University of Nevada, Reno

NRDC DATA STEWARD:
Repository for Sharing NRDC Research Data and Results

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in
Computer Science and Engineering

by
Reddy Ashok Padmaraju

Dr. Sergiu Dascalu/Thesis Advisor

December 2016
THE GRADUATE SCHOOL

We recommend that the thesis
prepared under our supervision by

REDDY ASHOK PADMARAJU

entitled

NRDC Data Steward

be accepted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

Sergiu Dascalu, Ph.D., Advisor
Lei Yang, Ph.D., Committee Member
Sage R. Hiibel, Ph.D., Graduate School Representative
David W. Zeh, Ph. D., Dean, Graduate School

December, 2016
ABSTRACT

Often it is difficult for the public to access real research data for information or to work with it to perform research activities. One of the obstacles that potential researchers encounter is finding the data sources to import data to begin fundamental analysis for their research. There are numerous organizations in the world that share research data, but only with some limited access to the public. Nevada Research Data Center (NRDC) is one such organization that holds and records research data every day. The research data and results are unstructured data which can be in the form of huge data sets, videos, photographs, etc. Providing larger access to NRDC data will help more people gain knowledge and possibly perform academic research activities. One of the solutions for enabling online access to the public is to build a repository to save and share NRDC research data and results. The proposed NRDC Data Steward is a web application that provides access to the repository which allows the public to view, download, and share the research data which can be used for performing academic research activities and knowledge sharing. A public user can also contribute to the application by adding/uploading research information that will contribute to the research data and its results, which in turn would help other users. This thesis presents the software specification, design, and prototype implementation of the NRDC Data Steward web-based application. The thesis also provides background on related data forms, describes NRDC Data Steward development tools, presents feedback on the application obtained from several software developers who participated in a testing session, and suggests directions for future work.

Keywords: data steward, NRDC, web application, data repository, unstructured data, data center, document sharing, research datasets
ACKNOWLEDGMENT

At this juncture, I feel deeply honored in expressing my sincere thanks to my advisor, Dr. Sergiu Dascalu, for the confidence he bestowed on me, corrected me and provided valuable insights leading to the successful completion of my research.

I would also like to thank my thesis committee members: Dr. Lei Yang and Dr. Sage Hiibel for their valuable inputs and continuous support in making my thesis a success.

I am grateful to several of my colleagues for their passionate participation during the integration testing and providing valuable comments on this research project, in particular, Nisha Sridharan, Manash Saha, Rui Wu, Jose Painumkal and Angela Chan.

Finally, I must express my gratitude to my family members for their unconditional love and incredible support throughout my years of study and to my friends for their encouragement during my research and thesis writing.

The material presented in this thesis is based upon work supported in part by The National Science Foundation under grant number IIA-1301726. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.
TABLE OF CONTENTS

ABSTRACT ....................................................................................................................... i
ACKNOWLEDGMENT ........................................................................................................ ii
TABLE OF CONTENTS ...................................................................................................... iii
LIST OF TABLES ............................................................................................................... v
LIST OF FIGURES ........................................................................................................... vi
LIST OF ACRONYMS ....................................................................................................... vii

1 INTRODUCTION ........................................................................................................... 1

2 BACKGROUND ON DATA FORMS ............................................................................ 4
  2.1 WHAT ARE DIFFERENT DATA FORMS? ................................................................. 4
      2.1.1 FULLY STRUCTURED DATA .......................................................................... 4
      2.1.2 SEMI-STRUCTURED DATA ........................................................................... 5
      2.1.3 UNSTRUCTURED DATA ............................................................................... 6
  2.2 COMPARISON OF DATA FORMS ......................................................................... 8

3 TOOLS FOR NRDC DATA STEWARD .................................................................. 10
  3.1 DEVELOPMENT ENVIRONMENT .......................................................................... 10
  3.2 FRONT-END DEVELOPMENT ............................................................................. 11
  3.3 BACK-END DEVELOPMENT .............................................................................. 13
      3.3.1 APPLICATION DEVELOPMENT ................................................................. 14
      3.3.2 RDBMS AND NOSQL DATABASE .............................................................. 15
  3.4 DRIVERS AND PACKAGES .............................................................................. 16

4 SPECIFICATION AND REQUIREMENTS .......................................................... 21
  4.1 FUNCTIONAL REQUIREMENTS .......................................................................... 21
  4.2 NON-FUNCTIONAL REQUIREMENTS .................................................................. 23
  4.3 USE-CASE MODELING ........................................................................................ 25
      4.3.1 USE-CASE DIAGRAM ............................................................................... 25
      4.3.2 DETAILED USE CASES ............................................................................ 32
  4.4 TRACEABILITY MATRIX ..................................................................................... 42
  4.5 GLOSSARY OF TERMS ....................................................................................... 43

5 SOFTWARE MODELING ........................................................................................ 44
  5.1 SYNOPSIS DIAGRAM ........................................................................................ 44
  5.2 SYSTEM-LEVEL DESIGN .................................................................................. 45
  5.3 DATA DESIGN ..................................................................................................... 48
5.4 DETAILED DESIGN .......................................................................................................................... 51
  5.4.1 SYSTEM ACTIVITY DIAGRAM ................................................................................................. 51
  5.4.2 PSEUDO-CODE ......................................................................................................................... 53

5.5 IMPLEMENTATION & WALK-THROUGH OF NRDC DATA STEWARD ........................................ 55
  5.5.1 HOME PAGE ............................................................................................................................ 56
  5.5.2 UPLOAD PAGE ......................................................................................................................... 56
  5.5.3 SEARCH PAGE ........................................................................................................................ 57
  5.5.4 CREATE GROUP PAGE ............................................................................................................. 58
  5.5.5 ADD USERS PAGE .................................................................................................................. 59
  5.5.6 ADMINISTRATOR PAGE .......................................................................................................... 60
  5.5.7 CONTACT PAGE .................................................................................................................... 63
  5.5.8 REGISTRATION PAGE ............................................................................................................. 63
  5.5.9 LOGIN PAGE ............................................................................................................................ 64

6 SOFTWARE TESTING QUESTIONNAIRE ......................................................................................... 66

7 CONCLUSIONS AND FUTURE WORK ......................................................................................... 70

REFERENCES ........................................................................................................................................ 73
LIST OF TABLES

Table 2.1: Comparison of unstructured, fully structured, and semi-structured data [5]............. 9
Table 4.1: Functional Requirements ................................................................. 22
Table 4.2: Non-functional Requirements .................................................................. 24
Table 4.3: Detailed Use Case for UserRegistration ....................................................... 32
Table 4.4: Detailed Use Case for VerifyEmail ............................................................ 33
Table 4.5: Detailed Use Case for ForgotPassword ......................................................... 34
Table 4.6: Detailed Use Case for SendTempPassword .................................................... 34
Table 4.7: Detailed Use Case for ResetPassword .......................................................... 35
Table 4.8: Detailed Use Case for Login .................................................................... 35
Table 4.9: Detailed Use Case for UploadData ............................................................. 36
Table 4.10: Detailed Use Case for GroupAccess ......................................................... 36
Table 4.11: Detailed Use Case for SearchData ............................................................ 37
Table 4.12: Detailed Use Case for DownloadData ......................................................... 37
Table 4.13: Detailed Use Case for ViewData ............................................................... 38
Table 4.14: Detailed Use Case for CreateGroup ......................................................... 38
Table 4.15: Detailed Use Case for AddUsers ............................................................... 39
Table 4.16: Detailed Use Case for DeleteUser ............................................................. 39
Table 4.17: Detailed Use Case for DeleteGroup ......................................................... 40
Table 4.18: Detailed Use Case for AdminAccess ......................................................... 40
Table 4.19: Detailed Use Case for Contact ........................................................-------- 41
Table 4.20: Detailed Use Case for Logout ................................................................. 41
Table 4.21: Requirements Traceability Matrix ............................................................. 42
LIST OF FIGURES

Figure 2.1: Sample table in a relational database system ...............................................................5
Figure 2.2: Sample XML code with tag ..........................................................................................6
Figure 2.3: Sample collection in a MongoDB (NoSQL) database ....................................................7
Figure 2.4: Data growth rate for structured and unstructured data. (Source: Edureca.co) .............8
Figure 4.1: Use-case diagram of NRDC Data Steward .................................................................26
Figure 5.1: Synopsis model of NRDC Data Steward ......................................................................44
Figure 5.2: High-level system design ............................................................................................46
Figure 5.3: System-activity diagram ..............................................................................................52
Figure 5.4: Home page ................................................................................................................56
Figure 5.5: Upload page ..............................................................................................................57
Figure 5.6: Search results page ....................................................................................................58
Figure 5.7: Create group page .....................................................................................................59
Figure 5.8: Add users page .........................................................................................................60
Figure 5.9: Delete group page .....................................................................................................61
Figure 5.10: Delete user page .....................................................................................................62
Figure 5.11: Admin access page .................................................................................................62
Figure 5.12: Contact page .........................................................................................................63
Figure 5.13: Registration page ....................................................................................................64
Figure 5.14: Login page .............................................................................................................65
Figure 6.1: Software testing questionnaire .....................................................................................66
# LIST OF ACRONYMS

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>NRDC</td>
<td>Nevada Research Data Center</td>
</tr>
<tr>
<td>HTML</td>
<td>Hyper Text Markup Language</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated Development Environment</td>
</tr>
<tr>
<td>HTTPS</td>
<td>Hyper Text Transfer Protocol Secure</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>CSS</td>
<td>Cascading Style Sheets</td>
</tr>
<tr>
<td>MongoDB</td>
<td>Mongo Database</td>
</tr>
<tr>
<td>NoSQL</td>
<td>Not Only Structure Query Language</td>
</tr>
<tr>
<td>MySQL</td>
<td>My Structured Query Language</td>
</tr>
<tr>
<td>BSON</td>
<td>Binary JSON</td>
</tr>
<tr>
<td>JSON</td>
<td>JavaScript Object Notation</td>
</tr>
<tr>
<td>BSD</td>
<td>Berkeley Software Distribution</td>
</tr>
<tr>
<td>CSRF</td>
<td>Cross-Site Request Forgery</td>
</tr>
</tbody>
</table>
1 INTRODUCTION

The Nevada Research Data Center (NRDC) [1] Data Steward, proposed in this thesis, is a web application that provides access to research data and results to the public. This web application would be helpful in data sharing and performing academic research. The public users can view or download the data from the NRDC Data Steward website and can also upload their research data and results to the website.

