SE640 Foundation of Software Engineering
Fall, 2020
Due: December 3, 2020
Professor: Dr. Maninder Singh

Sharatha D. JayaKumar
Keshav Singhal
Paul Court

Semantic Analysis Project

Table of Contents Table of Contents

Software Requirements Specification Document

Version: (0) Date: (12/03/2020)
Version: (1.0) Date: (12/9/2020)
Version: (2.0) Date (12/12/2020)
Version: (3.0) Date (12/13/2020)

1. INTRODUCTION

1.1 PURPOSE

The purpose of this document is to build a web interface to perform the semantic analysis of an SRS document and to implement change impact analysis. The web interface allows a user to upload an SRS file, then its content (i.e., requirements) are displayed in the browser. The web interface provides the capability to generate a semantic graph of the requirements. Also, while fixing a requirement post inspection (based on faults listed by the inspectors), the application provides ability to highlight all semantically similar requirements as per the semantic graph generated.

1.2 PROJECT SCOPE

The purpose of the Semantic analysis project is to build a web interface to perform the semantic analysis of an SRS document and to implement change impact analysis. The user can upload an SRS file as a pdf or a text file., then its content (i.e., requirements) are displayed in the browser. The file will not be more than 10MB in size and in no other format than the pdf or text format are identified. Once the SRS document file has been uploaded, then its content (i.e., requirements) should be displayed in the browser in a scrollable text box within the web page. The interface provides capability to generate a semantic graph of the requirements within the web page. Also, while fixing a requirement post inspection (based on faults listed by the inspectors), the application provides the user with the capability to highlight all semantically similar requirements as per the semantic graph generated. The semantic analysis project requires an internet browser (Chrome, Firefox, Explorer, or Edge) to be accessed. For this iteration of the semantic analysis project, our scope consists of the following tasks to perform.

1. Generate GUI prototype.
2. Evaluate GUI against the eight golden rules for interfaces.
3. Create web design implementation that runs on AMPPS environment.
4. Develop documentation supporting the tasks completed.
5. Test the implementation using a robust set of possible failure events.

Back to Top

2. OVERALL DESCRIPTION

2.1 PRODUCT FEATURES

A brief component design of a potential system with an external database is shown below. The external database is not implemented in this iteration of the software development life cycle.

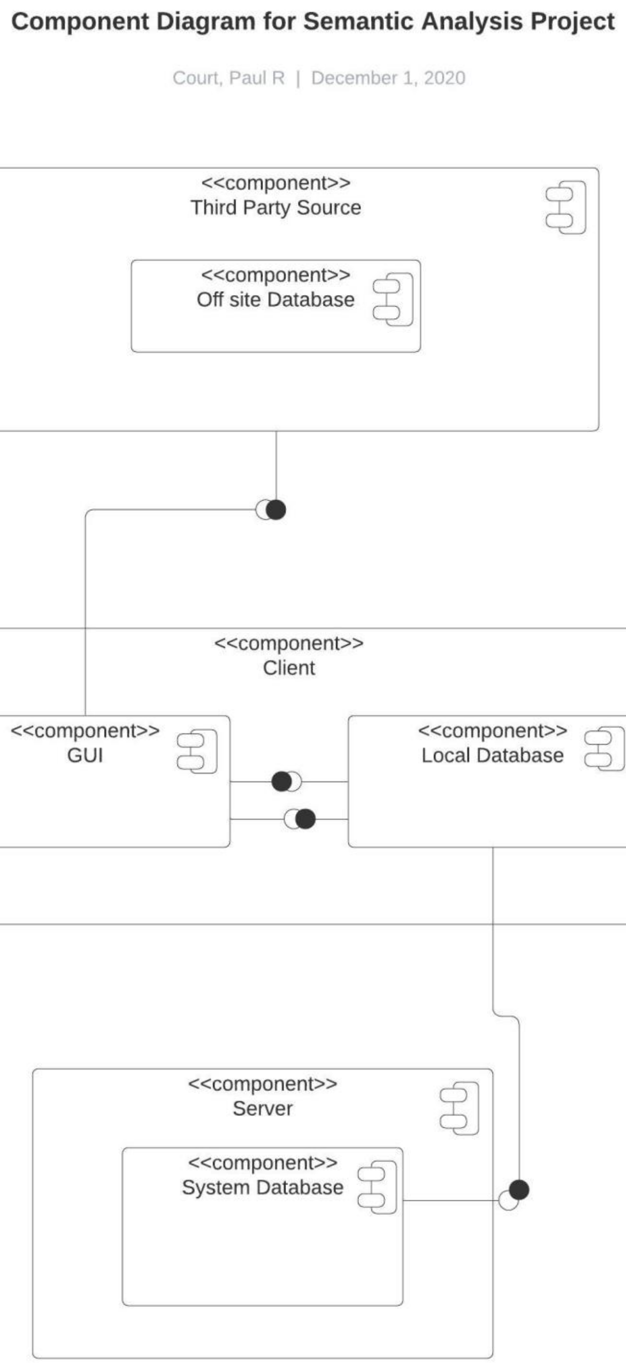
Component Diagram for Semantic Analysis Project
Court, Paul R | December 1, 2020
<<component>> Third Party Source
<<component>> Off site Database
<<component>> Client
<<component>> GUI
<<component>> Local Database
<<component>> Server
<<component>> System Database

Figure 1. Component diagram of the Semantic Analysis System.

