prompt the user as to whether they want to save the current work or close the application without saving and execute the action selected.

4.1.6.1 Database Management Screens

This section contains a screen shot of the COTS Database used with SWC program. See Figure 4.25.

FIGURE 4.25 SCREEN SHOT OF COTS DATABASE GUI



Server: mysql.sourceforge.net ▸ Database: phpmyadmin ▸ Table: libros

	Field	Type	Attributes	Null	Default	Extra	Action
☐	idlibro	varchar(6)		No			
☐	titulo	varchar(128)		No			
☐	autor	varchar(128)		No			
☐	edicion	varchar(16)		No			
☐	editorial	varchar(20)		No			
☐	ciudad	varchar(20)		No			
☐	año	year(4)		No	0000		
☐	resumen	longtext		No			
☐	coleccion	varchar(20)		Yes	*NULL*		
☐	materias	longtext		No			

Check All / Uncheck All *With selected:*

Print view   Relation view   Propose table structure

Add 1 field(s) (• At End of Table (At Beginning of Table (After idlibro Go

Indexes:

Keyname	Type	Cardinality	Action	Field
PRIMARY	PRIMARY	1		titulo
idlibro	UNIQUE	1		idlibro
resumen	FULLTEXT	1		resumen 1
ciudad	FULLTEXT	1		ciudad
idlibro_2	FULLTEXT	1		idlibro

Create an index on 1 columns Go

Space usage:

Type	Usage
Data	136 Bytes
Index	9,216 Bytes
Total	9,352 Bytes

Row Statistics:

Statements	Value
Format	dynamic
Rows	1
Row length ø	136
Row size ø	9,352 Bytes
Creation	Apr 28, 2005 at 05:41 AM
Last update	Apr 28, 2005 at 05:41 AM
Last check	Apr 28, 2005 at 06:08 AM

Run SQL query/queries on database phpmyadmin

```
SELECT * FROM `libros` WHERE 1
```

Fields:
idlibro
titulo
autor
edicion

4.1.6.2 Database Management Chart Diagrams

This section contains the Database Management Class Diagram and Sequence Chart. See Figures 4.26 and 4.27.

FIGURE 4.26 DATABASE MANAGEMENT CLASS DIAGRAM

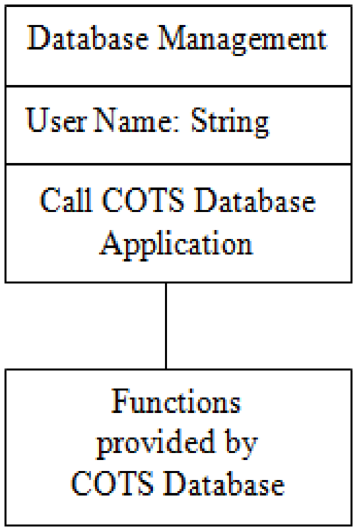
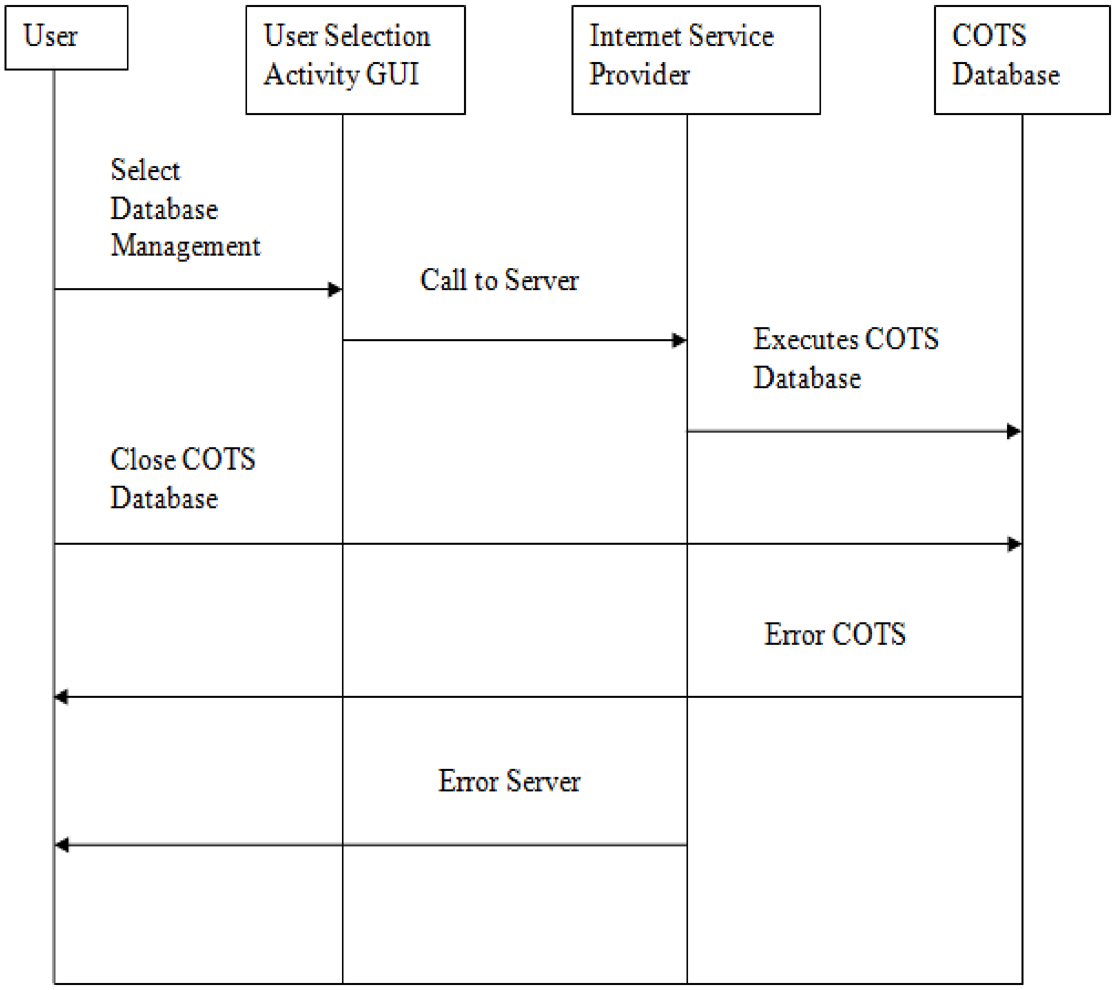

Database Management
User Name: String
Call COTS Database Application
Functions provided by COTS Database

FIGURE 4.27 DATABASE MANAGEMENT SEQUENCE CHART



4.1.7 Initiate Web Crawler

The Initiate Web Crawler module shall provide the user the capability to crawl the Internet for information specified by the user. The Initiate Web Crawler GUI shall be accessed by navigating the SWC User Activity Selection Menu GUI. The user can set the crawl parameters for the Internet crawl using data entry. When the Internet crawl is done with its crawl, the Initiate Web Crawler module presents the user with any matches found. The Initiate Web Crawler module shall be called with the COTS Database Application and is presented to the user for user. The user can store the current task in the PC memory during or upon completion of the activity. The user closes the application when the work is completed. The system shall prompt the user as to whether they want to save the current work or close the application without saving and execute the action selected. The COTS Database application shall returns the user to the Initiate Web Crawler GUI.

4.1.7.1 Initiate Web Crawler Screens

This section contains samples of the displays used for the Initiate Web Crawler. See Figures 4.28 and 4.29.