The main functionalities of this application include viewing, downloading, and uploading the research data and results online. Group administrators can give specific permissions to the public or a group of people for accessing the files in the data steward. The NRDC Data Steward application has three levels of access given to the user to access data from the online system:

1. Public access: research data and results are open for public users to search, view, and download.
2. Restricted access: research data and results are available only to the users who are provided with access by the admin or the author of the data.
3. Admin access: users with administrative access can view, download, and delete the data that are not compliant with the NRDC. They also have the authority to revoke membership of users and give administrative authorities to other users.
The front end of the NRDC Data Steward application is designed using HTML, CSS, and JavaScript. The back end of the application is coded in Python/Flask. The application also uses a NoSQL server database (MongoDB) and MySQL for storing and retrieving information. The application is coded using Microsoft Visual Studio 2015 on a Microsoft Windows platform.

Any user can upload the research data, search, discover, and download the data that is available to the public. The intended users of this web application are students, academic faculty, scientists, and anyone is interested in acquiring knowledge related to the NRDC research activities. The NRDC Data Steward web application benefits the users in acquiring the research knowledge that might be hard to access by the general audience.

The data uploaded or downloaded from the NRDC Data Steward application is not a structured data. The unstructured data takes many forms, such as research datasets, conference papers, presentation slides, video, pictures, group of pictures, surveys, questionnaires etc. Managing the unstructured data in different forms is a challenge. Unlike traditional databases that can handle structured data, implementing and management of unstructured data is more difficult.

Google search engine is one of the best products available that can search and manage unstructured data. This search engine is helpful in viewing and downloading data based on the search keywords entered by the user. It produces vast results that include research documents, pictures, videos, and much more. Yahoo search engine and Bing are other examples of search engines that can handle unstructured data. All these major search engines use MapReduce
algorithms for retrieving results faster. MapReduce is one of the features that is provided by big data.

The NRDC Data Steward application provides access to most of the data and results related to the research activities that can be accessed by the public. Making research data accessible to the public is one of the innovative steps that can help the public to acquire knowledge about research activities.

In the remainder of this thesis, the chapters are arranged as follows: Chapter 2 offers background of different data forms available; Chapter 3 presents an overview of front-end, back-end tools and the drivers and packages needed to develop the NRDC Data Steward web application; Chapter 4 provides the specification and requirements of the this application; Chapter 5 focuses on the detailed design and implementation of the NRDC Data Steward web application; Chapter 6 summarizes the feedback given by the software developers in response to the software testing questionnaire; and Chapter 7 concludes the thesis with final remarks and outlines possible future directions of work to the application.
2 BACKGROUND ON DATA FORMS

Nevada Research Data Center (NRDC) currently holds a large number of datasets and research data in its data center [1]. Some of the data residing in servers of the research data center is currently not accessible to the public. The data stored in NRDC servers is unstructured data and this chapter discusses different data forms, how they are accessed, and also provides a comparison of various properties of these data forms.

2.1 WHAT ARE DIFFERENT DATA FORMS?

Any data that can be stored or processed essentially falls in one of the three data forms [2], as below, based on its properties (discussed later):

- Fully Structured Data
- Semi-Structured Data
- Unstructured Data

2.1.1 FULLY STRUCTURED DATA

Fully structured data refers to any data that has a fixed field in a file or record. Such well- formatted data is stored in the form of tables with rows and columns, spreadsheets etc. One of the
examples of such fully structured data is used in Relational Database Management Systems (RDBMS). In RDBMS, the data is stored in tables. In Figure 2.1, a sample table is retrieved from a relational database system. The fundamental structure in RDBMS is easily understandable since the design of RDBMS is well defined. Fully structured data is defined as “An instance of such a schema is some data that conforms to this specification” [2].

![Sample table in a relational database system](image)

Structured Query Language (SQL) is the standard data access language. However, SQL is limited in the real world, especially when working with complex data. Prior knowledge of the structure in the relational database is required to create and execute ad hoc queries.

2.1.2 SEMI-STRUCTURED DATA

Semi-structured data contains tags or other markers to separate semantic elements to enforce hierarchies of records and fields within data. Semi-structured data does not have a formal structure to the data models.
EXtensible Markup Language (XML), and other markup languages, emails, etc., are all forms of semi-structured data. Figure 2.2 presents a sample XML code with tags showing the hierarchical structure of data. Semi-structured data formats support hierarchical data that simplifies data models representing complex relationships between entities and support for listing of objects, avoiding messy translations of lists into a relational data model.

```xml
<?xml version="1.0" encoding="utf-8"?>
<beacons>
  <realtime host="t.hulu.com">
    <beacon type="error" send="always" cdn-specific="true">
      <event name="applicationerror.appname" send="never" />
      <event name="connectionerror.loadtimeout" send="never" />
    </beacon>
    <beacon type="session" send="always" cdn-specific="true" />
  </realtime>
  <beacon type="playback" send="never" cdn-specific="true">
    <event name="start" send="always" />
  </beacon>
</beacons>
```

*Figure 2.2: Sample XML code with tag*

### 2.1.3 UNSTRUCTURED DATA

As per the technology giant IBM, the basic understanding of unstructured data is that it is any data that is not managed by any standard database management system (DBMS) [3]. It refers to information that is not in the pre-organized format or doesn’t have a pre-defined data model. The information may contain text, dates, numbers, symbols, mathematical expressions, etc. Unstructured data, without having any pre-defined data model, results into a need for new ways to
process information. Data with some form of structure to it may still be recognized as unstructured data if the structure is not helpful in the processing task at hand. Modern techniques such as data mining, text analytics, and noise-text analytics provide different methods to find patterns in, or otherwise interpret the information. The list of unstructured data formats includes, but is not limited to text documents, images, videos, program codes, eBooks, analog data, with extensions .doc, .pdf, .jpeg, .png, .pptx, .xls, .bmp etc.

Collection in NoSQL database is equivalent to the table in Relational database. A collection is a group of several documents that are often used for similar purpose. Each document in a collection consists of key-value pairs. A collection can hold different types of data and does not need to have any structure. Documents in NoSQL are like tuple or rows in relational database systems. Figure 2.3 is an example of unstructured data from collection ‘zips’ with several documents. Each document has a comma separating the key-value pair.

Figure 2.3: Sample collection in a MongoDB (NoSQL) Database
Figure 2.4 shows the growth rate of structured and unstructured data over a period of ten years from 2005 to 2015. It is estimated by International Data Corporation (IDC) that by 2020 the number will reach 40,000 EB, or 40 Zettabytes (ZB) [4]. According to Lucas Mearian of the Computerworld, “By 2020, there will be 5,200 GB of data for every person on Earth” [29]. So, the need for managing and processing unstructured data is growing. This leads to developing new software applications, databases, and file systems to manage unstructured data effectively.

![Figure 2.4: Data growth rate for structured and unstructured data. (Source: Edureca.co)](image)

### 2.2 COMPARISON OF DATA FORMS

A comparison of the three data Types - Unstructured, Fully Structured, and Semi-Structured data is shown in Table 2.1 [5]. Unstructured data consists of 80% of the total data available on the web, including videos, photos, audio files, presentations, web pages, and other business documents that are represented in character and binary data. It is very flexible, scalable, and does not have any transaction management. It relies only on textual queries.
Fully structured data is information that can be stored in a database SQL in the form of rows and columns as relational database tables. Unlike unstructured data, there is matured transaction management and the flexibility of the information is schema-dependent. It also does not have easy scalability and is very robust. It relies on a structured query that allows complex joins.

Semi-structured data is information that does not reside in a relational database but the properties of semi-structured data make it easier to analyze. Common examples of semi-structured data are XML and Resource Description Format (RDF). The transaction management is adapted from RDBMS but it is not matured. It is like unstructured data with respect to flexibility and scalability. It relies on queries over anonymous nodes.

Table 2.1: Comparison of unstructured, fully structured, and semi-structured data [5]

<table>
<thead>
<tr>
<th></th>
<th>Unstructured</th>
<th>Fully Structured</th>
<th>Semi-Structured</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Technology</strong></td>
<td>Character and binary data</td>
<td>Relational database tables</td>
<td>XML/RDF</td>
</tr>
<tr>
<td><strong>Transaction Management</strong></td>
<td>No transaction management, no concurrency</td>
<td>Matured transaction management, various concurrency techniques not matured</td>
<td>Transaction management adapted from RDBMS, not matured</td>
</tr>
<tr>
<td><strong>Version Management</strong></td>
<td>Versioned as a whole</td>
<td>Versioning over tuples, rows, tables, etc.</td>
<td>Not very common, versioning over triples or graphs is possible</td>
</tr>
<tr>
<td><strong>Flexibility</strong></td>
<td>Very flexible, absence of schema</td>
<td>Schema-dependent, rigorous schema</td>
<td>Flexible, tolerant schema</td>
</tr>
<tr>
<td><strong>Scalability</strong></td>
<td>Very scalable</td>
<td>Scaling DB schema is difficult</td>
<td>Schema scaling is simple</td>
</tr>
<tr>
<td><strong>Robustness</strong></td>
<td>-</td>
<td>Very robust, enhancements since 30 years</td>
<td>New technology, not widely spread</td>
</tr>
<tr>
<td><strong>Query-Performance</strong></td>
<td>Only textual queries possible</td>
<td>Structured Query allows complex joins</td>
<td>Queries over anonymous nodes are possible</td>
</tr>
</tbody>
</table>
3 TOOLS FOR NRDC DATA STEWARD

NRDC Data Steward web application is developed with a combination of front-end and back-end tools in Microsoft Visual Studio IDE. Several drivers and packages are installed to develop a web application with a mix of different languages and successfully transfer data to and from databases. This chapter discusses tools and software applications that are used to design and develop NRDC Data Steward Web application.