[Back to Top]

2.2 OPERATING ENVIRONMENT

The operating environment for the student course management system is as listed below.

- ▪ Client/server system
- ▪ Operating system: Windows
- ▪ Database: File System (temporary folder /uploads)
- ▪ Platform: HTML/CSS/PHP/Python

2.3 DESIGN AND IMPLEMENTATION CONSTRAINTS

There are several constraints either in design or implementation. The first of which is the file formats that are allowable. Only files with a .txt or .pdf are able to be uploaded. Once uploaded, the files can be displayed, but only the .txt files can be modified. It is the preferred file format for that reason. .pdf files cannot be converted to semantic similarity tables and therefore, cannot be viewed in a similarity graph.

This version of the semantic analysis system cannot convert documents to similarity tables and hence, will only work with the dummy file sent by Dr. Singh. Once that functionality is provided, the requirements document will be evaluated for similarity and a similarity table can be saved in the graph_file_LDA.txt file. For now, a few dummy files of possible semantic tables developed for the previous parts of this project have been uploaded and will be substituted for the ones that will eventually be provided by the AI that evaluates an SRS document for semantic similarity. At that point, the graph of the semantic similarities uploaded can be shown and the requirements similar to any selected requirement will be able to be highlighted.

Lastly, as the python code exists now, the nodes in the graph are not labeled and the strength of the association between the requirements has not been considered.

Back to Top

3. SYSTEM FEATURES

3.1 FUNCTIONAL REQUIREMENTS

The user of the semantic analysis web interface should be able to do the following:

- ▪ The user should be able to upload an SRS file as a pdf or a text file. The file should not be more than 10MB in size.
- ▪ The user should be able to view and edit the available SRS documents in the browser. then its content (i.e., requirements) are displayed in the browser.
- ▪ The user should be able to generate a semantic graph of the related requirements. ▪ The user should be able to view the faults listed by the inspectors.
- ▪ While fixing a requirement post inspection (based on faults listed by the inspectors), the application provides user the capability to highlight all semantically similar requirements as per the semantic graph generated.
- ▪ The user should be able to make changes to the related requirements in the browser.

SRS line by line.

1. Upload a selected SRS document. (.txt or .pdf not more than 10MB)
2. Save the document to a temporary folder or database.
3. Display contents of the file in a text box with scroll bar.
4. Generate a semantic graph as provided via a menu option.
5. Display the graph as a pop-up or in a space provided.
6. Provide a fault-finding assistance that allows similar semantic requirements to be highlighted. (use the graph produced in requirement 4 to help with this.)

Back to Top

4. EXTERNAL INTERFACE REQUIREMENTS

4.1 USER INTERFACES

The front-end software for the display is to be created with PHP and HTML/CSS, each taking the majority role depending on the functionality necessary with CSS supporting both. The back-end software uses PHP, Python, and File System provided by the local host computer environment with potential to save to an off-site database in successive versions. Here is a mock-up of the GUI for the semantic analysis. This sketch was done using Paint 3D.

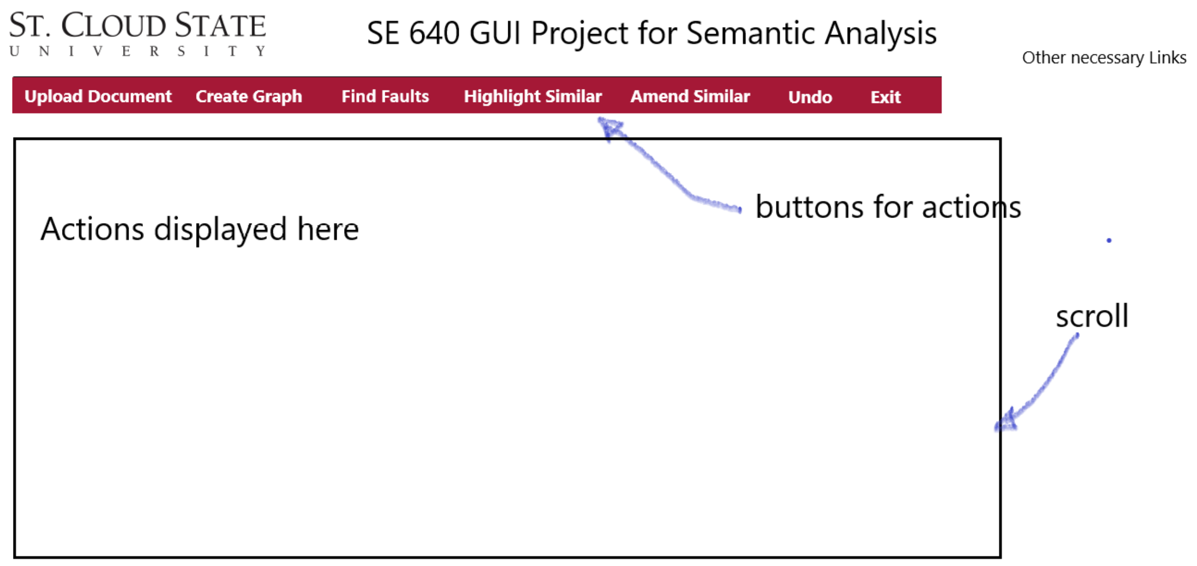

Figure 2: Prototype of Semantic Analysis Graphic User Interface.