FIGURE 4.28 INITIATE WEB CRAWLER GUI

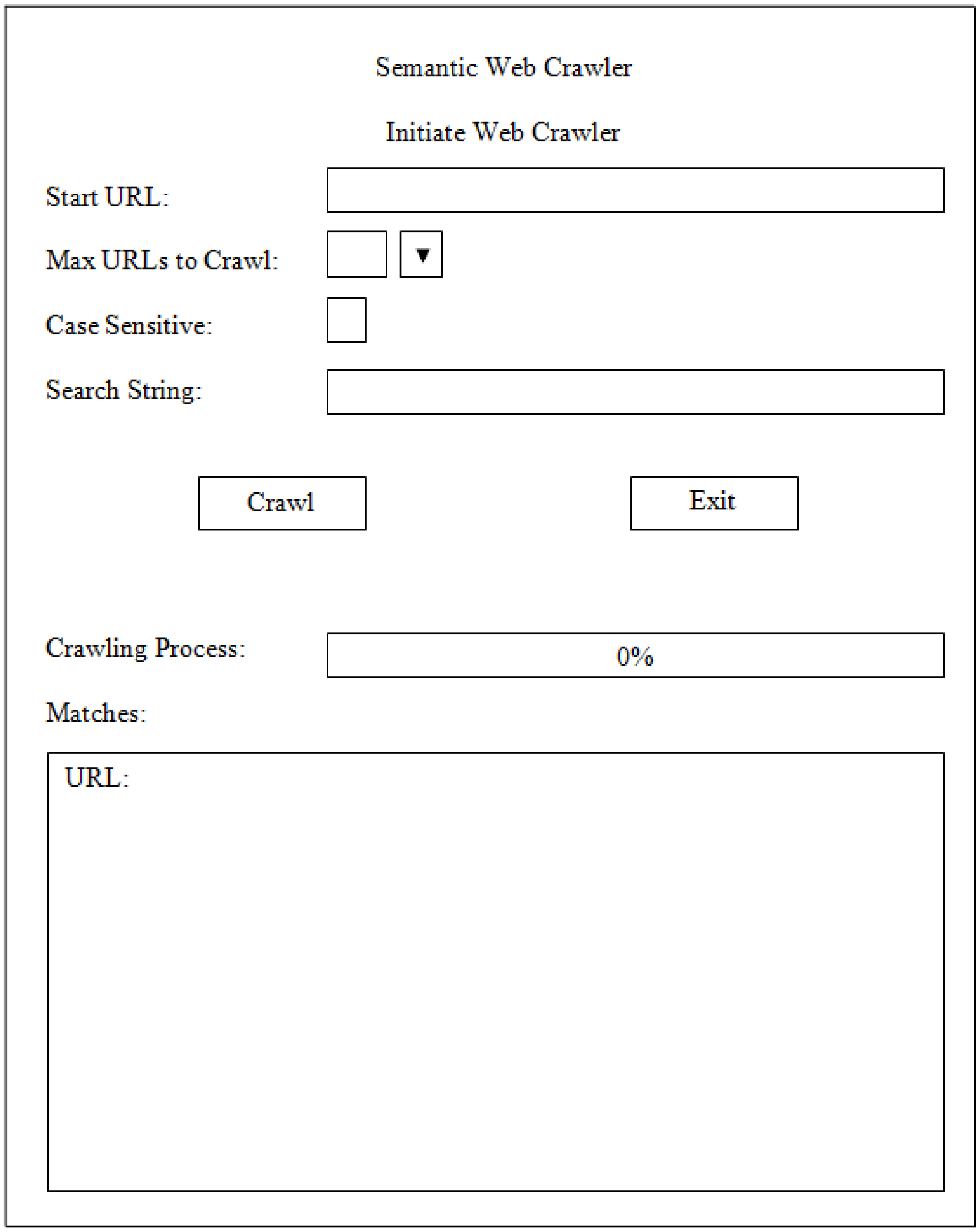
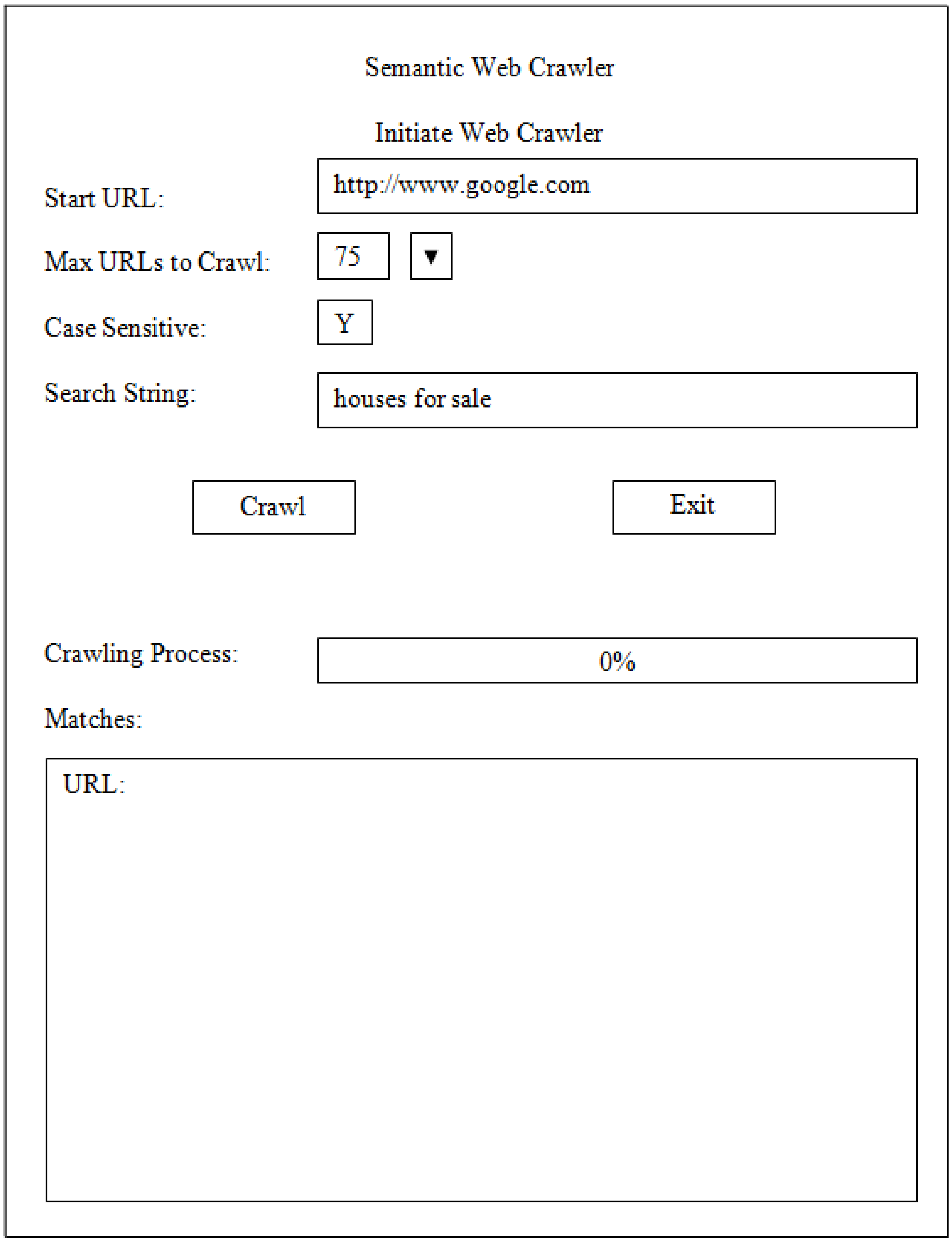


FIGURE 4.29 INITIATE WEB CRAWLER GUI WITH SEARCH PARAMETERS


Semantic Web Crawler

Initiate Web Crawler

Start URL: http://www.google.com

Max URLs to Crawl: 75 ▼

Case Sensitive: Y

Search String: houses for sale

Crawling Process: 0%

Matches:

URL:

4.1.7.2 Initiate Web Crawler Chart Diagrams

This section contains the Initiate Web Crawler Class Diagram and Sequence Chart. See Figures 4.30 and 4.31.