3.1 DEVELOPMENT ENVIRONMENT

Visual Studio IDE from Microsoft is a software development platform that supports many built-in tools that are helpful in developing windows applications, websites, web applications, and web services. Visual Studio uses WinAPI (Windows Application Programming Interface), Windows Forms, Windows Presentation Foundation, Windows Store, and Microsoft Silverlight all of which help to produce both native and managed codes [6]. Features of Visual Studio include Code Editor with Microsoft’s IntelliSense, Debugger, Designer, and Extensibility in the form of macros, add-ins, and packages. Visual Studio supports coding in different languages that are either built-in such as ASP.NET C#, ASP.NET Visual Basic, and C/C++ or those available via language services installed for Python, Ruby, Node.js, etc. Code Editor in Visual Studio supports programming in these languages and includes Microsoft’s IntelliSense for code completion.
3.2 FRONT-END DEVELOPMENT

The front-end development is defined as the process of building client-side applications that are viewed in the end user's web browser and other devices [7]. In any web application, the user will first see and interact with the front-end that makes it essential for the developers to keep in mind the aesthetics as well as the performance of the application. In recent years, end users have come to expect web pages to load in 3 seconds or less [8]. Therefore, performance and speed are also considered to be important features along with aesthetics of the web page while developing web applications. The involvement of users and understanding of task requirements are a few characteristics of a human-centered design [9].

The front-end of NRDC Data Steward web application is designed using the following languages-

- Hyper Text Markup Language (HTML)
- Cascading Style Sheets (CSS)
- JavaScript

HTML is developed by World Wide Web Consortium (W3C) and Web Hypertext Application Technology Working Group (WHATWG) [10]. It is considered as the skeleton that provides a structure to every web page on the internet and is the standard language for building web pages and web applications. HTML defines the structure of a web page that a web browser can understand and display as multimedia web pages on the user’s screen. HTML follows international formatting and it is a simplified version of Standard Generalized Markup Language
(SGML) [11]. All the elements in HTML are represented by start and end tags. The W3C that maintains both HTML and CSS standards, encourages the use of CSS for describing presentation in HTML web pages. The W3C has deprecated many elements in HTML that may become obsolete by newer constructs [12], but at the same time it suggested the user agents to continue the support for these deprecated elements for backward compatibility until they become obsolete.

Cascading Style Sheets (CSS) is style sheet language used to describe presentation in HTML web pages. The specifications of Cascading Style Sheets are maintained by the W3C that provides a free CSS validation service for CSS documents by applying current standards. CSS is useful to describe HTML elements that are to be displayed on user’s screen. There are three ways to include style sheets to any web page:

- Internal Stylesheet: This holds CSS code in the head section of a webpage and makes it easier to apply coded styles within the web page.
- External Stylesheet: It is a .css file that is coded in a different file and then linked to multiple web pages. Therefore, the layout of multiple web pages can be controlled by manipulating CSS files.
- Inline Styles: To apply a style to a specific tag in HTML, the inline “style” attribute can be used. Inline Styles are particularly useful if the change done is not used repeatedly through the site.

A CSS syntax consists of a selector and a declaration block. The selector points to the patterns that are used to select the HTML elements that need styling and the declaration block contains one or more CSS property names and values.
JavaScript is a high-level, dynamic programming language. It helps to produce a dynamic and responsive web page along with CSS and HTML. It has an API that makes working with texts, arrays, and dates efficient, but does not include networking, storage, or graphic facilities. Unlike Java, JavaScript relies on a runtime environment, such as a web browser, to provide an ability to import or include HTML scripts.

### 3.3 Back-End Development

The back-end of a website consists of three parts: an application, a server, and a database. These three features communicate with each other to power the front-end of the website. The back-end of a web application is important for database interactions, business logic, and overall performance of the application. Problems in back-end can significantly damage the performance of the web application by constantly throwing errors and crashing the application frequently. Programming languages such as Python, Java, ASP.NET, etc., are used to build any server-side application. Server-side programming languages help the application, server, and database to communicate with each other. The backend of NRDC Data Steward web application is developed using the following:

- Server-side application – Python
- Database
  - Relational Database – MySQL
  - NoSQL Database – MongoDB
3.3.1 APPLICATION DEVELOPMENT

The back-end application development for NRDC Data Steward is accomplished using Python programming language. Python is a high-level, object-oriented, and dynamic programming language with the design philosophy emphasizing the code readability [13]. The popularity of the Python programming language lies in the simplicity of the code that makes it easy to read, learn and remain a powerful language. Python fully supports both structured and object-oriented programming and features from other languages such as functional programming and aspect-oriented programming. The free and open source Python software is managed by the non-profit Python Software Foundation. Major implementations of Python are CPython, IronPython, Jython, MicroPython and PyPy.

The development of Python programming language is conducted through the Python Enhancement Proposal (PEP) process that collects inputs on issues from the community and proposes new enhancements to Python [14]. The design principles for Python, also known as Zen of Python [15] is a collection of 20 guidelines influencing the design of Python language, and some of them are listed below [15]:

- “Beautiful is better than ugly”
- “Explicit is better than implicit”
- “Simple is better than complex”
- “Complex is better than complicated”
- “Flat is better than nested”
- “Sparse is better than dense”
3.3.2 RDBMS AND NOSQL DATABASE

The back-end of NRDC Data Steward web application communicates with a Relational Database Management System designed using MySQL and an Unstructured NoSQL database developed using MongoDB. This web application stores and retrieves user information from My Structured Query Language (MySQL) containing a fixed number of columns. The files and datasets that are uploaded or downloaded to/from NRDC Data Steward are stored in MongoDB.

MySQL

MySQL is a relational database management system. It is an open-source database application that has made its source available to the public under GNU General Public License. The creator of MySQL database was MySQL AB, but now it is owned by Oracle Corporation. Though MySQL is open source and available for free to the public, some of the functionalities are offered as paid editions under proprietary agreements. As per DB-Engines rankings, MySQL stands as the second most widely used relational database system behind Oracle Database [16]. It was implemented in C and C++ language and the server supports operating systems such as FreeBSD, Linux, Windows, Solaris, and OS X. The data in relational data management systems are stored as structured tables in the form of rows and columns. By the definition of RDBMS, all its products implemented presents data as tabular form. Structured Query Language (SQL) is the query language used for storing and retrieving data to/from the database. RDBMS has a rich functionality but is not known for being particularly scalable. It scales up very well by adding more hardware to get higher performance from a single node, but it is not particularly good at scaling out and using commodity hardware.
MongoDB

MongoDB is a NoSQL document-oriented database. This free and open-source cross-platform NoSQL database, was developed by MongoDB Inc and released an initial version in 2009. It is written in C, C++, and JavaScript and supports operating systems such as Windows, Linux, OS X, Solaris, and FreeBSD. In the MongoDB data model, the data stored in collections are presented in JavaScript Object Notation (JSON). One of the main reasons for using JSON is that development teams can design data models to support common data access patterns. MongoDB database models are not predicated on joining data from multiple tables together, it is much easier to distribute or shard data across multiple servers. This results in a wide variety of options when deploying a database from many inexpensive commodity machines to a larger and more powerful server. Since joins and multi-table transactions are difficult for processing in parallel, scaling up is the best option for a relational system that would increase hardware expenses to serve from a single system. The schema design capabilities in MongoDB supports models requiring atomic reads and writes only to individual documents. MongoDB is designed to support agile software engineering practices.

3.4 DRIVERS AND PACKAGES

A driver or connector is a program that is used to bridge a standard Application Programming Interface (API) with multiple database types that use different protocols, data representations, and communication patterns. It is the standard means of connecting to a database within programming environments and applications. A distribution package, that is commonly referred to as “package” or “distribution”, is a versioned archive file that contains different Python
packages, modules, and other resource files that are used to distribute a release. Drivers and packages needed to be installed for successful design and development of NRDC Data Steward web application.

**MySQL Connector/Python:**

“MySQL Connector/Python” is the driver or connector that is helpful to communicate with MySQL database servers from Python programs. For creating a connection to the database server, the “connect()” constructor should be used. Below is the example showing how to create a connection to MySQL database server:

```python
import MySQLdb
db = MySQLdb.connect("localhost", "root", "testpassword", "nrdc")
db.close()
```

The connection arguments, as shown above for Python MySQLdb.connect, includes host name, user name, password, and database respectively. This creates a connection to MySQL database server running on localhost. A valid username and password combination is needed to connect to the server and to a database in the server.

**PyMongo:**

PyMongo is the MongoDB recommended driver to work with MongoDB from Python programming language. PyMongo supports different versions of CPython 2.6, 2.7, 3.3+, PyPy, and PyPy3 [17]. PyMongo can be installed on all platforms using “pip”, “easy_install”.
python -m pip install pymongo
python -m easy-install pymongo

Once the distribution of PyMongo is installed, the driver is ready to connect to the MongoDB server from Python. If an instance of MongoDB is not already running on the client system, it can be started by simply executing a “mongod” command in command prompt. A connection to a MongoDB instance running on the default host and port (‘localhost’, 27017) can be created using “MongoClient()”.

```python
import pymongo
connection = pymongo.MongoClient()
db = connection.nrdc
```

This code is to create a connection to the MongoDB database server in Python language. After creating a connection to the instance of MongoDB running on the client system, the program connects to the “nrdc” database in MongoDB where the uploading, downloading of documents to/from MongoDB is done.

**Jinja2:**

Jinja2 is a designer-friendly template engine with inheritance and bytecode compilation. Jinja2 was written in Python and is licensed under Berkeley Software Distribution (BSD). Some of the features of Jinja2 include sandboxed execution, powerful automatic HTML escaping system for XSS prevention, template inheritance, configurable syntax, and template designer helpers [18]. Template inheritance is a powerful part of Jinja2 that allows the user to create a base template that
consists of all the common elements, such as website and blocks, and can be used to override the child template. Similarly, the contents of the parent block can be rendered by calling “super” in the child template.