The GUI shown above was designed with the Golden Rules of Interface Design in mind. Those attributes are listed below and summarized in the table provided by Sharatha.

S. No	Golden Rules	Our Design – Figure-1
1	Strive for consistency	**Satisfied**; We used the consistent terminology for various actions, e.g.,

		upload document, read/edit document button, submit/reset button, the color theme of our design is consistent throughout.
2	Cater to universal usability	**Somewhat Satisfied;** Easily accessible links that have appropriate names however, picture icons may be added to make the links more universal. These can be incorporated in the next version.
3	Offer informative feedback	**Satisfied**; In the ‘Upload Document’ section of our interface, if a file format other than the txt, doc, or docx has been chosen, the feedback ‘This file extension is not allowed. Please upload a txt, doc, or docx file. These are the errors shown to the users to offer informative feedback.
4	Design dialogs to yield closure	**Satisfied;** We divided the consecutive actions that have to be performed by the User for the semantic analysis of an SRS document into the sequential option menu sections on the front page of the application to yield closure of the activities.
5	Prevent errors	**Somewhat Satisfied;** So far, our application does not need any sections with error prevention options like the dropdown boxes, radio button option, etc... to prevent unwanted selections. We can incorporate this rule if needed in the next version
6	Permit easy reversal of actions	**Satisfied;** We permitted the user, easy reversal of actions. E.g., In the ‘Available SRS Documents’ section of our app, there is an Edit section in which the user can edit the uploaded document. In the section, there is a ‘Reset’ button to permit the user to go back to the original state of the document.
7	Keep users in control	**Satisfied;** We tried to keep the user in control. E.g., In the ‘Upload’ section of our application, after the user chooses the file, the file name is

		displayed as text next to the chosen file to keep the user in control prompting before the upload option. If the wrong file has been chosen by the user, there is a chance to choose the correct one. Links at the top of the page are available and will allow users to control what happens.
8	Reduce short-term memory load	**Satisfied;** Links at the top of the page will allow users to have low consequence explorations of functionality, therefore reducing the need to remember functionality of selected options.

Table 1. The 8 golden rules evaluation for Semantic Analysis group project SE 640.

Back to Top

4.2 HARDWARE INTERFACES

A Windows enabled machine that has a browser that supports PHP, Python, and a local File System.

4.3 SOFTWARE INTERFACES

Software Used	**Description**
Operating System	We have chosen Windows operating system for its best support, user-friendliness, and wide availability.
Database	To save SRS documents from a outside location we will use the chosen File System. Outside database accessibility will be addressed in successive versions.
PHP/HTML/CSS/Python	PHP/HTML alternate the lead roles in pages implemented depending on the necessary functionality. CSS supports PHP and HTML.
.txt and .pdf	These are the supported uploads to the system. .pdf files are not fully functional at this time.

Table 2: Software interfaces used and descriptions.

Back to Top

4.4 COMMUNICATION INTERFACES

This project supports all types of web browsers. We are using simple HTML web forms for the SRS semantic analysis. HTML interfaces with PHP, PHP interfaces with HTML, and PHP uses a shell_exec to interface with the python code to produce the graph and echo the results of the run.

Back to Top

5. DATA REQUIREMENTS

5.1 DATA DICTIONARY

As mentioned previously, the requirements document can be uploaded in .txt and .pdf format only at this time. The .pdf files are not fully functional. A .txt file of requirements can be uploaded, but the feature necessary to graph those requirements has yet to be implemented. A text file of the tabular representation of the semantic similarity is necessary to be accessible by the python code to create the appropriate graphical depiction. A dummy file is provided with the necessary format. That text file should contain integer values of the requirement being examined for similarity followed by at least one space, then the requirement deemed to be like it via the inspection process. A printout of the dummy file included with the python code is shown below.

```
1 2  - requirement 1 is semantically similar to requirement 2.
2 3  - requirement 2 is semantically similar to requirements 3, 4, and 5.  etc.
2 4
2 5
4 5
```

Table 3: Tabular formatting necessary for creating the semantic graph.

5.2 DATA STORAGE AND RETENTION

All requirements documents uploaded to the system are stored in a folder on the local hard drive of the system running the application. It can be found http://localhost/pages/uploads/. In this iteration, there is no way to remove the items from the /uploads/ directory. This functionality was not mentioned and is not implemented. It can be included in successive iterations of the SDLC.

Back to Top

6. QUALITY ATTRIBUTES

6.1 TESTING

Only black box testing was considered for this project. Simple input and output evaluations of the implementations.

Requirement	Functionality/Link	Test	Result
The user should be able to upload an SRS file as a pdf or a text file.	Upload Document	Upload .txt document	Successful upload to /uploads/ directory.
		Upload .pdf document	Successful upload to /uploads/ directory.
		Upload .doc document	Appropriate error message output.
The file should not be more than 10MB in size.	Upload Document	None done….no files large enough.	
		Upload a file with the same name as an existing.	The new document is saved in the directory over writing the old.
The user should be able to view the available SRS documents in the browser.	Available SRS Documents	Clicked on Read Document button for one of the files listed.	Able to view the document in scrolling window provided
The user should be able to edit the available SRS documents in the browser.	Available SRS Documents	Clicked on Edit Document button for one of the .txt files listed.	Able to edit the document in the browser window and select what to do with it.
		Clicked on Edit Document button for one of the .pdf files listed.	Not able to edit this type of document.
The user should be able to generate a semantic graph of the related requirements.	Create Graph	Clicked View Graph button for one of the files listed.	Graph shown as a pop-up. Only partially satisfied as the dummy files were run…not actual semantic table. See Figure 3 and 4 below.
The user should be able to view the faults listed by the inspectors.	Highlight Similar Click then Highlight Similar Requirements	Select an uploaded file from the list.	The SRS appears in the scrollable window. The user can select a requirement.