FIGURE 4.30 INITIATE WEB CRAWLER CLASS DIAGRAM

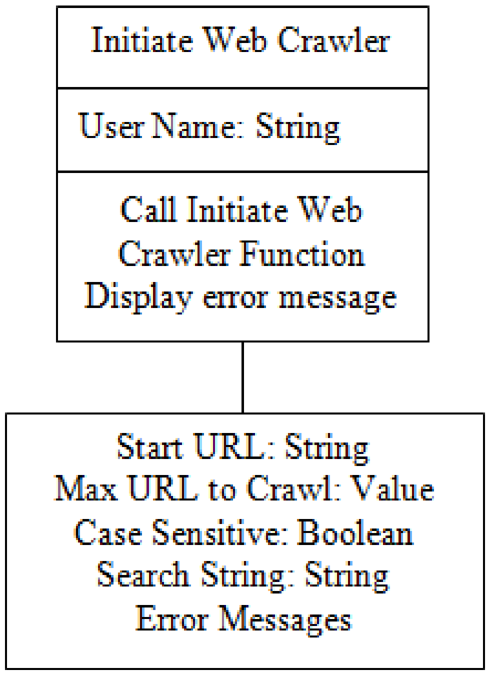
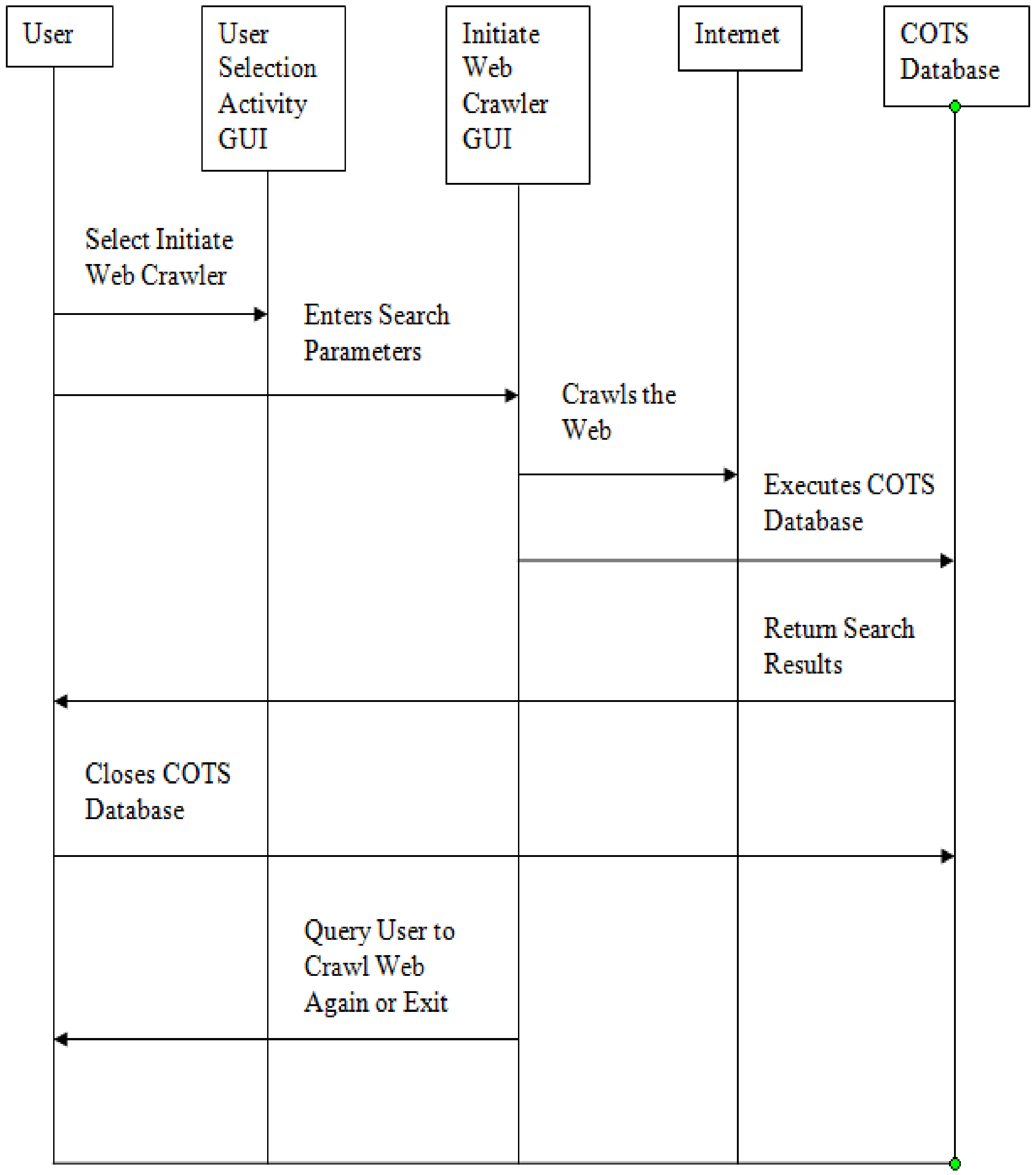


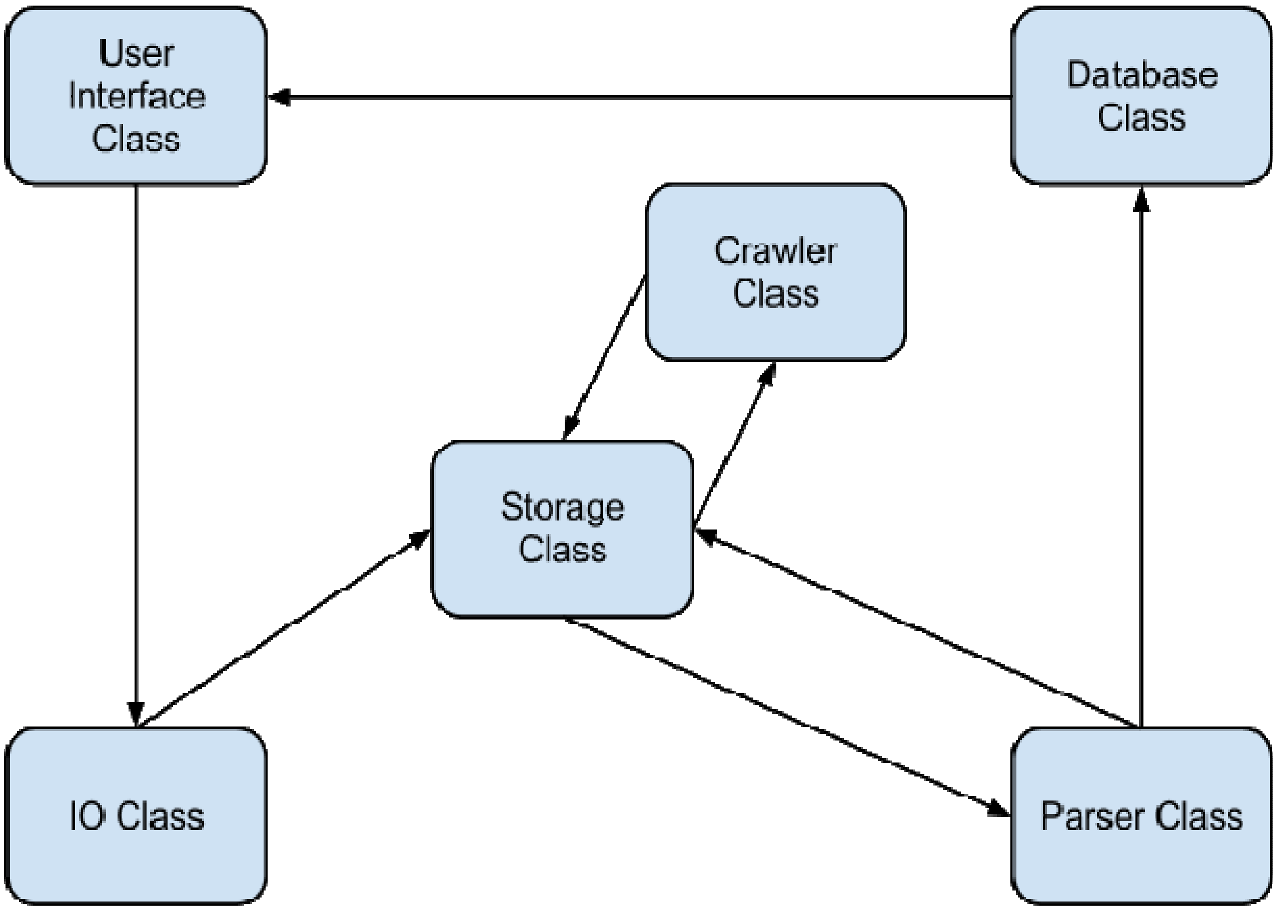
FIGURE 4.31 INITIATE WEB CRAWLER SEQUENCE CHART



4.2 Detailed Object Model

Figure 4.32 shows the diagram of the SWC object model.