**Pip:**

Pip is a Python recommended tool managing Python packages in the client system. It can be used to perform operations like installing and uninstalling Python packages. Pip is an alternative for easy_install that implements the idea of requirement files, giving power to the user to install packages. Pip supports installing from wheel that is a Build Distribution introduced in PEP 427 [19]. Installing, upgrading, uninstalling packages is done by executing simple commands as shown below:

```
 pip install packagename
 pip uninstall packagename
```

**Flask:**

Flask is a popular extensible web framework for developing web applications in Python. It is a lightweight web framework written in Python language and based on the werkzeug toolkit and Jinja2 template engine. It supports cross-platform and is licensed under BSD. Many application features, and extensions can be added to Flask, such as object-relational mapper, form validation, file upload handling, authentication technologies for security, and much more. Flask web framework consists of features like support for unit testing, templating via Jinja2, session management using cookies, extensive documentation, and Google App Engine compatibility.
**Flask-WTF:**

Form validation in Flask is available with the help of the Flask-WTF extension. It is a simple integration of Flask and WTForms including a secure form with a Cross-Site Request Forgery (CSRF) token, file upload and reCAPTCHA [20]. WTForms contain field definitions, delegate validation, aggregate errors, and in general function as a glue holding everything together [21]. A subclass of Form defines the fields declaratively as class attributes. The in-line validators for these files in the form provide custom validation without any need to write a one-time use validator by defining a method with convention “validate_fieldname”.

**Flask-Mail:**

Flask-Mail is an extension of Python flask that provides an interface to send messages from views and scripts by setting up SMTP. Flask-Mail can be installed using Pip from the command line. Parameters for Flask-Mail including: MAIL_SERVER, MAIL_PORT, MAIL_USE_SSL, MAIL_USERNAME, MAIL_PASSWORD, should be configured in the web application for using email functionality. After configuring the settings for Flask-Mail, an instance of Mail class should be created. Methods in Mail class include: send() – to send contents of message class object; connect() – to open a new connection to mail host; send_message() – to send message object. Methods in Message class include: attach() – to send an attachment through mail; and add_recipient() – to add a new recipient to the mail.
4 SPECIFICATION AND REQUIREMENTS

Requirements analysis is the first step in the Software Development Life Cycle [22]. In this step, high-level requirements and project goals are described based on the initial analysis. The members in the project should obtain a thorough understanding of what the system needs to provide for the user [23]. This chapter discusses the functional requirements, non-functional requirements, and use-case modeling of the NRDC Data Steward web application.

4.1 FUNCTIONAL REQUIREMENTS

Functional requirements describe what an application or system should do by outlining the behavior of the system’s functionality. These are high-level functionalities designed to achieve project goals. Typically, functional requirements may include calculations, technical details, data manipulation, data processing, business rules, administrative functions, certification requirements, etc. [24].

Functional requirements for the NRDC Data Steward web application are listed below in Table 4.1. These requirements are labeled as ‘1’, ‘2’, or ‘3’. Requirements that are labeled as ‘1’ have already been implemented in the NRDC Data Steward web application. Requirements that are labeled as ‘2’ may be implemented for better functionality in the NRDC Data Steward web application. Requirements that are labeled as ‘3’ will be implemented as a part of future upgrades.
**Table 4.1: Functional Requirements**

<table>
<thead>
<tr>
<th>R01</th>
<th>NRDC Data Steward shall allow users to register in the system.</th>
</tr>
</thead>
<tbody>
<tr>
<td>R02</td>
<td>NRDC Data Steward shall require valid username and password combination to login.</td>
</tr>
<tr>
<td>R03</td>
<td>NRDC Data Steward shall allow users to verify registration by entering random code sent to their email address.</td>
</tr>
<tr>
<td>R04</td>
<td>NRDC Data Steward shall allow users to receive a temporary password to registered email in case they forgot the password.</td>
</tr>
<tr>
<td>R05</td>
<td>NRDC Data Steward shall allow users to create a new password using temporary login.</td>
</tr>
<tr>
<td>R06</td>
<td>NRDC Data Steward shall allow users to upload data with public access without signing into the system.</td>
</tr>
<tr>
<td>R07</td>
<td>NRDC Data Steward shall allow users to search research data available to public without signing into the system.</td>
</tr>
<tr>
<td>R08</td>
<td>NRDC Data Steward shall allow users to download research data available to public without signing into the system.</td>
</tr>
<tr>
<td>R09</td>
<td>NRDC Data Steward shall allow users to view research data available to public on a client browser without signing into the system.</td>
</tr>
<tr>
<td>R10</td>
<td>NRDC Data Steward shall allow users to contact the administrator in case of any questions.</td>
</tr>
<tr>
<td>R11</td>
<td>NRDC Data Steward shall allow authorized users to upload data with restricted access to groups.</td>
</tr>
<tr>
<td>R12</td>
<td>NRDC Data Steward shall allow authorized users to search research data available to them.</td>
</tr>
<tr>
<td>R13</td>
<td>NRDC Data Steward shall allow authorized users to download research data available to them.</td>
</tr>
<tr>
<td>R14</td>
<td>NRDC Data Steward shall allow authorized users to view research data available to them on client web browser.</td>
</tr>
<tr>
<td>R15</td>
<td>NRDC Data Steward shall allow authorized users to create groups.</td>
</tr>
<tr>
<td>R16</td>
<td>NRDC Data Steward shall allow authorized users to add other registered users to groups they created.</td>
</tr>
<tr>
<td>R17</td>
<td>NRDC Data Steward shall allow the administrator to remove users registered in the system.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>R18</td>
<td>[1]</td>
</tr>
<tr>
<td>R19</td>
<td>[1]</td>
</tr>
<tr>
<td>R20</td>
<td>[2]</td>
</tr>
<tr>
<td>R21</td>
<td>[2]</td>
</tr>
<tr>
<td>R22</td>
<td>[2]</td>
</tr>
<tr>
<td>R23</td>
<td>[2]</td>
</tr>
<tr>
<td>R24</td>
<td>[3]</td>
</tr>
<tr>
<td>R25</td>
<td>[3]</td>
</tr>
<tr>
<td>R26</td>
<td>[3]</td>
</tr>
<tr>
<td>R27</td>
<td>[3]</td>
</tr>
<tr>
<td>R28</td>
<td>[3]</td>
</tr>
<tr>
<td>R29</td>
<td>[3]</td>
</tr>
</tbody>
</table>

### 4.2 NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements define an overall property of the application as a part of the system architecture. These non-functional requirements are also referred as quality attributes, quality goals, non-behavioral requirements, and constraints [25]. Some examples of non-
functional requirements include the performance of the system, scalability, capacity, availability, reliability, security, usability, and data integrity [24]. To meet project/organizational goals, non-functional requirements play an important role as failing could result in poor performance, low security, less scalability, etc. Non-functional requirements of NRDC Data Steward web application are listed in Table 4.2.

Table 4.1: Non-functional Requirements

<table>
<thead>
<tr>
<th>No</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>N01</td>
<td>NRDC Data Steward shall be a cross-platform web application developed in Python Flask.</td>
</tr>
<tr>
<td>N02</td>
<td>NRDC Data Steward shall maintain simple user interface.</td>
</tr>
<tr>
<td>N03</td>
<td>NRDC Data Steward shall store uploaded data in MongoDB collection.</td>
</tr>
<tr>
<td>N04</td>
<td>NRDC Data Steward shall store metadata of the uploaded file in MongoDB collection.</td>
</tr>
<tr>
<td>N05</td>
<td>NRDC Data Steward shall allow the user to upload a maximum of 16MB of data.</td>
</tr>
<tr>
<td>N06</td>
<td>NRDC Data Steward shall encrypt the password before updating in the database.</td>
</tr>
<tr>
<td>N07</td>
<td>NRDC Data Steward shall store registered user information in MySQL database.</td>
</tr>
<tr>
<td>N08</td>
<td>NRDC Data Steward shall store groups information in MongoDB database.</td>
</tr>
<tr>
<td>N09</td>
<td>NRDC Data Steward shall be implemented in Python Flask web framework.</td>
</tr>
<tr>
<td>N10</td>
<td>NRDC Data Steward shall download data from MongoDB collection.</td>
</tr>
<tr>
<td>N11</td>
<td>NRDC Data Steward shall upload unstructured data type such as pdf, txt, doc, jpg, png, mp4, mp3, ppt, etc.</td>
</tr>
</tbody>
</table>
4.3 USE-CASE MODELING

Modeling use cases is one of the early steps in Unified Object Modeling approach. It is an important and valuable technique in the ‘requirement analysis’ step in the software development life cycle. Use-case models typically describe how different types of users interact with the system. Each use case must describe how the system should react under given condition by successfully achieving the goal of that functionality set by stakeholders. The use-case models must contain basic model elements such as actors, use-case and associations and advanced use case models consists of elements such as subject, use-case package, generalizations, and dependencies [26].

**Actors**: An actor in use-case modeling can be a human, a machine, an organization, or a software that interacts with the system.

**Use Case**: Each use-case in the model represents a sequence of actions that an actor in the application performs within the system to achieve a goal [27].

**Association**: An association in use-case model describes the relationships between an actor and the use case that is associated with it.

4.3.1 USE-CASE DIAGRAM

The use-case diagram of the NRDC Data Steward web application is shown in Figure 4.1. It shows the users of the system, or actors, various use cases, and association or relationship between an actor and the use case. There are three different actors for NRDC Data Steward web
application: Public user, Registered user, and Administrator. Each actor is associated with a set of use-cases or functionalities in the system that are available exclusively for them. The use-cases of the web application are shown in oval boxes and all the associations between the actors and the use cases are indicated by solid lines.