While fixing a requirement post inspection (based on faults listed by the inspectors), the application provides user the capability to highlight all semantically similar requirements as per the semantic graph generated.	Click -select- to show the semantically similar requirements to the requirement line number.	Scrolled to a requirement.	Displays the semantically similar requirement and their line numbers. Note: If changes are actually made, they must be done using the Edit Document functionality
The user should be able to make changes to the related requirements in the browser.	Available SRS Documents….click Edit Document.	Edited a line of the document and clicked Submit.	The document was saved as edited. The old file was overwritten.
The user should be able to make changes to the related requirements in the browser.	Available SRS Documents….click Edit Document.	Edited a line of the document and clicked Reset.	The document was unchanged.

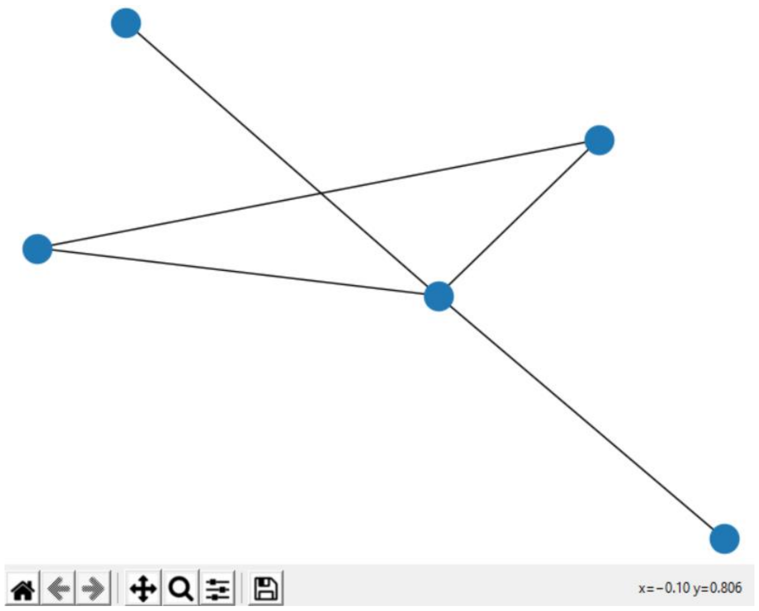

Figure 3: Pop-up window of semantic graph of the original dummy file sent with the python code Graph_LDA_2.0.py.

Back to Top

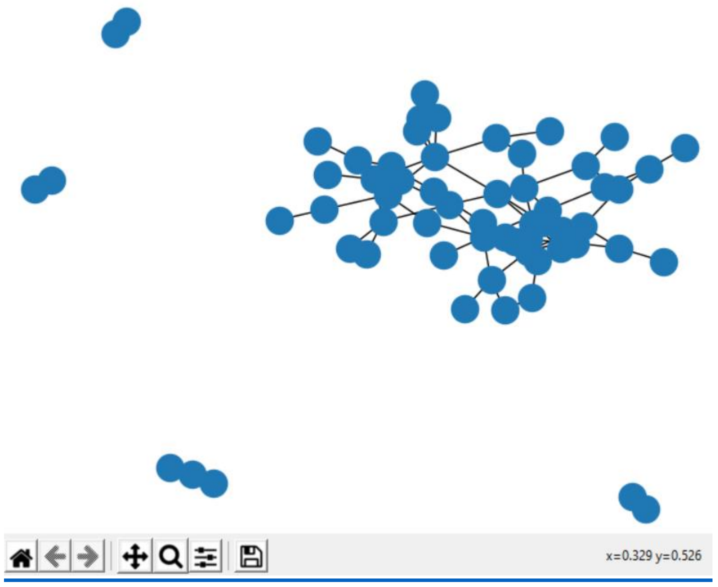

Figure 4: Pop-up window of semantic similarity graph of the PGCS SRS document as seen by Paul Court for an earlier assignment sent to the python code Graph_LDA_2.0.py as graph_file_LDA.txt.

6.2 SECURITY

We have taken no security measures during this iteration. No code masking or hiding of data or implementation codes has been done.

7. IMPLEMENTATION CODES

7.1 EXPLAINATION OF IMPLEMENTATION

The code used to implement the GUI consists of php and HTML. Place the 'pages' folder in the Ampps/www/ directory along with the SematicAnalysisMain.html. Be sure the Ampp application is running. Open a browser and type localhost/SemanticAnalysisMain.html to explore the functionalities. I have a few files loaded into the 'uploads' folder found in the 'pages' directory. There are issues uploading documents with the .doc and .docx. Apparently, there are issues with licensing that will need to be addressed.

Semantic Analysis main page. (successive pages and the css code that assists them are hidden in a local directory folder named "pages" with images stored in a sub-directory "Images".
I also had a folder named "Images" that held the graphics for the St. Cloud State logo and other necessary graphics. They were borrowed from the web page found at https://www.stcloudstate.edu/. We are not using this for monetary gain so their color scheme and graphics use should be of no copywrite infringement. If necessary, these details can be amended.