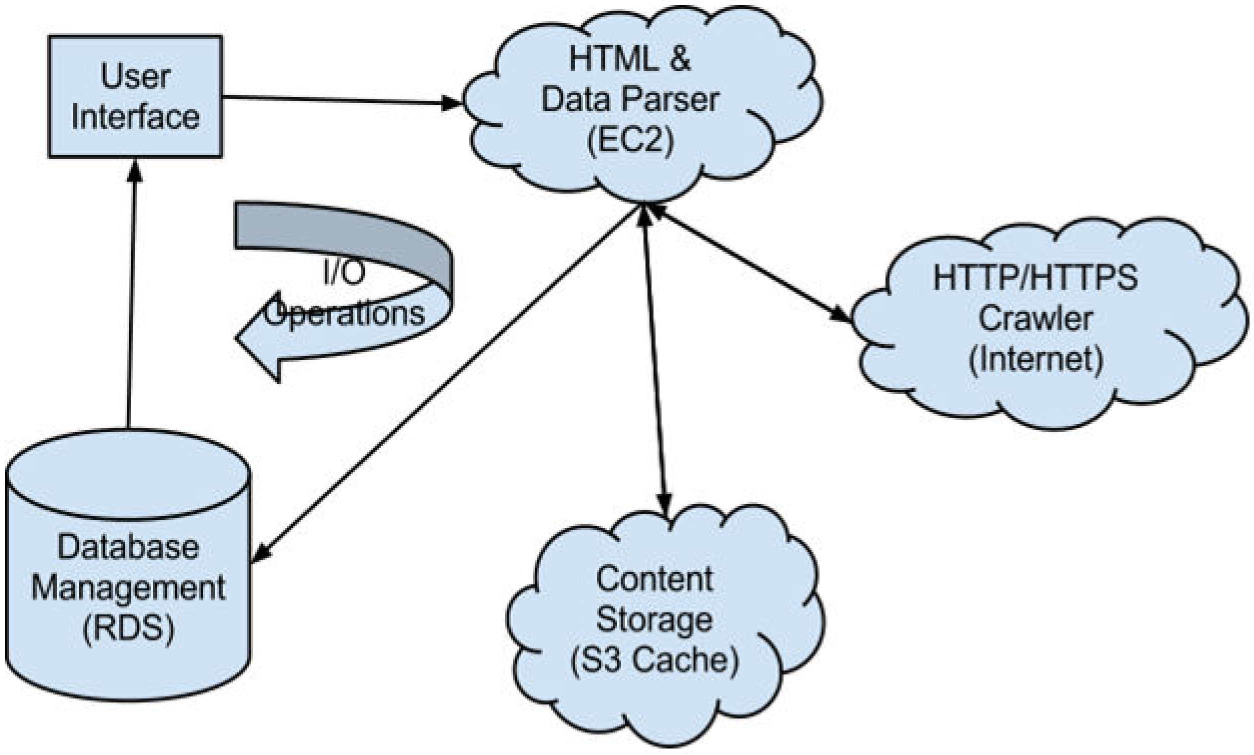
FIGURE 4.32 SWC OBJECT MODEL



4.3 Detailed Dynamic Model

Figure 4.33 shows the diagram of the SWC dynamic model.

FIGURE 4.33 SWC DYNAMIC MODEL


User Interface
HTML & Data Parser (EC2)
I/O Operations
HTTP/HTTPS Crawler (Internet)
Database Management (RDS)
Content Storage (S3 Cache)

4.4 Detailed Functional Model

Figures 4.34 thru 4.40 show the diagrams of the SWC functional model.