Figure 4.1: Use-case diagram of NRDC Data Steward
UC01: UserRegistration

The user must initially register to NRDC Data Steward web application to access restricted files and for additional functionalities. The user must enter their personal information, such as a unique username, full name, valid email address, and password in the registration page to access authorized user functionalities.

UC02: VerifyEmail

A verification email with the verification code will be sent to the email address that the user entered on the registration page. The user must verify the email address after their first login by typing the verification code when requested. Once the verification code is validated, the user will gain access to authorized user functionalities in NRDC Data Steward web application.

UC03: ForgotPassword

The user can use this functionality in case they forget the password that they used while registering to NRDC Data Steward web application. The user must enter a valid username and email combination that the user had used during registration.

UC04: SendTempPassword

A temporary password will be sent to the registered email address if the user forgets their password. This temporary password will allow the user to login to NRDC Data Steward web application to create a new password.
**UC05: ResetPassword**

NRDC Data Steward web application allows the user to reset their password once the user logs into the application using the temporary password. The user must create a new password and confirm it in the ResetPassword page. The new password created by the user will be their permanent password to login into NRDC Data Steward web application.

**UC06: Login**

Users who have registered to NRDC Data Steward web application and whose email has already been verified can use the registered username and password combination to login into the NRDC Data Steward web application. After login, the users will be redirected to the home page where they have access to various functionalities in the web application.

**UC07: UploadData**

Any user can upload a set of research data and results to the NRDC Data Steward web application using this functionality. The user needs to submit a form with information that includes their full name, email address, author name, the title of the document, description of the document, multiple tags, and the attached file. After the user submits the form, the application will upload the attached file along with the metadata information of the file to the MongoDB server where all the uploaded documents are saved.
UC08: GroupAccess

Group access to the uploaded files can be given by the registered user or administrator while uploading the files to NRDC Data Steward. If the user uploads files to the web application without logging in, then a “public” access will be given to the files by default. If the user wishes to upload a file with limited access to certain registered groups or users in the NRDC Data Steward web application, then they must enter valid group names while uploading the documents.

UC09: SearchData

NRDC Data Steward web application allows the user to search for research data and results in the application database. This application will search for matching keywords in multiple database fields. The user can search for information in the application by entering keywords for author name, the title of the document, description of the document, uploader’s name, file name and tags of the data. The results for the search will be displayed in the search page along with a link to view or download the file.

UC10: DownloadData

The results displayed in the search page will have a clickable link to download data. Different file formats that are not supported to be viewed on the webpage will be automatically downloaded to the client system provided the users have the recommended software to view the file once it has been downloaded successfully.
UC11: ViewData

While using NRDC Data Steward web application, users can view any research data and result in client browser with supporting add-ins. File formats such as jpg, pdf, mp4, png, html etc can be viewed on most of the browsers if appropriate add-ins are installed. After viewing the file in the browser, the users can save the document to their personal computers by right clicking on the file that they are currently viewing.

UC12: CreateGroup

The registered user in NRDC Data Steward web application has the access to create group functionality. The user who created the group will automatically be the group administrator. Only group administrators have access to add other registered users to the group they created. If the user decides to give access to a limited number of registered users, they should create a group and fill the group name field in the form in upload page.

UC13: AddUsers

The group administrators can use this functionality to add other registered users to the group they created. This ensures integrity by not allowing users other than group administrators to add users. Multiple users can be added to the group by entering their usernames as comma-separated values. The users in the group will have access to view/download research data and results that are authorized to them by the uploader.
**UC14: DeleteUser**

Administrators of NRDC Data Steward web application can use this functionality to remove users from the database. Administrators can delete registered user if they violate any terms and conditions of NRDC Data Steward or upon request from the user. The administrator can remove multiple users from the system by simply entering usernames in comma-separated values in a delete user web page. Once the user is removed from the system, they are required to submit a fresh registration form for gaining access to authorized user functionalities. Registered users other than administrators do not have access to use this functionality.

**UC15: DeleteGroup**

Administrators of NRDC Data Steward web application use this functionality to delete any registered groups upon request from the group administrators or if they violate terms and conditions of NRDC Data Steward web application. This is an administrator functionality and the registered users do not have access to this functionality. Administrators can enter group names in comma-separated values to delete multiple groups from the database.

**UC16: AdminAccess**

Administrator access to the NRDC Data Steward web application can be given to registered user only by those who already have administrator access. This ensures integrity to the NRDC Data Steward web application by allowing only current administrators to give administrator access to other users. Once the administrator access is given to the users, they will have access to other administrator functionalities such as removing the user from the system, deleting group from the
database and adding new administrators to the system.

**UC17: Contact**

Users in NRDC Data Steward web application can contact administrators using the contact form. Once the user fills and submits the form online, the application will send a notification to the administrator for review along with the user details. This allows the administrator to respond to the user with an appropriate answer.

**UC18: Logout**

This functionality allows the user to completely sign out from the NRDC Data Steward web application. The session for this login will expire and all session variables will be nullified.

### 4.3.2 DETAILED USE CASES

This section presents the detailed use-cases of NRDC Data Steward web application. Details for each use case such as actors, preconditions, flow of events, postconditions of use case are listed.

**Table 4.2: Detailed Use Case for UserRegistration**

<table>
<thead>
<tr>
<th>Use Case ID</th>
<th>UC01</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor(s)</td>
<td>Public user</td>
</tr>
<tr>
<td>Precondition(s)</td>
<td>Public user must have a valid email address to register</td>
</tr>
</tbody>
</table>
| Flow of Events | 1. Public user must navigate to registration page  
2. Public user must choose unique username  
3. Public user must enter user information such as full name, email and password |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Postcondition(s)</td>
<td>If registration is successful, add user details in MySQL database and display “Registration successful” message. Else display appropriate error messages for review.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use Case ID</th>
<th>UC02</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor(s)</td>
<td>Public user</td>
</tr>
<tr>
<td>Precondition(s)</td>
<td>Public user must have submitted registration form with valid email address</td>
</tr>
</tbody>
</table>
| Flow of Events | 1. Public user will receive an email with passcode  
2. User will login to NRDC Data Steward web application using their login credentials used while registration  
3. Public user must enter passcode when logging in for the first time |
| Postcondition(s) | If the passcode entered by the user is correct, then navigate to NRDC Data Steward web application home page. Else display “incorrect passcode” error message |

*Table 4.3: Detailed Use Case for VerifyEmail*
### Table 4.4: Detailed Use Case for ForgotPassword

<table>
<thead>
<tr>
<th>Use Case ID</th>
<th>UC03</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor(s)</td>
<td>Registered user, Administrator</td>
</tr>
<tr>
<td>Precondition(s)</td>
<td>User should be registered in NRDC Data Steward web application</td>
</tr>
</tbody>
</table>
| Flow of Events | 1. The user must navigate to forgot password page  
2. The user must enter a valid username and email address combination in forgot password page  
3. The user must submit the form after reviewing page errors |
| Postcondition(s) | If username and email address combination is correct, then trigger SendTempPassword. Else display “Invalid combination” error message |

### Table 4.5: Detailed Use Case for SendTempPassword

<table>
<thead>
<tr>
<th>Use Case ID</th>
<th>UC04</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor(s)</td>
<td>Registered user, Administrator</td>
</tr>
<tr>
<td>Precondition(s)</td>
<td>User should submit ForgotPassword form</td>
</tr>
</tbody>
</table>
| Flow of Events | 1. Generate 6 characters’ random password  
2. Replace the previous password with a temporary password.  
3. Send temporary password to the user’s email address that is registered on database |
| Postcondition(s) | The users should receive temporary password to the email address listed on database |
Table 4.6: Detailed Use Case for ResetPassword

<table>
<thead>
<tr>
<th>Use Case ID</th>
<th>UC05</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor(s)</td>
<td>Registered user, Administrator</td>
</tr>
<tr>
<td>Precondition(s)</td>
<td>User or administrator should have received temporary password</td>
</tr>
</tbody>
</table>
| Flow of Events | 1. User or administrator should navigate to login page  
2. User or administrator should use username and temporary password to login into NRDC Data Steward web application  
3. User or administrator should create a new password and confirm it by submitting reset password page |
| Postcondition(s) | If resetting password is successful, redirect user to home page |

Table 4.7: Detailed Use Case for Login

<table>
<thead>
<tr>
<th>Use Case ID</th>
<th>UC06</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor(s)</td>
<td>Registered user, Administrator</td>
</tr>
<tr>
<td>Precondition(s)</td>
<td>User or administrator should have registered to NRDC Data Steward web application</td>
</tr>
</tbody>
</table>
| Flow of Events | 1. User or administrator should navigate to login page  
2. User or administrator should use valid username and password combination in login page |
| Postcondition(s) | If the login is successful, redirect the user to home page. Else display error messages |
### Table 4.8: Detailed Use Case for UploadData

<table>
<thead>
<tr>
<th>Use Case ID</th>
<th>UC07</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor(s)</td>
<td>Public user, Registered user, Administrator</td>
</tr>
<tr>
<td>Precondition(s)</td>
<td>None</td>
</tr>
</tbody>
</table>
| Flow of Events | 1. User or administrator should navigate to upload page  
2. User or administrator should input file details such as author name, the title of the document, description for the document.  
3. User or administration should select file to upload |
| Postcondition(s) | If the upload is successful, display “upload successful” message. Else display “upload unsuccessful” error message |

### Table 4.9: Detailed Use Case for GroupAccess

<table>
<thead>
<tr>
<th>Use Case ID</th>
<th>UC08</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor(s)</td>
<td>Public user, Registered user, Administrator</td>
</tr>
<tr>
<td>Precondition(s)</td>
<td>None</td>
</tr>
</tbody>
</table>
| Flow of Events | 1. User or administrator should navigate to upload page  
2. User or administrator should fill upload form.  
   a. If public user: access is given to “public”  
   b. If registered user: specify group names to give access  
3. Add registered group names to uploaded document’s metadata |
| Postcondition(s) | None |
### Table 4.10: Detailed Use Case for Search Data