Back to Top

7.2 HTML/CSS/PHP CODES

The code files used for the development of the home page and successive pages of the site are found below.

SemanticAnalysisMain.html:

```
<html>
 <meta name="author" content="Paul Court">
 <meta name="description" content="This project has been developed for Dr. Maninder Sign SE640
Fall of 2020.  It will help with semantic analysis of
software requirements specifications.  The development team consists of Keshav Singhal, Paul
Court, Sharatha Jayakumar, and Dr. Maninder Singh.">

  <head>
        <title>SE 640 Semantic Analysis Project</title>
        <link rel="stylesheet" href="pages/stylesheet.css">
  </head>

<body bgcolor = "#ffffff">
<hr width = "100%" height = "1px" color = "#B62010">

<p align = "center">
<table width = "100%" bgcolor = "#FFFFFF" border = "0px" cellpadding = "4 px">

 <tr>
  <td width = "20%">
<p align = "left"> <img src = "pages/Images/header-logo.png" alt = "St. Cloud State University
Logo"> </p>
  </td>
  <td width = "60%" align = "center">
  <font size = "+3" >

               <img src = "pages/Images/stc.png" width = "35 px" alt = "SCSU 'C' Logo">
               &nbsp;&nbsp; SE 640 Semantic Analysis Project &nbsp;&nbsp;
               <img src = "pages/Images/stc.png"  width = "35 px" alt = "SCSU 'C' Logo"> <br>

  </font>
  </td>
  <td width = "20%"> </td>
 </tr>

 <tr bgcolor = "#B62010" height = "15px">
   <td colspan = "3" >
     &nbsp;&nbsp; &nbsp;&nbsp;&nbsp;&nbsp; &nbsp;&nbsp;
        <a href = "pages/uploadDoc.php" target = "lowerFrame"> Upload Document </a>&nbsp;&nbsp;
&nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp;
        <a href = "pages/listFiles.php" target = "lowerFrame"> Available SRS Documents
</a>&nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp;
        <a href = "pages/graphCreator.php" target = "lowerFrame"> Create Graph </a>&nbsp;&nbsp;
&nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp;
        <a href = "pages/highlightSimilar.php" target = "lowerFrame"> Highlight Similar
</a>&nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp; &nbsp;&nbsp;
        <a href = "https://www.stcloudstate.edu/" target = "_top"> SCSU Home Page </a>
   </td>
 </tr>
 <tr>
   <td colspan = "3" height = "750px">
   <iframe src = "pages/initialPage.html" width = "90%" height = "75%" name = "lowerFrame" id =
"lowerFrame">
   <p> Frames are not supported by this browser! </p>
   </iframe>
   </td>
 </tr>
</table>
</p>
</body>
</html>
```

Back to Top

uploadDoc.php

```
<!-- HTML edited by Paul Court php written by Keshav Singhal -->
<html>
 <meta name="author" content="Keshav Singhal">
 <meta name="description" content="This page allows the user to upload a document from a chosen
directory and
 stores it into the local directory named 'uploads'.">

  <head>
        <title>SE 640 Semantic Analysis Project</title>
        <link rel="stylesheet" href="stylesheet.css">
  </head>

<body bgcolor = "#ffffff">

<?php
error_reporting(0);
    $currentDirectory = getcwd();
    $uploadDirectory = "/uploads/";

    $errors = []; // Store errors here

    $fileExtensionsAllowed = ['txt', 'doc', 'docx']; // These will be the only file extensions
allowed

    $fileName = $_FILES['the_file']['name'];
    $fileSize = $_FILES['the_file']['size'];
    $fileTmpName  = $_FILES['the_file']['tmp_name'];
    $fileType = $_FILES['the_file']['type'];
    $fileExtension = strtolower(end(explode('.',$fileName)));

    $uploadPath = $currentDirectory . $uploadDirectory . basename($fileName);

    if (isset($_POST['submit'])) {

      if (! in_array($fileExtension,$fileExtensionsAllowed)) {
        $errors[] = "This file extension is not allowed. Please upload a txt, doc or docx file.
";
      }

      if ($fileSize > 4000000) {
        $errors[] = "File exceeds maximum size (4MB)";
      }

      if (empty($errors)) {
        $didUpload = move_uploaded_file($fileTmpName, $uploadPath);

        if ($didUpload) {
          echo "The file " . basename($fileName) . " has been uploaded.  ";
        } else {
          echo "An error occurred. Please contact the administrator.  ";
        }
      } else {
        foreach ($errors as $error) {
          echo $error . "These are the errors " . "\n";
        }
      }

    }
?>

<!-- HTML form -->

<form action="uploadDoc.php" method="post" enctype="multipart/form-data">
        <font size = "+2">Click <b>Chose File</b> to upload a new document then
<b>Upload</b>:</font><br><br>
        <input type="file" name="the_file" id="uploadDoc">
               <br><br>
        <input type="submit" name="submit" value="Upload">
    </form>
</body>
</html>
```