FIGURE 4.34 SWC USER LOGIN FUNCTIONAL MODEL

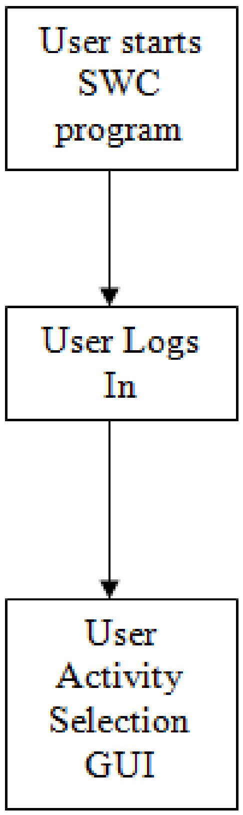


FIGURE 4.35 SWC USER REGISTRATION FUNCTIONAL MODEL

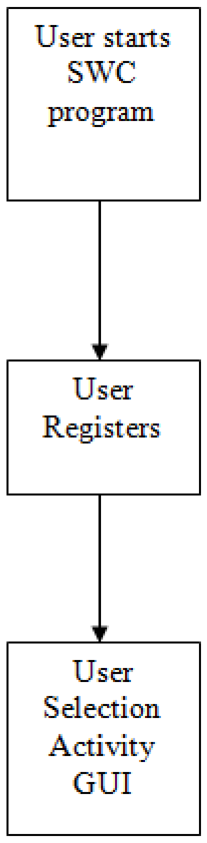


FIGURE 4.36 SWC USER ACTIVITY SELECTION GUI FUNCTIONAL MODEL

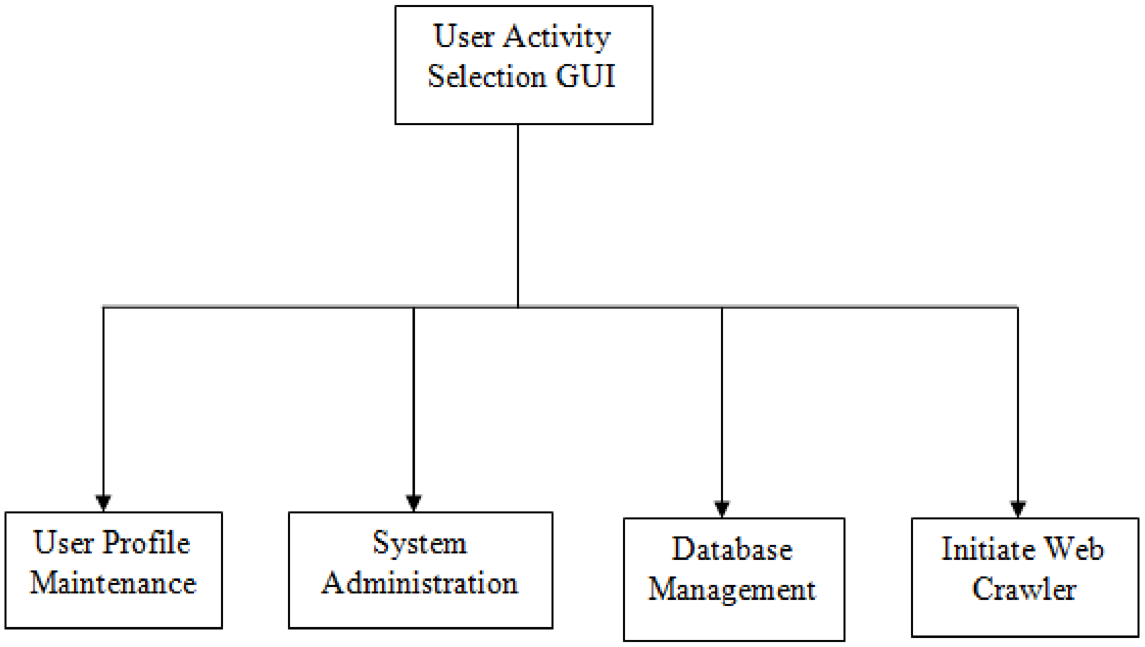


FIGURE 4.37 SWC USER PROFILE MAINTENANCE FUNCTIONAL MODEL

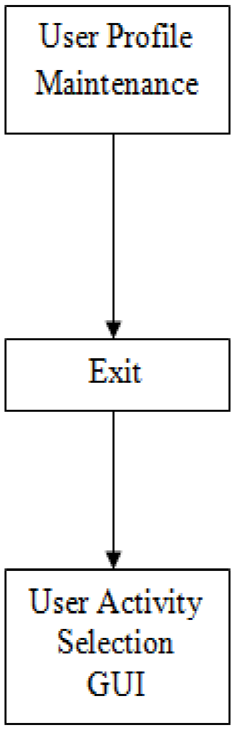


FIGURE 4.38 SWC SYSTEM ADMINISTRATION FUNCTIONAL MODEL

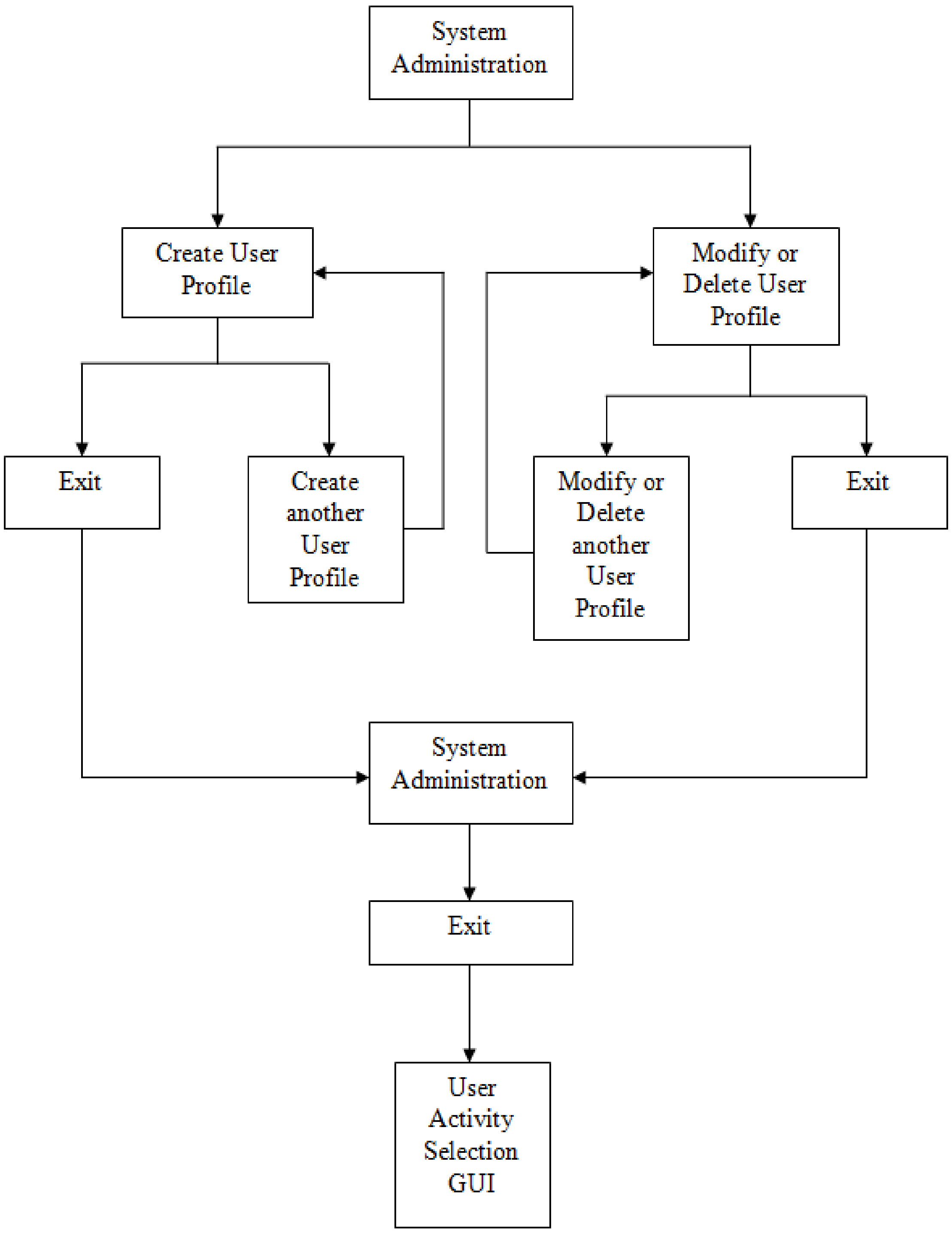

System Administration
Create User Profile
Modify or Delete User Profile
Exit
Create another User Profile
Modify or Delete another User Profile
Exit
System Administration
Exit
User Activity Selection GUI

FIGURE 4.39 SWC DATABASE MANAGEMENT FUNCTIONAL MODEL

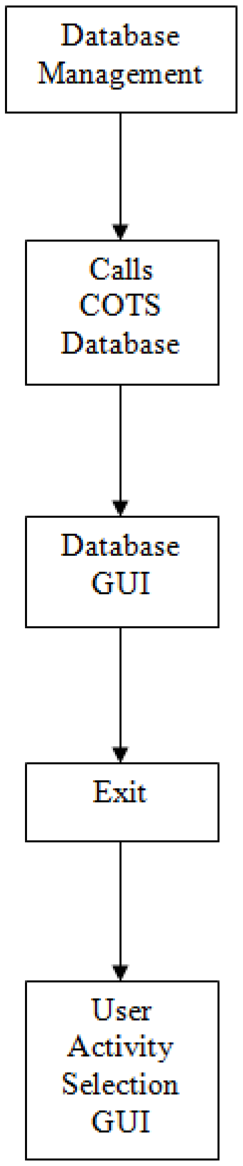
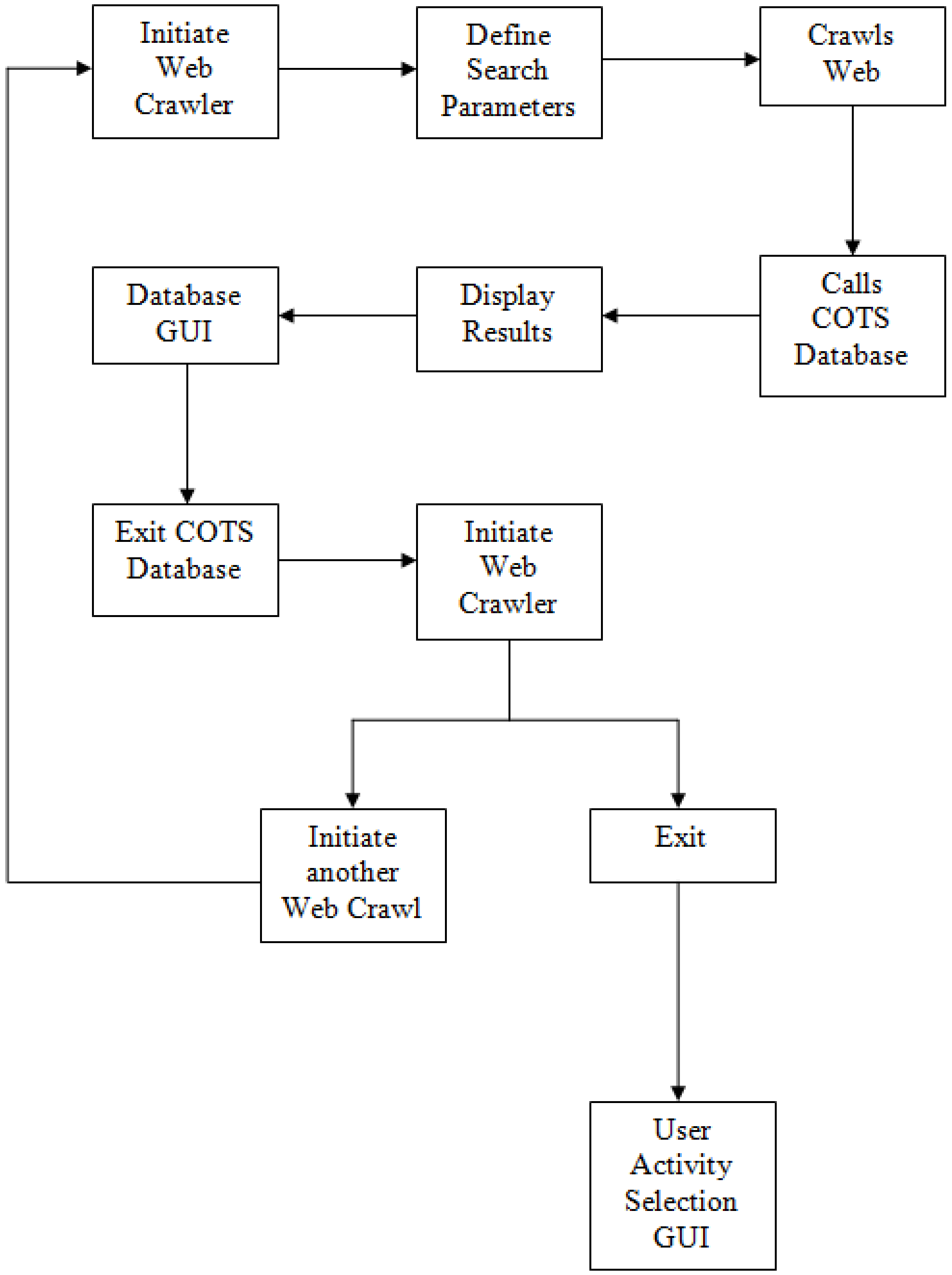