<table>
<thead>
<tr>
<th>Use Case ID</th>
<th>Actor(s)</th>
<th>Precondition(s)</th>
<th>Flow of Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC09</td>
<td>Public user, Registered user, Administrator</td>
<td>None</td>
<td>4. User or administrator should navigate to search page</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5. Enter keywords to search in database</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>6. The application will display search results</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Postcondition(s): Display all the search results that are matching keywords. Else display “No results found”</td>
</tr>
</tbody>
</table>

### Table 4.11: Detailed Use Case for Download Data

<table>
<thead>
<tr>
<th>Use Case ID</th>
<th>Actor(s)</th>
<th>Precondition(s)</th>
<th>Flow of Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC10</td>
<td>Public user, Registered user, Administrator</td>
<td>Search results should have displayed on results page</td>
<td>1. User or administrator should click on the unique link display on results page for each search result</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2. Open a new tab allowing user or administrator to save data to local machine</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Postcondition(s): Download the data to the local machine and close the new tab</td>
</tr>
</tbody>
</table>
### Table 4.12: Detailed Use Case for ViewData

<table>
<thead>
<tr>
<th>Use Case ID</th>
<th>UC11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor(s)</td>
<td>Public user, Registered user, Administrator</td>
</tr>
<tr>
<td>Precondition(s)</td>
<td>Search results should have displayed on results page</td>
</tr>
<tr>
<td>Flow of Events</td>
<td></td>
</tr>
<tr>
<td>1. User or administrator should click on the unique link display on results page for each search result</td>
<td></td>
</tr>
<tr>
<td>2. Open a new tab allowing user or administrator to view data if appropriate software is installed in local machine</td>
<td></td>
</tr>
<tr>
<td>Postcondition(s)</td>
<td>Display the data in browser on local machine</td>
</tr>
</tbody>
</table>

### Table 4.13: Detailed Use Case for CreateGroup

<table>
<thead>
<tr>
<th>Use Case ID</th>
<th>UC12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor(s)</td>
<td>Registered user, Administrator</td>
</tr>
<tr>
<td>Precondition(s)</td>
<td>Registered user or Administrator should login into the system successfully</td>
</tr>
<tr>
<td>Flow of Events</td>
<td></td>
</tr>
<tr>
<td>1. User or administrator should navigate to Create group page</td>
<td></td>
</tr>
<tr>
<td>2. User or administrator must choose unique group name</td>
<td></td>
</tr>
<tr>
<td>3. User or administrator must enter description for group before submitting request</td>
<td></td>
</tr>
<tr>
<td>Postcondition(s)</td>
<td>If group created successfully, display “Group created successfully” message, else display “Group creation unsuccessful” message</td>
</tr>
</tbody>
</table>
Table 4.14: Detailed Use Case for AddUsers

<table>
<thead>
<tr>
<th>Use Case ID</th>
<th>UC13</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor(s)</td>
<td>Registered user, Administrator</td>
</tr>
<tr>
<td>Precondition(s)</td>
<td>Registered user must have created group successfully in create group page</td>
</tr>
</tbody>
</table>
| Flow of Events | 1. User or administrator should navigate to add users page  
2. User or administrator must enter group name  
3. User or administrator must enter usernames in comma-separated values and submit page |
| Postcondition(s) | If usernames are added successfully, display “users added successfully” message, else display “adding users unsuccessful” message |

Table 4.15: Detailed Use Case for DeleteUser

<table>
<thead>
<tr>
<th>Use Case ID</th>
<th>UC14</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor(s)</td>
<td>Administrator</td>
</tr>
<tr>
<td>Precondition(s)</td>
<td>User must be registered in database</td>
</tr>
</tbody>
</table>
| Flow of Events | 1. Administrator should navigate to delete user page  
2. Administrator should enter usernames in comma-separated values  
3. Inform users regarding their account deletion |
| Postcondition(s) | If usernames are deleted successfully, display “users deleted successfully” message, else display “deleting users unsuccessful” message |
### Table 4.16: Detailed Use Case for DeleteGroup

<table>
<thead>
<tr>
<th>Use Case ID</th>
<th>UC15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor(s)</td>
<td>Administrator</td>
</tr>
<tr>
<td>Precondition(s)</td>
<td>User group must be registered in database</td>
</tr>
<tr>
<td>Flow of Events</td>
<td></td>
</tr>
<tr>
<td>1. Administrator should navigate to delete group page</td>
<td></td>
</tr>
<tr>
<td>2. Administrator should enter group names in comma-separated values</td>
<td></td>
</tr>
<tr>
<td>3. Notify group administrator regarding their group deletion</td>
<td></td>
</tr>
<tr>
<td>Postcondition(s)</td>
<td></td>
</tr>
<tr>
<td>If groups are deleted successfully, display “group deleted successfully” message, else display “deleting group is unsuccessful” message</td>
<td></td>
</tr>
</tbody>
</table>

### Table 4.17: Detailed Use Case for AdminAccess

<table>
<thead>
<tr>
<th>Use Case ID</th>
<th>UC16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor(s)</td>
<td>Administrator</td>
</tr>
<tr>
<td>Precondition(s)</td>
<td>User must be registered in database</td>
</tr>
<tr>
<td>Flow of Events</td>
<td></td>
</tr>
<tr>
<td>1. Administrator should navigate to admin access page</td>
<td></td>
</tr>
<tr>
<td>2. Administrator should enter usernames in comma-separated values to provide administrator access</td>
<td></td>
</tr>
<tr>
<td>3. Users will now have access to administrator functionalities</td>
<td></td>
</tr>
<tr>
<td>Postcondition(s)</td>
<td></td>
</tr>
<tr>
<td>If giving access is successful, display “user added as administrator” message, else display “user not added as administrator” message</td>
<td></td>
</tr>
</tbody>
</table>
### Table 4.18: Detailed Use Case for Contact

<table>
<thead>
<tr>
<th>Use Case ID</th>
<th>UC17</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor(s)</td>
<td>Public user, Registered user</td>
</tr>
<tr>
<td>Precondition(s)</td>
<td>None</td>
</tr>
</tbody>
</table>
| **Flow of Events** | 1. Users should navigate to contact page  
2. User should enter their information such as full name, email address, subject and brief message  
3. An email will be sent to the administrations along with the user details provided in Step 2 |
| Postcondition(s) | Display “email sent successfully” message |

### Table 4.19: Detailed Use Case for Logout

<table>
<thead>
<tr>
<th>Use Case ID</th>
<th>UC18</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actor(s)</td>
<td>Registered user, Administrator</td>
</tr>
<tr>
<td>Precondition(s)</td>
<td>Registered user or Administrator should login into the system successfully</td>
</tr>
</tbody>
</table>
| **Flow of Events** | 1. Users should navigate to logout page  
2. Clear session variables for full name, user name, admin status etc  
3. User will be redirected to index page |
| Postcondition(s) | Display “logged out successfully” message |
4.4 TRACEABILITY MATRIX

A traceability matrix for requirements and use cases are shown in Table 4.21. This traceability matrix helps developers and testers for appropriately referencing the functional requirements to the use cases to track if all requirements have been taken into consideration during integration testing.

Table 4.20: Requirements Traceability Matrix

<table>
<thead>
<tr>
<th>Requirements</th>
<th>UC01</th>
<th>UC02</th>
<th>UC03</th>
<th>UC04</th>
<th>UC05</th>
<th>UC06</th>
<th>UC07</th>
<th>UC08</th>
<th>UC09</th>
<th>UC10</th>
<th>UC11</th>
<th>UC12</th>
<th>UC13</th>
<th>UC14</th>
<th>UC15</th>
<th>UC16</th>
<th>UC17</th>
<th>UC18</th>
<th>UC19</th>
<th>UC20</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC01</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UC02</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UC03</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UC04</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UC05</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UC06</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UC07</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UC08</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UC09</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UC10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UC11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UC12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UC13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UC14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UC15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UC16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UC17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UC18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UC19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>UC20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
### 4.5 GLOSSARY OF TERMS

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Steward</td>
<td>Management and oversight of the organization's high-quality data</td>
</tr>
<tr>
<td>Unstructured Data</td>
<td>Information that does not have a pre-defined model or not organized in a pre-defined manner</td>
</tr>
<tr>
<td>Cross-platform</td>
<td>Computing methods that are implemented and inter-operated on multiple computer platforms</td>
</tr>
<tr>
<td>Python</td>
<td>A high-level, object-oriented and dynamic programming language</td>
</tr>
<tr>
<td>Flask</td>
<td>A popular extensible web framework for developing web applications using Python</td>
</tr>
<tr>
<td>Structured Data</td>
<td>Data that resides in a fixed field within a record or a file such as relational databases and spreadsheets</td>
</tr>
<tr>
<td>Responsive web pages</td>
<td>An approach to present web design aimed to provide an optimal viewing experience</td>
</tr>
<tr>
<td>MongoDB</td>
<td>A NoSQL document-oriented database for storing unstructured data</td>
</tr>
<tr>
<td>MySQL</td>
<td>An open-source RDBMS that stores data in the form of rows and columns in tables</td>
</tr>
<tr>
<td>NoSQL</td>
<td>A mechanism for data storing and retrieval from unstructured databases</td>
</tr>
</tbody>
</table>
5 SOFTWARE MODELING

In this chapter, the detailed system design of the NRDC Data Steward web application is discussed. Software design artifacts such as synopsis diagram, system-level diagram, data design, system activity diagram, pseudocode, and user interface design are presented in detail.

5.1 SYNOPSIS DIAGRAM

The synopsis diagram of NRDC Data Steward web application is shown in Figure 5.1. This diagram shows different activities that the users can perform in NRDC Data Steward web application and all the external data storages that the system has.