Back to Top

listFiles.php

```
<!-- HTML edited by Paul Court php written by Keshav Singhal -->
<html>
 <meta name="author" content="Keshav Singhal">
 <meta name="description" content="This page will list the uplaoded files available for display
or for edit.">

  <head>
        <title>SE 640 Semantic Analysis Project Available Files</title>
        <link rel="stylesheet" href="stylesheet.css">
  </head>

<body bgcolor = "#ffffff">
<font size = "+2"> Select a file from the list of uploaded documents available.  If empty, choose
<b>Upload Document</b> to add a file.</font>

<table style="width:100%">
  <tr>
    <td><b>Filename</b></td>
    <td><b>Link to Read</b></td>
    <td><b>Link to Edit</b></td>
  </tr>
 <tbody>
<?php
  $files = array_slice(scandir('./uploads/'), 2);
  foreach ($files as $file){
        echo'<tr>';
        echo'<td>'. $file.'</td>';
        echo''." ".'';
        echo''." ".'';
        echo'<td><a href="./uploads/'.$file.'">'."Read Document".'</a></td>';
        echo''." ".'';
        echo''." ".'';
        echo'<td><a href="http://localhost/pages/fileEditor.php?fileName='.$file.'" target =
"_new">'."Edit Document".'</a></td>';
        echo'</tr>';
        echo '<br>';
    }
?>
 </tbody>
</table>

</body>
</html>
```

[Back to Top]

fileEditor.php

```
<!-- HTML edited by Paul Court php written by Keshav Singhal -->
<html>
 <meta name="author" content="Keshav Singhal">
 <meta name="description" content="This page allows the user to edit and submit an SRS
document.">

  <head>
        <title>SE 640 Semantic Analysis Project SRS Editor</title>
        <link rel="stylesheet" href="stylesheet.css">
  </head>

<body bgcolor = "#ffffff">
<font size = "+2"> Edit the text and click <b>Submit</b> or <b>Reset</b> when finished!</font>
<br><br>

<?php

// configuration
$FileName= $_GET['fileName'];
$file = "./uploads/$FileName";
$url = "fileEditor.php?fileName=$FileName";
```

```
// check if form has been submitted
if (isset($_POST['text']))
{
    // save the text contents
    file_put_contents($file, $_POST['text']);

    // redirect to form again
    header(sprintf('Location: %s', $url));
    printf('<a href="%s">Moved</a>.', htmlspecialchars($url));
    exit();
}

// read the textfile
$text = file_get_contents($file);

?>

<!-- HTML form -->
        <form action="" method="post">
                <textarea name="text" rows = "40%" cols = "100%"><?php echo
htmlspecialchars($text) ?></textarea>
                <input type="submit" />
                <input type="reset" />
        </form>

</body>
</html>
```

Back to Top

createGraph.php

```
<!-- HTML edited by Paul Court php written by Keshav Singhal -->
<html>
 <meta name="author" content="Paul Court">
 <meta name="description" content="This page allows the user to graph an SRS document.">

  <head>
        <title>SE 640 Semantic Analysis Project Graph Creator</title>
        <link rel="stylesheet" href="stylesheet.css">
  </head>

<body bgcolor = "#ffffff">




<?php

//Creates a pop-up window with the graph of the semantic similarity from the table of
//values given in the dummy file.  The dummy file will have to be replaced with the current
//version of the semantic similarity developed from the algorithm and sent to the
//Graph_LDA_2.0.py file as graph_file_LDA.txt when the link "View Graph" is activated.

// configuration
$FileName= $_GET['fileName'];
$url = "createGraph.php?fileName=$FileName";
$newName = "graph_file_LDA.txt";

//Code to assign each table that was manually created to the graph_file_LDA.txt file that would
//otherwise be created by the code for semantic inspection.  This section should be replaced when
//the implementation is completed.


if ($FileName == "PGCS_raw.txt")
{
                $oldName = "PGCS_Table.txt";
}
elseif ($FileName == "WOW_SRS_Grouped_Edited.txt")
{
                $oldName = "WOW_Table.txt";
}
elseif ($FileName == "WOW_SRS_Grouped.txt")
{
                $oldName = "WOW_Table.txt";
}
else
```

```
{
        $oldName = "originalDummy.txt";
}

copy( $oldName, $newName);

echo "The file passed from previous page is ";
echo $FileName;
echo '<br>';
echo "The semantic similarity table for this file has been saved into ";
echo $newName;
echo '<br>';
echo "The results of the graphical analysis appear below:  ";
echo '<br>';

$output = shell_exec('python Graph_LDA_2.0.py');

//When you exit the graph, the output from the python program is placed in the window for
evaluation.
//The output is a bit messy, but that issue can be resolved in the python file.

echo $output;

?>


</body>
</html>
```

initialPage.html

```
<html>
 <meta name="author" content="Paul Court">
 <meta name="description" content="Opening page">

  <head>
        <title>SE 640 Semantic Analysis Project Opening Page</title>
  </head>

<body bgcolor = "#ffffff">

  <blockquote>
  <font size = "+2" color = "#000000">

  This webpage will read the contents of a requirements document and upload it to a database for
use for the purpose of
  its inspection for semantic similarity.  If changes need to be made in a specific requirement,
this application will
  automatically highlight other similar requirements to help make adjustments in that requirement
as well.
  </font>
  </blockquote>

</body>
</html>
```