FIGURE 4.40 SWC INITIATE WEB CRAWLER FUNCTIONAL MODEL



4.5 HTTP/HTTPS Crawler Object

This Object will interact with the Internet and making sure that data is pulled in a quick and orderly manner. It will not try to interpret data but rather create the necessary sockets and systems necessary to merely grab it.

4.5.1 Get HTTP Class

Due to the requirements of creating the necessary sockets to download a particular web page using the standard HTTP Get and Post Protocols, the below methods are necessary.

4.5.1.1 Download Url Method

4.5.1.1.1 Narrative

This method carries the job of fetching the URL and creating a hash tag to facilitate retrieval for the page. This then writes the file to a load page that is stored in the cache.

4.5.1.1.2 Input Data Elements

The input functionality is injected into the function as hash tags attached to the URL.

Name	Description	Data Type	Purpose	Source
Cached Path	Hash of Url and Cache Folder	const char	To pass on to next method	const char *path
Url	Address to download	string	To download a web page	string url
Url Hash	Hash of Url	string	File Name to save Url Page	string urlhash

4.5.1.1.3 Output Data Elements

The output is saved into cache for later retrieval by the user in the form of a path, URL, and URLhash.

Name	Description	Data Type	Purpose	Source
Cached Path	Hash of Url and Cache Folder	const char	To find the downloaded file	const char *path
Url	Address to download	string	To pass along to load page	string url
Url Hash	Hash of Url	string	File Name to grab from	string urlhash

4.5.1.1.4 Processing Characteristics

The processing characteristics are pointers to the path and string formatting of the URL's.

4.5.1.1.4.1 Algorithms

```
function starts {
        Create a string named "command" that outputs a WGET command with the url
                and the hash of that url into the cache folder.
        Execute that command string into the system's command line
        Run the Load Page Method with that same url and hash you used initially.
} function ends
```

4.5.1.1.5 Utilization of Other Elements

The LoadPage element shall be utilized in the process of pagination and loading of the pages.

4.5.1.1.6 Limitations

The limitations of the function are restricted to active URL's.

4.5.2 Get Proxy Class

Due to the requirements of pushing all of the data through the necessary ports in order to properly crawl a web page via a proxy of some kind, this process is required when a site begins to block the crawl process; this class will have to be developed at a later date.

4.5.3 Get Cookies Class

Due to the requirements of handling session related information such as saving a user state when visiting a web site or page throughout the crawl process. Sessions may even be saved in cache for reuse later; this class will have to be developed at a later date.

4.5.4 Get Post Param Class

Due to the requirements of putting together the necessary data given either through arguments or the parsing process recursively to allow for features like pagination or submit functions in the crawling process that use a post method of moving from page to page, this class will have to be developed at a later date.

4.6 I/O Operations Object

This Object will take in arguments and interpret them allowing the application to understand what it needs to do in all of its other components.

4.6.1 Argument Interpreter Class

Due to the requirements of taking the interpreted arguments and give them life, it will call all the necessary Sub Classes in the Parser Class to handle not only the flow of parsing but the speed and efficiency allowing functions to become recursive when they need to be, this class will have to be developed at a later date, the below methods are necessary.

4.6.1.1 Process Arguments Method CSC

4.6.1.1.1 Narrative

The Process Arguments method shall process the string inputs that are inputted by the user. The Process Arguments method shall crawl the Internet searching for any matches of the string inputs.

4.6.1.1.2 Input Data Elements

The input data elements shall be inputted by the user. The data elements entered by the user can be a one word string, several word strings, and/or values.

Name	Description	Data Type	Purpose	Source
Inputs	User or System entered inputs	std::string	To pull in arguments	std::string inputs

4.6.1.1.3 Output Data Elements

The output data elements of the Process Arguments method shall be any matches of the string inputs that are inputted by the user.

Name	Description	Data Type	Purpose	Source
Url	Address to Download	string	To download the page	string url
Regex (regular expression)	Items to be parsed	vector<string>	To handle parsing	vector<string> regex
Pagination	Url for Pagination	string	To handle pagination	string paginate
Output	Database Information	string	To output the data	string output

4.6.1.1.4 Processing Characteristics

The processing characteristics of the Process Arguments method are the processing of the string inputs and finding matches of the string input in the Internet.

4.6.1.1.4.1 Algorithms

```
function start {
    switch statement
        Process a Link Argument and go to generate hash method
        Process a Data Argument and go to parse Page method
        Process a Pagination Argument
        Process a Database Argument for Output
} function ends
```

4.6.1.1.5 Utilization of Other Elements

The elements that are utilized in this function are the Internet crawl and finding matches of the string inputs. The Parse Page method is also utilized by parsing the pages that match the string inputs.

4.6.1.1.6 Limitations

The limitations of the method are restricted to Web pages that are not blocked.

4.6.2 Links Argument

Due to the requirements of translating the arguments involved with the necessary links and what sites will be crawled by the crawler, this class will have to be developed at a later date.

4.6.3 Items Argument

Due to the requirements of translating the arguments involved with when information is considered an item in a page with multiple data points that may need collecting, this class will have to be developed at a later date.

4.6.4 Data Argument

Due to the requirements of translating the arguments involved with the individual data points themselves as well as how and what name they are stored under within an item, this class will have to be developed at a later date.

4.6.5 Pagination Argument

Due to the requirements of translating the arguments involved with the moving from one page to another and whether the movement is singular (in the case of collecting more data points for a specific item) or as a continuation of the item mass as a whole (in the case of moving to page 2 or 3 in order to collect more items), this class will have to be developed at a later date.

4.6.6 Proxy Argument Class

Due to the requirements of translating the arguments involved with multiple proxer server and port information fed to the system in the event that a site tries to block the crawl, it will be smart enough to use multiple proxies if inputted to attempt a re-crawl again and again until it finds one that works, this class will have to be developed at a later date.

4.6.7 Cache Argument Class

Due to the requirements of translating the arguments involved with where and how to cache the page information as well as it will allow us to determine the length of cache time so if a page is too old and needs to be re-downloaded, this argument will determine the length and age of the requirements, this class will have to be developed at a later date.

4.6.8 Scale Argument Class

Due to the requirements of translating the arguments involved with allowing the crawl to scale across multiple instances, it will separate each link and its necessary arguments into those instances so that the system can quickly and effectively crawl multiple pages instantly across multiple servers; this class will have to be developed at a later date.

4.6.9 Parameters Argument Class

Due to the requirements of translating the arguments involved with setting and collecting the parameters necessary to handle pagination through the Post process in the crawler download method. It will determine what parameters need to be collected and in what order they will be listed, this class will have to be developed at a later date.

4.6.10 Output Argument

Due to the requirements of translating the arguments involved with the output of the collected data, this class will have to be developed at a later date.

4.7 Content Storage Object

This Object will take the data that's downloaded from the web by the crawler and ensure that it's properly up to date. It will save the data with a hash name of the web address the file was pulled from so that the system can quickly figure out and understand the data pulled.