![Synopsis diagram of NRDC Data Steward](image)

*Figure 5.1: Synopsis diagram of NRDC Data Steward*
MongoDB document storage and MySQL database are the external storages that are used in the NRDC Data Steward web application. The View Data functionality provides access to the users based on the different levels of access given to the data. Upload Data functionality helps the users to upload the research data in different formats such as datasets, conference paper, presentation slides, videos, pictures, surveys etc. while the Download Data functionality provides access to the users to download the content from the application. In Search Data functionality, the users can search the research data based on the keywords and the View Data functionality will display the results on the web page. The Group Access functionality provide flexibility of giving specific access to the documents that have been uploaded by the user. Create Group functionality is used to combine multiple users into a single group and provide access to particular files to the members of the group. Delete User, Delete Group, and Admin Access are the functionalities given to an administrator. An administrator uses Delete User functionality to delete any user from the NRDC Data Steward database upon their request or if the user is violating the terms and conditions of NRDC Data Steward web application. Delete Group functionality is an administrator functionality that helps the administrator to delete any user or group created by the user in the NRDC Data Steward. Administrator access can be given to any registered user in the system by another administrator using Admin Access functionality in the web application.

5.2 SYSTEM-LEVEL DESIGN

A high-level system design of NRDC Data Steward web application is shown in Figure 5.2. The system design shows the structure and different subsystems in NRDC Data Steward web application. It depicts the different subsystems or modules in this web application and that subsystems are interrelated.
Figure 5.2: High-level System design
The Registration subsystem allows any user to register in the NRDC Data Steward web application. The VerifyEmail subsystem is triggered by the registration subsystem that sends a verification code to the email address that was used by the user during registration. The Forgot Password subsystem allows the user to reset their password. Once the user submits the valid combination of username and email address, this subsystem triggers the email server to send a temporary password to the registered email address. The Reset Password subsystem allows the user to sign in to NRDC Data Steward web application using the temporary password sent to the user’s registered email address. The user can create a new password and log in to NRDC Data Steward web application.

The Login subsystem allows the user to sign in to the NRDC Data Steward web application using a valid username and password combination that is registered in the system. After login, the user gains access to different subsystems based on their level of authorization. A registered user has access to all User Functions including subsystems like upload, search, download, view, create group, and add users. The user with administrator access has access to Admin Functions that includes access to all user functions along with functions such as delete user, delete group, and admin access subsystems. The Upload subsystem allows the user to upload a set of research data and also give access to public or specific group. The Search module allows the user to search NRDC Data Steward for research data. The View and Download modules allow the user to view the research data on the client system. The Create Group subsystem allows the user to create a new group and the Add Users subsystem allows the admin to add other registered users to the group that the user created.
The Administrator on NRDC Data Steward web application has access to subsystems such as Delete User, Delete Group, and Admin Access. The Delete User subsystem allows the administrator to remove any registered user from the system. The Delete Group subsystem allows the administrator to remove any group that the user has created. The Admin Access subsystem allows the administrator to provide administrative access to other registered users.

5.3 DATA DESIGN

NRDC Data Steward web application stores all the user information, uploaded data and groups information in back-end databases. The registered user information is stored in “user_table” in the “nrdc” database created in MySQL. The user uploads and groups are stored in the “nrdc” database in MongoDB.

The “user_table” stores the registered user information that the user filled in registration page. The “user_table” consists of following columns:

username | fullname | emailID | password | isAdmin | verified | randomnum | resetpassword | regdate | lastlogin

In the user_table, the column username is a primary key and is helpful in uniquely identifying any user in NRDC Data Steward web application. The “username” column is the unique user id, “fullname” is the name of the user, “emailID” is the email address of the user, password is the encrypted password, “isAdmin” indicates whether the user is an administrator, “verified” indicates if the user email was verified, “randomnum” stored 6-digit integer that is used to verify user’s
email address, “resetpassword” field indicates if the user has reset password, “regdate” and “lastlogin” are date datatypes that stores the date when the user first registered and the date of most recent login. The NOT NULL constraint is given to columns username, fullname, emailID and password. A default character ‘N’ is assigned to isAdmin, verified, and resetpassword columns. If the resetpassword field has ‘Y’, this represents that the password field has a randomly generated 6-character password that helps the user to temporarily login and reset password.

The “uploads” collection in MongoDB stores all the research data that is uploaded to the NRDC Data Steward web application. Each document in the uploads collection consists of the following fields:

```plaintext
_id | binFile | description | tags | filename | fullname | username | group | title | filetype | authorname
```

where “_id” is an ObjectID that uniquely identifies a document in upload collection. The “binFile” field stores the binary format of any file that the user uploaded from upload page and the rest of the fields for each document in uploads collection is the metadata for the file stored in binFile. The “description” field consists of a brief explanation regarding the uploaded file. The “tags” field is a list of keywords that categorizes the uploaded file. The user can add any number of keywords to the tags field. The “filename” stores the original name of the file when the user uploaded the file. The “fullname” field consists of the name of the user who uploaded the file to the database. The “username” field stores the user id of the user uploading the file to the database. If the user is logged into NRDC Data Steward web application, then the username stores the username value from session. If the user is not logged into the NRDC Data Steward web application, then the
username stores an ‘N/A’. The “group” field consists of the list of group names that have access to view or download this document from database. If the user uploads files to NRDC Data Steward web application without signing in, then the group fields store ‘public’ and anyone can view or download that file from database. The “title” field stores the title of the uploaded file that the user filled in upload form. The “filetype” field stores the type of the file that helps the application to identify that software is needed to view file in client system. The “authorname” field stores the author name of the file that the uploader entered in upload form.

The “userGroups” collection in MongoDB database stores the groups information that the users have created in the create group page in NRDC Data Steward web application. The fields in userGroups collection are as shown below:

<table>
<thead>
<tr>
<th>_id</th>
<th>groupName</th>
<th>groupAdmin</th>
<th>description</th>
<th>users</th>
</tr>
</thead>
</table>

Where _id is an ObjectID that is helpful in uniquely identifying any document in “userGroups” collection. The “groupName” field uniquely identifies the name of a group that the user created in create group page. Even though no primary key constraint is given to groupName field, a code is implemented in application that will check if the group name already exists in the database. The “groupAdmin” consists of the username of the user who created the group in create group page. The “description” field consists of a summary of the group. The “users” field is a list that stores usernames of users that were added by the group admin in add users page.
5.4 DETAILED DESIGN

5.4.1 SYSTEM ACTIVITY DIAGRAM

The System-Activity diagram of NRDC Data Steward web application is shown in Figure 5.3. The system activity diagram shown in Figure 5.3 illustrates the flow from one activity or module to another activity or module in NRDC Data Steward web application. The information inflow and outflow from the MySQL storage happens mainly in registration and login module. The data from registration page flows and inserts a row in MySQL database. The login activity validates the user credentials by checking the username and password combination in user table in MySQL. If the credentials entered by the user are correct, then the user can view main screen that has options like upload data, search data, view or download data etc. In the user functions, the user uploaded data is stored in MongoDB database. The user search happens in the MongoDB on several fields in uploads collection. If results are found for the search, the application will display the search results. Once the results are shown on search page, the view and download activities help the user to view file on client machine.
Figure 5.3: System-Activity Diagram
5.4.2 PSEUDO-CODE

Sample pseudo-code segments from the detailed design of the NRDC Data Steward web application are presented in this subsection.

verify login credentials
if successful
    proceed to main screen
else
    display error message
    if user forgot password
        send temporary password to email id
    return to login screen

user chooses the options displayed on main screen

case UploadData:
    wait for the user to enter author name, title, description
    wait for the user to enter group names to give access
    wait for the user to select file from file system
    insert a new document with file and file information in MongoDB database
    if upload is successful
        display success message
    else
        display an error message

case SearchData:
    wait for the user to enter keywords to search
    search keywords for a match in multiple fields in uploads collection
if match found
   display search results on user screen
else
   display an error message

**case ViewData:**
wait for the user to select the file to view from search results page
check access level of the user and the file
if authorized user
   retrieve file from uploads collection in MongoDB database
display retrieved data on the screen
else
   display an error message

**case DownloadData:**
wait for the user to select the file to download from search results page
check access level of the user and the file
if authorized user
   retrieve file from uploads collection in MongoDB database
download retrieved data to client machine
else
   display an error message

**case CreateGroup:**
wait for the user to enter groupname
insert a document with the group information in userGroups collection
wait for the user to enter group member usernames
if user name exists in the database
   add user to the group
else
   display an error message
case DeleteUser:
    wait for the administrator to enter username
    if username exists in the database
        delete user info row from database
    else
        display an error message

case DeleteGroup:
    wait for the administrator to enter groupname
    if groupname exists in the database
        delete group info document from database
    else
        display an error message

case AdminAccess:
    wait for the administrator to enter username
    if username exists in the database
        update isAdmin = 'Y' in database table for the user
    else
        display an error message

5.5 IMPLEMENTATION & WALK-THROUGH OF NRDC DATA STEWARD

This section discusses the prototype implementation of NRDC Data Steward web application. After the design phase, the development and implementation of NRDC Data Steward has begun. This section focuses on the implementation of the different subsystems along with screenshots of the user interface.
5.5.1 HOME PAGE

Upon launching the NRDC Data Steward web application on the screen, the home page appears. The home page has a simple design, as shown in Figure 5.4, with links to other functionalities such as upload, search, register, login and contact. The user gets a brief introduction about NRDC such as the mission statement and history of NRDC in the About us section of the homepage.

![Home page screenshot](image)

Figure 5.4: Home page

5.5.2 UPLOAD PAGE

The user navigates to the upload page to upload a set of research data and results to the NRDC Data Steward web application. Figure 5.5 below is the screenshot of the upload page. While uploading a file in the upload page, the user should enter information regarding the file such as
author name, title of the document, brief description, tags, and group name for group access. If the user is not logged into NRDC Data Steward, then they must fill two extra fields that are full name and email address of the uploader. The mandatory fields that the user must enter for successfully uploading file are author name, title of the document, if the user is logged in and full name, email address of the uploader in addition, if the user is not logged into NRDC Data Steward.