Back to Top

graphCreator.php

```
<!-- HTML edited by Paul Court php written by Keshav Singhal -->
<html>
 <meta name="author" content="Keshav Singhal">
 <meta name="description" content="This page will list the uplaoded files available for display
or for edit.">

  <head>
        <title>SE 640 Semantic Analysis Project Available Files</title>
        <link rel="stylesheet" href="stylesheet.css">
  </head>

<body bgcolor = "#ffffff">
<font size = "+2"> Select a file from the list of uploaded documents available.  If empty, choose
<b>Upload Document</b> to add a file.</font>
```

```
<table style="width:100%">
  <tr>
    <td width = "50%"><b>Filename</b></td>
    <td><b>View Graph</b></td>
  </tr>
 <tbody>
<?php
  $files = array_slice(scandir('./uploads/'), 2);
  foreach ($files as $file){
        echo'<tr>';
        echo'<td>'. $file."</td>";
        echo''." ".'';
        echo''." ".'';
        echo'<td><a href="http://localhost/pages/createGraph.php?fileName='.$file.'">'."View
Graph".'</a></td>';
        echo'</tr>';
        echo '<br>';
    }
?>
 </tbody>
</table>

</body>
</html>
```

higlightSimilar.php

```
<!-- HTML edited by Paul Court php written by Keshav Singhal -->
<html>
 <meta name="author" content="Keshav Singhal">
 <meta name="description" content="This page will list the uplaoded files available for display
or for edit.">

  <head>
        <title>SE 640 Semantic Analysis Project Available Files</title>
        <link rel="stylesheet" href="stylesheet.css">
  </head>

<body bgcolor = "#ffffff">

<font size = "+2"> Select a file from the list of available documents to view <b>Highlighted
Similar</b> requirements </font>

<table style="width:100%">
  <tr>
    <td><b>Filename</b></td>
    <td><b>Link to Highlight Similar</b></td>
  </tr>
 <tbody>
<?php
  $files = array_slice(scandir('./uploads/'), 2);
  foreach ($files as $file){
        echo'<tr>';
        echo'<td>'. $file.'</td>';
        echo''." ".'';
        echo''." ".'';
        echo'<td><a
href="http://localhost/pages/displayhighlightedSimilar.php?fileName='.$file.'">'."Highlight
Similar Requirements".'</a></td>';

        echo'</tr>';
        echo '<br>';
    }
?>
 </tbody>
</table>

</body>

</html>
```

Back to Top

displayhighlightedSimilar.php

```
<!-- HTML edited by Paul Court php written by Sharatha Jayakumar -->
<html>
 <meta name="author" content="Sharatha">
 <meta name="description" content="This page allows the user to view the highlighted related
requirements from SRS document.">

  <head>
       <title>SE 640 Semantic Analysis Project Highlight Related</title>
       <link rel="stylesheet" href="stylesheet.css">
  </head>

<body bgcolor = "#ffffff">
<font size = "+2"> Highlighted Related Requirements </font>
<br><br>


<?php
// configuration
// configuration
$FileName= $_GET['fileName'];
$file = "./uploads/$FileName";
$url = "displayhighlightedSimilar.php?fileName=$FileName";

// read the textfile
$text = file_get_contents($file);
$asArr1 = explode( PHP_EOL, file_get_contents("graphTable.txt") );
?>

<!-- HTML form -->
       <form action="" method="post">
       Select the Requirement to Highlight related requirements:
    <select name="country" onchange="this.form.submit()">
    <option value="" disabled selected>--select--</option>
       <?php
       foreach( array_keys($asArr1) as $val1 ){
    echo "<option value=".$val1.">".$val1."</option>";
       }
       ?>
    </select> <br> <br>


<?php
$tableFileName = "";

if ($FileName == "PGCS_raw.txt")
{
               $tableFileName = "PGCS_Table.txt";
}
elseif ($FileName == "WOW_SRS_Grouped_Edited.txt")
{
               $tableFileName = "WOW_Table.txt";
}
elseif ($FileName == "WOW_SRS_Grouped.txt")
{
               $tableFileName = "graphTable.txt";
}
else
{
       $tableFileName = "graphTable.txt";
}

$finalArray11 = array();
$asArr1 = explode( PHP_EOL, file_get_contents($tableFileName) );
$result = array(array());

foreach( $asArr1 as $val1 ){
  $tmp1 = preg_split('/\s+/', $val1);
      if ((isset($result[$tmp1[0]])) and isset($tmp1[1])) {
               array_push($result[$tmp1[0]],$tmp1[1]);

    } else {
        $result[$tmp1[0]] = array();
               array_push($result[$tmp1[0]],$tmp1[1]);
    }
}
```

Back to Top

```
json_encode($result);

       $style = "";

$lines = explode(PHP_EOL, htmlspecialchars($text));

        if(isset($_POST["country"])){

    $style = "style='display:none;'";


       $arrIndex=$_POST["country"];
          $lines = explode(PHP_EOL, htmlspecialchars($text));

               echo "Graph Table Entries : ".json_encode($result[$arrIndex]).'<br><br>';
               $count = 1;
               foreach ( $lines as $string => $value) {
                      $color="#0";
            $newString=$value.'<br>';
            $String='';
//                    echo $count.$result[$arrIndex];
                      if (in_array(strval($count), $result[$arrIndex])){
                             $color="#ff0000ff";
                      }
                      echo '<div class="text"><span style="color:', $color,
'">',$newString,'</span></div>';
                      $count++;
     }

   }
?>
       <div class="greetings" <?php echo $style;?>>
  <p><?php
          $lines = explode(PHP_EOL, htmlspecialchars($text));

               $count = 1;
               foreach ( $lines as $string => $value) {
            $newString=$value.'<br>';
            $String='';
                      echo '<div class="text"><span style="color:', $color,
'">',$newString,'</span></div>';
                      $count++;
     }?>

  </p>
<div>

</form>

</body>
</html>
```