4.7.1 Cache Generator Class

Due to the requirements of taking the data and hash values and making sure that they are properly saved and sorted in the cache folder, the below methods are necessary.

4.7.1.1 Check For Cache Folder Method

4.7.1.1.1 Narrative

The Check For Cache Folder method shall check to see if there is already an existing cache folder in the database with the data found from the matching string inputs. If there is no existing cache folder, then a cache folder for the data is created. If there is already an existing cache folder, then no new cache folder is created.

4.7.1.1.2 Input Data Elements

The input data elements are nonexistent for this method.

4.7.1.1.3 Output Data Elements

The output data elements are nonexistent for this method.

4.7.1.1.4 Processing Characteristics

The processing characteristics of the Check For Cache Folder method are the processing of checking for an existing cache folder for the matched string inputs and if no cache folder exists then one is created.

4.7.1.1.4.1 Algorithms

```
function start {
        If statement to check whether or not “cache” folder exists in main directory.
                when the folder does not exist then run MKDIR command and grant folder all
                  permissions...
} function ends
```

4.7.1.1.5 Utilization of Other Elements

The Check For Cache Folder method does not utilize any other elements.

4.7.1.1.6 Limitations

The limitation of the Check For Cache Folder method is the amount of memory available.

4.7.1.2 Generate Url Hash Method

4.7.1.2.1 Narrative

The Generate Url Hash method shall generate a hash table for the URLs. The URL hash table shall be used to increase cache efficiency.

4.7.1.2.2 Input Data Elements

The input data elements are the URLs and shall be put into a hash table.

Name	Description	Data Type	Purpose	Source
Url	Url to be hashed	string	To turn Url into a hash and path	std::string url

4.7.1.2.3 Output Data Elements

The output data elements are the hash tables of the URLs.

Name	Description	Data Type	Purpose	Source
Path	Path with Hash	const char*	To help system find path to file	const char *p
Url	Original Url	string	To allow system to be recursive	string url
Url Hash	Hash of Url	string	To allow system to store cache	string hash

4.7.1.2.4 Processing Characteristics

The processing characteristics of the Generate Url Hash method are the generated URL hash tables. The use of the URL hash tables shall increase cache efficiency.

4.7.1.2.4.1 Algorithms

```
function start {
    Create all of the necessary has char variables
    Take incoming Url and generate a Url Hash from it
    Pass what is a long integer hash into string form
    Create a Path by tacking the “cache/” older title and then adding the hash to it
    Turn String back into a constant char
    Print and then send all three new data variables to Load Page Method.
```

} function ends

4.7.1.2.5 Utilization of Other Elements

The loadPage method shall be utilized in the process of pagination and loading of the hash tables.

4.7.1.2.6 Limitations

The limitations of the Generate Url Hash method are restricted to active URLs.

4.7.2 Cache Filter Class

Due to the requirements of pulling the hash data from the cache and loading it up to the necessary parser functions, the below methods are necessary.

4.7.2.1 Load Page Method

4.7.2.1.1 Narrative

The Load Page method is utilized to extract the page that is required form the software program. This function shall retrieve the page from the file system if it is available at that location. If the page does not reside in the file system, the function shall call and utilize the download URL function to extract the page required.

4.7.2.1.2 Input Data Elements

The input elements are the path, URL, and the hashed version of the URL. These include pointers read files commands.

Name	Description	Data Type	Purpose	Source
Path	Path with Hash	const char*	To help system find path to file	const char *p
Url	Original Url	string	To allow system to be recursive	string url
Url Hash	Hash of Url	string	To allow system to store cache	string hash

4.7.2.1.3 Output Data Elements

The output elements are the line of the file system that are required for the program calls. These include file check status and end of file characteristics.

Name	Description	Data Type	Purpose	Source
Path	Path with Hash	const char*	To allow system to be recursive	const char *p
Url	Original Url	string	To allow system to be recursive	string url
Url Hash	Hash of Url	string	To allow system to be recursive	string hash
Global Page	HTML Page from Url	vector<string>	To allow system to store Page	vector<string> page

4.7.2.1.4 Processing Characteristics

The processing characteristics are pointers that direct to paths that are given for the file system. Also, the hashed URL system is utilized in the event that the path to the file system produces an empty file. In this case, the download URL function is implemented.

4.7.2.1.4.1 Algorithms

```
function start {
        Use IfStream to download file using the Path data variable
        If Statement is created to open IfStream
                While statement used to output contents of file into a vector of string
                close the IfStream
        Else if file doesn’t exist then go to Download Url Method
} function ends
```

4.7.2.1.5 Utilization of Other Elements

The elements that are utilized in this function are the file streams and the file checking system to determine when and if the system file is complete or empty.

4.7.2.1.6 Limitations

Limitations of the function shall be implementing a pointer to a path directing to the file system and also the function calls to the download URL method.

4.7.3 Cloud (S3) Cache Integration Class

Due to the requirements of integrating the cached environment with Cloud solutions such as S3 and DynamoDB, it will allow the quick and efficient processing of data across cloud systems for the purpose of scalability, this class will have to be developed at a later date.

4.7.4 Cache Argument Interpreter Class

Due to the requirements of outputting data from the Cache Argument and separating it into its necessary components, this class will have to be developed at a later date.

4.8 HTML & Data Parser Object

This Object will take the data that's been downloaded and cached by the system so far and figuring out how to parse it for the necessary information using the arguments introduced through the IO class.

4.8.1 Parsing HTML Pass-through Class

Due to the requirements of moving amongst pages as the argument is given, so if the data is two or three pages into a selection, this system will pick out the necessary Url's necessary to move across multiple pages to grab the data the user is looking for; this class will have to be developed at a later date.

4.8.2 Parsing HTML Data Pages Class

Due to the requirements of parsing of the actual items and data points throughout the page as well as handle the pagination arguments as they come, the below methods are necessary.

4.8.2.1 Parse Page Method

4.8.2.1.1 Narrative

The parse page method implements the job of going line by line through the page and parsing out the required data using the regex (regular expression) method call function. This function is important in separating the required data from the unusable data and utilizing the storage to secure the parsed data.

4.8.2.1.2 Input Data Elements

The input data elements are the vector of strings that utilize a parse method called regex. Also, the system shall read the file to determine if the file is read line by line.

Name	**Description**	**Data Type**	**Purpose**	**Source**
Global Page	HTML page from Url	vector<string>	To load HTML line by line	vector<string> page

4.8.2.1.3 Output Data Elements

The output data elements are the parseLine function that utilizes a line pointer and also a function call to the regex function.

Name	Description	Data Type	Purpose	Source
Page Line	HTML of Url per Line	string*	To help parse page line by line	string *line
Regex	Regex Commands	vector<string>	To store the needed Regex	vector<string> regex

4.8.2.1.4 Processing Characteristics

The processing characteristics are that the function goes to the file system and reads a file line by line to utilize the regex parsing function to parse out the required data. Also, there is a checking system to make sure that the entire file is read.

4.8.2.1.4.1 Algorithms

```
function start{
        For statement that Parses through each line of the Page Vector
                Send both the HTML Line and the Regex Vector to the Parse Line Method
} function ends
```

4.8.2.1.5 Utilization of Other Elements

Other elements that shall be used as necessary shall be function calls to the regex method to start the parsing process. A file checking system shall be implemented to review the file for completeness.

4.8.2.1.6 Limitations

The limitations of the method shall be used to check the file system and also to call the parseLine function in order to parse out the required data.

4.8.2.2 Parse Line Method

4.8.2.2.1 Narrative

The basic concept of this method is to take single lines of HTML code and compare them to the simple regex method. This allows us to run multiple regex queries on a single line so if multiple bits of data are all on the same line we can locate and find it all with ease.

4.8.2.2.2 Input Data Elements

The two basic data elements being brought into this field are the actual string html lines that we are filtering through in the Parse page method as well as the regex vector were all of our individual regex queries can be compared to the individual lines of text.