![Upload Research Data](image)

**Figure 5.5: Upload Page**

### 5.5.3 SEARCH PAGE

The user navigates to the search page as shown in Figure 5.6 to search for research data and results that are stored in NRDC Data Steward server. The user needs to enter specific keywords in the search field and submit the form by clicking on the search button. The results align with the keywords entered by the user will be displayed on the same page. Every corresponding result
displayed on the search results page contains the title of the document, author name, description of the document, name of the uploader, and the original file name. The user clicks on the title name to view or download the document.

![Search results page](image)

*Figure 5.6: Search results page*

### 5.5.4 CREATE GROUP PAGE

The user navigates to the create group page for creating a group. Groups can also be used for providing restricted access for the members of the group. The user must define the group’s name and description, that are mandatory fields. The group name must be unique. The Figure 5.7 shows the create group page in the NRDC Data Steward web application that performs the above described functions.
The group admin navigates to the add users page in Figure 5.8. The page displays all the groups that the current user has created or is listed as an administrator. The admin enters the group name that is listed in the table and enters the other registered users to be added in the group. This page allows the group admin to add more than one participant by separating the names using commas. Only registered users of the Data Steward can be added to the group.
5.5.6 ADMINISTRATOR PAGE

The user who is registered as administrator of the group can access this page. From the Admin tab, the administrator can choose to navigate to one of the options listed in the drop down box.

The pages that are currently available for the administrator are delete user page, delete group page, admin access page. In the delete group page, the administrator should enter a valid group name to perform delete group operation. Figure 5.9 is the screenshot of the delete group page that can only be access by the administrator. Group name field is mandatory.
Similar to the delete group page, the administrator has access to delete one or more registered user accounts from the NRDC Data Steward at any given time. The delete user page in the Data Steward web application is as shown in Figure 5.10. The administrator must enter the username to delete their account information from the application. The administrator can remove multiple users from the system by entering usernames in comma separated values in the username field shown in Figure 5.10.
The admin access page allows the administrators of the NRDC Data Steward web application to give administrator privileges to other registered users in the system. The administrator navigates to the admin access page from admin tab to perform this operation. To provide administrative privileges as shown in the Figure 5.11, the admin should submit the username.
5.5.7 CONTACT PAGE

The contact form provides a platform for the users to contact the NRDC with questions and queries. The user is required to enter their full name and valid email id along with the subject and content of the question. This function is shown in the Figure 5.12.

![Contact Form](image)

*Figure 5.12: Contact page*

5.5.8 REGISTRATION PAGE

For users to obtain access to the files in the database, they are required to register as a user in the NRDC website. To register, the user must provide details such as username, full name, email address and password. The user is required to retype the password for confirmation. The registration screen is shown in the Figure 5.13.
Figure 5.13: Registration page

5.5.9 LOGIN PAGE

The registered users can log in to the website to access the files in the database. As shown in the Figure 5.14, the user must provide the username and password that were used during registration in order to login. If the user forgets their password, an option called ‘Forgot password’ is available to reset the password for the username.
Figure 5.14: Login page
This chapter describes the testing of NRDC Data Steward web application and presents feedback from software developers and potential users of this application. As part of integration testing, the software developers ran the prototype of NRDC Data Steward web application in the Windows platform. This integration testing questionnaire followed a software testing session in that the web application was tested and receive feedback from software developers. The software testing questionnaire is shown in Figure 6.1.

Q1. Did the software work as expected? Please comment to what extent [specification].
Q2. What features do you think are important for this software application? [specification].
Q3. What features are less important for application testing against its specification?
Q4. Related to Q2 and Q3 above, what features would you recommend to either add and remove to/from this application?
Q5. Did you find any software deficiencies? [e.g., bugs or lack of responsivity]
Q6. Do you have any technology recommendation for future versions? [e.g., architecture solutions, programming languages, development tools]
Q7. Do you have any other recommendation to improve the development process and the quality of this software application?
Q8. What is your overall impression about the purpose of this software? Please comment.
The summary of answers for each of the above questions are discussed below. A total of 5 IT professionals have tested the application in windows platform.

Question Q1 in the testing questionnaire was aimed to get feedback from testers whether the software application is working as expected. We received a positive response from the testers. The testers were satisfied with the current functionalities of NRDC Data Steward web application. The features included in NRDC Data Steward web application are very clear and easily understandable. All the functionalities are operating as expected and flawlessly. Searching and retrieving files from NRDC server seemed easy for the testers. Some of the suggestions by the testers for better operability are to add some select boxes and checkboxes to make the selection process quicker.

Question Q2 was designed to receive feedback from testers regarding the important features for NRDC Data Steward web application. We received different responses from the testers. According to the testers, the main features of NRDC Data Steward web application include search functionality and uploading different formats of the file. Setting up groups and sharing the document with a specific set of users is useful too, per testers. Other important features of the NRDC Data Steward web application are registering users, logging in, group creation, and type of access rights.

Question Q3 refers to the features that the testers think are less important for testing. The feedback we received from the testers is that all the functionalities included in NRDC Data Steward web application are important. However, according to two of the testers, the Contact Us page in
the application that allows the users to submit requests or report bugs to the administrator seemed to be less important. One of the suggested alternatives to replace Contact Us functionality is to simply mention the email addresses of the administrator(s) on home page of the NRDC Data Steward web application to that the users can send email messages.

Question Q4 is related to questions Q2 and Q3. Based on the feedback that the testers gave in previous questions, this question was aimed to receive feedback from the testers regarding recommendations to add or remove features from the application. From the feedback that we received from the testers, one of the common suggestions is to include filters for searching keywords in metadata fields in the NRDC Data Steward database. This would improve the search performance by searching keywords only in user selected fields. Notify the uploader of the document when their document is shared among the set of users. Include functionality to assign read, write, and share permissions to other users. In the document search include a feature to search for keywords within the document along with the metadata. Integrate social media networks option to share the document link to other users via Facebook, Twitter etc. and allow user registration using existing Facebook, Gmail, or Yahoo accounts. Suggested addition of features for registration module include the ability to unregister from the application and possibility to edit account settings.

Question Q5 in the testing questionnaire was intended to verify whether the testers encountered any software deficiencies while running NRDC Data Steward web application. The testers did not find any bugs or lack of responsivity while using the application.
Question Q6 was intended to get feedback on technology currently used and recommendations for future versions such as architectural solutions, programming languages and development tools. The testers liked the idea of uploading and downloading documents to/from MongoDB server and did not have any suggestions for back-end development. The testers suggested the use of Bootstrap for the front-end user interface since the bootstrap components along with React can be used to make the user-interface more interactive and attractive.

Question Q7 was aimed at receiving any other recommendations that could improve the development process and the quality of the NRDC Data Steward web application. The testers have suggested to add few administrative functions and file storage to improve the quality of the application. One of the suggestions was to add a module for the administrator to back up all the data in another database for recovery. Use of loggers for debugging in case of any application failure was also suggested. Also, the application can be dockerized to improve the portability and scalability of the application. To improve the quality of this application, the testers also recommended to allow the application to upload larger files.

Question Q8 in the software testing questionnaire we received positive responses from the testers. The functionalities and ideas implemented in this application impressed the testers. They found the application pretty user-friendly, easy to use, and the documents seem to have enough protection.
7 CONCLUSIONS AND FUTURE WORK

In this thesis, we introduced a web-based platform for sharing research data and results on the public domain. The web platform allows users to upload data that can be accessed by the public or private groups for faster data sharing. The files available in the Nevada Research Data Center can be accessed from this web platform, along with options to contribute to the existing datasets.

To enhance the user experience, we have provided functions for people to create groups and share data that will be available exclusively to users in the group. Furthermore, the logins and passwords are protected by a second level authentication sub-system where random passwords are generated and sent to registered email addresses of first-time users.

The uploaded files are stored in the MongoDB database, that is an unstructured database that can store large files along with its metadata. Unlike most data storage platforms that store the file in the file system and the corresponding link and its meta-data in the database, NRDC Data Steward stores the file as binary data in the database along with the metadata. This also enables the user to upload multimedia files such as photos, videos, presentations, and webpages.

This work aims to provide a better and more efficient platform for academic and scholarly data sharing that can be useful for students, professors, data analysts, researchers, and other
professionals. This web platform known as the NRDC Data Steward is a prototype for a globally accessible database.

The main contributions of the work presented in this thesis include a thorough survey of different data forms, research on tools to develop this data-centered application, requirements specification, detailed system design, data design, implementation of the application and finally analysis of the answers received from software developers following their software testing session.

As part of this thesis, I have achieved a significant success in developing a working prototype of the NRDC Data Steward web application using new software technologies that are not available in many web applications. Unlike the traditional uploading and downloading to/from the server filesystem, I have designed the application such that all uploading and downloading of data happens to/from the MongoDB server. The metadata of the uploaded file is stored along with the file itself in MongoDB, unlike traditional web applications that store the file in the server filesystem and the metadata along with the link to the file is stored in the relational database systems. This, in summary, is a modern and advanced, technological approach.

In the future, we plan to increase the storage limit for each file that the user uploads to the NRDC database. Currently, MongoDB allows users to store files that are 16 MB or less. This file size limit can be extended by using a GridFS in MongoDB, allowing the users to upload larger files of data. The Grid file system in MongoDB divides the uploaded file into chunks each of size 255 KB [28].
The platform created for this thesis can also incorporate the option to report inappropriate documents to the system, to maintain its integrity. The present system allows only the group admin to add members to a group that can be developed further to allow the creator of the group to assign administrative authority to other members of the group. Tags for the uploaded files can be provided currently only by the uploaders but, in the future, we may include the option for receiving tag suggestions from other users as well.

Currently, the search indexes in MongoDB are limited, that results in slower searches. By creating new indexes in the future, we can provide faster search results in the database. Another aspect that can be added to the search option is allowing the users to search for content based on categories such as author name, date etc.

To make the platform more user-friendly, we could convert the NRDC Steward into a responsive web page that would enable users to access it from all devices such as tablets and phones.

Finally, we also plan to create a mobile application for the same purpose for immediate access and simple viewing.
REFERENCES