Back to Top

stylesheet.css

```
/* Link styles. */

a:link
{
  background-color: #B62010;
  color: #ffffff;
  padding: 5px 5px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
  font-family:  Arial;
  font-size:  16;
}

a:visited
{
  background-color: #B62010;
  color: #ffffff;
  padding: 5px 5px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
  font-family:  Arial;
  font-size:  16;
}

a:hover
{
  background-color: #999999;
  color: #B62010;
  padding: 5px 5px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
  font-family:  Arial;
  font-size:  16;
}

a:active
{
  background-color: #B62010;
  color: red;
  padding: 5px 5px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
  font-family:  Arial;
  font-size:  16;
}
```

Back to Top

7.3 PYTHON

Graph_LDA_2.0.py

```
import networkx as nx
from networkx.algorithms.community import k_clique_communities
import matplotlib.pyplot as plt
from pprint import pprint  # pretty-printer

graphFile='graph_file_LDA.txt' # This is a graph file that generated semantic similarity using a
model. Refer the attached dummy graph file sent to you in the email
#finalGraph='finalGraph.txt' # Kept there for future expansion
G=nx.Graph()
G2=nx.DiGraph()
#LSA graph
graph_file=open(graphFile,"r")

#the following code converts text values into integer values
theints=[]
for val in graph_file.read().split():
    theints.append(int(val))

graph_file.close()

print(theints)
start_vertext=[]
end_vertex=[]
count=1 #if count=1 then start vertex and if count=2 then end vertex

source=0
while source<len(theints):
    start_vertex=theints[source]
    source=source+1
    end_vertex=theints[source]
    source=source+1
    #print(source," ",start_vertex," ",end_vertex)
    G.add_edge(start_vertex,end_vertex)

nx.draw(G)
plt.show()
pprint(G.edges())
```

```
print('################################################')
print('###################### CENTRALITY ##########################')
pprint(nx.degree_centrality(G))

print('################################################')
print('###################### CLIQUES ##########################')
pprint(list(nx.find_cliques(G)))

#pprint(list(nx.number_of_cliques(G,nodes=None, cliques=None)))

pprint('######################## K-Clique Communities ########################')
pprint(list(k_clique_communities(G,3)))
pprint('######################## K-Core ########################')

source=0
while source<len(theints):
    start_vertex=theints[source]
    source=source+1
    end_vertex=theints[source]
    source=source+1
    #print(source," ",start_vertex," ",end_vertex)
    G2.add_edge(start_vertex,end_vertex)

G2.remove_edges_from(G2.selfloop_edges())

pprint(G2.edges())
pprint(list(nx.k_core(G2,k=5)))
graph_file.close()
```

Back to Top

7.4 DUMMY FILES

originalDummy.txt

```
1 2
2 3
2 4
2 5
4 5
```

WOW_Table.txt

```
1
2       6
2       10
2       15
3       5
3       15
4       10
5       6
6       20
7       16
7       17
7       18
7       19
8       9
8       10
8       20
9       11
9       12
9       20
10      11
10      12
10      14
10      15
10      20
11      12
11      20
12      20
13
14
15
16      17
```

Back to Top

```
16      18
16      19
17      18
17      19
18      19
19
20
```

PGCS_Table.txt

```
1
2       31
3       29
4       33
4       34
4       35
5       9
5       14
5       22
5       28
5       42
6       11
6       15
6       20
6       21
6       22
6       28
6       47
6       48
6       58
7       16
7       24
7       37
7       39
7       40
7       51
7       20
8       9
8       13
8       26
8       34
8       37
9       12
9       18
9       21
9       25
9       28
10      12
10      20
10      25
10      26
10      36
10      51
11      47
12      18
12      20
12      43
12      47
13      24
14      15
15
16      38
17
18      21
18      47
18      48
18      57
19      42
20      21
20      47
20      48
21      47
21      48
21      57
22      49
22      52
23      41
```

Back to Top

```
24
25      36
26
27
28
29
30
31      35
32      33
33
34
35
36
37      38
38      39
38      40
39      40
40      41
41
42      45
42      44
43
44      45
45
46      50
47      48
48      53
49      52
50
51
52      54
52      58
53
54      58
55
56      57
57
58
59
60
61
62
```

Back to Top

8. Summary

The project proved to be challenging. The workload was shared appropriately. Each team member contributed per their overall strengths and time demands. The functionality and designs were thoroughly displayed in this document.

As key functionality absent is the ability to read a requirements document and determine the semantic table of similarity with the strength of similarity noted. The graph created by the python code does not address the labeling of the nodes or the strength of the semantic relationship between the requirements.

Keep in mind by selecting from the Highlight Similar menu, the intent is to view its semantically similar requirements. Necessary changes in the original requirement found at this time have to be made via the Available SRS Documents/Edit Document process. Then the requirements document should be re-evaluated to determine whether the changes shift the nature and strength of its semantic similarities.

Any further explanations can certainly be elicited from any individual from the group, if necessary.