Name	Description	Data Type	Purpose	Source
Page Line	HTML of Url per Line	string*	To help parse page line by line	string *line
Regex	Regex Commands	vector<string>	To store the needed Regex	vector<string> regex

4.8.2.2.3 Output Data Elements

The only data to be outputted is the information that has been sent to the RegexLine function. This can be done quickly and easily in comparison.

Name	Description	Data Type	Purpose	Source
Page Line	HTML of Url per Line	string	To pass page line	string line
Regex Pattern	Individual Pattern to Filter	string*	To pass the needed pattern	string *pattern

4.8.2.2.4 Processing Characteristics

The items being processed are simple, the regex comparisons are first sent through a for loop where the individual line is paired up with the regex itself. Certain items are removed like spaces and then sent to the regexLine() function.

4.8.2.2.4.1 Algorithms

```
function start{
    For statement that Parses through the Regex Vector for each individual Pattern
        Send both the HTML Line and the Pattern to the Regex Line Method
} function ends
```

4.8.2.2.5 Utilization of Other Elements

No other elements are utilized in this method.

4.8.2.2.6 Limitations

The only real limitation is that of the parsing that needs to be done in a sub function and thus this method does not itself handle any of the regex or parsing of the line.

4.8.2.3 Regex Line Method

4.8.2.3.1 Narrative

This method is where the rubber meets the road, where we take what has been passed along to us both by the parse Line functions as well as the regex vector and we do the physical parsing of the line for a particular regex request.

4.8.2.3.2 Input Data Elements

In this method we now receive the individual line of HTML text and the individual regex that first needs to be compiled by the regexcomp() function before it then gets parsed by the regexexec() functions.

Name	Description	Data Type	Purpose	Source
Page Line	HTML of Url per Line	string	To pass page line	string line
Regex Pattern	Individual Pattern to Filter	string*	To pass the needed pattern	string *pattern

4.8.2.3.3 Output Data Elements

For the most part the data output here is the final data we want to collect for the crawler. We get to take a look at what all the hard work has gone and where the crawler can really do its magic properly.

Name	Description	Data Type	Purpose	Source
Data	Matched Data	string	To exit any matched data	string data

4.8.2.3.4 Processing Characteristics

The processing required in this method include running your regex query through a regex compiler method first before then running it through a regex execution method that takes and figures out what the compiler has put together for us.

4.8.2.3.4.1 Algorithms

```
function start {
        List number of Regex Matches and great a regex_t variable
        Run both the regex_t variable and the pattern through the regcomp method
        If statement that takes modified regex_t and line and run both through regexec method
                When matches are found print them to the screen and add it to data variable for
                export
        Run regfree method in order to free memory
} function ends
```

4.8.2.3.5 Utilization of Other Elements

The elements we use that are external include the added regex compiler and execution functions that are found within the standard regex includes found in C++.

4.8.2.3.6 Limitations

Our only real limitation is how well we can select our regex to be able to parse the data; the less we understand the necessary regex needed to go through this crawler the less we can do.

4.9 Database Management

This Object will figure out how the parsed data will output, though it's called the database class it does not mean that the information will always be saved in database form.

4.9.1 Database Argument Interpreter

Due to the requirements of taking the interpreted arguments given in the database selection and go through the process and options the user selected to go through once the site itself has been crawled, this class will have to be developed at a later date.

4.9.2 Save to File

Due to the requirements of saving the data to file, in this case a CSV file separated by semi colons with the actual data inside parentheses, the below methods are necessary.

4.9.2.2 Save As CSV Method

4.9.2.2.1 Narrative

This method has a very simple job, it takes each individual set of collected data that has been placed inside the vector matrix (vector inside of a vector) and writes it to a file that is essentially gives the data in a usable format for the end user.

4.9.2.2.2 Input Data Elements

The Input of the vector matrix for Data is inputted into the method and is then separated and sorted by the system to create a CSV file.

Name	Description	Data Type	Purpose	Source
Data	2 dimensional Vector	vector<vector<string> >	To Store relevant output data	vector<vector<string> > &data

4.9.2.2.3 Output Data Elements

The Output of the formulated vector matrix is saved to file with each data element being separated by a semi-colon and the main vector titles are listed at the very top of the file.

Name	Description	Data Type	Purpose	Source
Data File	Output of completed data	fout	To output nested vectors	fout Stream

4.9.2.2.4 Processing Characteristics

Essentially it's rotating the vector matrix so that the main vector titles are listed at the top and the secondary vector is listed below it.

4.9.2.2.4.1 Algorithms

```
function start {
    Create a 2d array or vector
    use FOpen to create a file in the system
    For loop to go through first rows of 2d variable
        For loop to then nest and go through columns of each row
            print individual data element to file
            Repeat Process to read the file's content back on the screen
            FClose to save and close the file
} function ends
```

4.9.2.2.5 Utilization of Other Elements

Only the File out stream will be used to move the individual sections of data from the matrix and into the file itself.

4.9.2.2.6 Limitations

The limitations of the method will essentially fall under the formatting of the CSV file itself as the separator format and item location.

4.9.3 Save to SQL Database

Due to the requirements of saving the data to a SQL database, once a connection to the database has been established by the Database Argument Interpreter, this class will have to be developed at a later date.

4.9.4 Save to NoSQL Database

This Class will handle the necessary functions involved with setting up or creating a Table in a NoSQL database. It will then do all the necessary functions to transition the data over to those tables as necessary; this class will have to be developed at a later date.

4.9.5 Output On Screen Data (Debug)

Due to the requirements of debugging on screen data dump to see if your arguments worked without necessarily needing to save the data to a file or database. This allows the developer building both the crawler core as well as the User Interface to see what the output for a set of commands are going to be before they implement them, this class will have to be developed at a later date.

4.10 User Interface

This Object will first understand what the user is doing and interpret that information into a system that the IO class can understand.

4.10.1 User Administration

Due to the requirements of creating what a user sees and how they interact with the system, this class will have to be developed at a later date.

4.10.2 System Administration

Due to the requirements of creating what a System Administrator sees and interacts with the system as a whole; this class will have to be developed at a later date.

4.10.3 Website Selection Interface

Due to the requirements of interacting the user needs in order to pull out and select the very specific data they are looking at in a web page, the concept here is that the user sees what appears to be a browser and possible data points are highlighted and given as an option for selection, this class will have to be developed at a later date.

4.10.4 Crawl Request Interface

Due to the requirements of taking the selections that the user makes in the Website Selection Interface and translates that into actionable command line arguments that the SWC Core needs in order to start the downloading process. It will also take the necessary database or output selection the user makes in order to handle the output arguments, this class will have to be developed at a later date.

5.0 CSCI Data

This section details the global variable data for the Semantic Web Crawler Software Program.

Name	Description	Data Type	Purpose	Source
Page Vector	Lines of HTML	String	To be parsed through	vector<string>
Regex Vector	Regex Commands	String	Used to parse through lines	vector<string>

6.0 CSCI Data Files

This section details the use of system data files. The data file to CSC/CSU cross-reference is presented in Section 6.1. The data files and their attributes are presented in Section 6.2.

6.1 Data File to CSCI/CSU Cross-Reference

Data files to their corresponding CSU or CSC access points.

Data File	CSC	CSU
Administrator Accounts	Account Management Monitoring	All
Database	All	All
System Error Log	All	All

6.2 Data File Name

The CSCI data files consist of the database and other system files that list the data files used by the system.

Name	Description	File Type	Purpose	Source
Administrator Accounts	Master Accounts with full Access Privileges	Random Access	Separate file for System to Access Authorized System Changes	System Administrator
Database	System Database	Random Access	Stores User Information, Monitoring Records	GUI Inputs, Monitoring CSC, All CSU’s
System Error Log	Audit Log of System Errors	Sequential	Stores System Errors, Failed Login	System Software

7.0 Requirements Traceability

SRS Section	SRS Description	SDD Section	SDD Description
3.4.1	User Registration	4.1.1	User Registration
3.4.2	User Login	4.1.2	User Login
3.4.3	User Activity Selection	4.1.3	User Activity Selection
3.4.5	System Administration	4.1.4	System Administration
3.4.6	Database Management	4.1.5	Database Management
3.4.7	Initiate Web Crawler	4.1.6	Initiate Web Crawler