pylinkvalidator
Design Document Module Guide : newAGEtech, Group H

Genevieve Okon (Okong), Abraham Omorogbe(Omorogoa),
Eric Le Forte(Leforte)

November 6, 2015

Contents

List of Figures

List of Tables

Revision History

1 Revision History

Revision	Revision Date	Description of Change	Author
1	3-11-15	Initiate Design Document	Genevieve Okon
2	3-11-15	Defined anticipated and likely changes	Genevieve Okon
3	3-11-15	created Tracability matrix	Abraham Omorogbe
4	3-11-15	created Module Hierarchy	Abraham Omorogbe
5	3-11-15	Proofreading of design document	Genevieve Okon
6	3-11-15	created pert and Grant chart	Genevieve Okon
7	4-11-15	Proofreading and merging of overall content	Geneviieve Okon
8	4-11-15	Use Hierarchy Between Modules	Eric Le Fort
9	4-11-15	Connection Between Requirements and Design	Eric Le Fort
10	5-11-15	created Hardware/behaviour Hiding Modules	Abraham Omorogbe
11	5-11-15	Software Decision Modules	Eric Le Fort
12	5-11-15	Table of Contents	Genevieve Okon

Table 1: Revision History

2 Introduction

The following document details the Module Interface Specifications for the implemented modules in Pylinkvalidator. This will Identify and describe the program modules that need to be built in detail so that developers or viewers can easily understand the program. Navigation through the program will be made easier for design and maintenance purposes. Complementary documents include the System Requirement Specifications.

The rest of the document is organized as follows. Section 2 lists the anticipated and unlikely changes of the software requirements. Section 3 summarizes the module decomposition that was constructed according to the likely changes. Section 4 specifies the connections between the software requirements and the modules. The design will be compared with the requirements provided in the SRS.

3 Anticipated & Likely Changes

Anticipated changes are the source of the information that is to be hidden inside the modules. Ideally, changing one of the anticipated changes will only require changing the one module that hides the associated decision.
AC1: The specific hardware on which the webcrawler is running.
AC2: The format of the initial input data.
AC3: The format of the input parameters.
AC4: The format of the final output data.
AC5: The algorithm used for the pylinkvalidator.
AC6: The implementation of the html parsers.
AC7: How the overall control of the search modules will be made.
AC8: The implementation for the visual version of the structure model

2.2 Unlikely Changes The module design should be as general as possible. However, a general system is more complex. Sometimes this complexity is not necessary. Fixing some design decisions at the system architecture stage can simplify the software design. If these decision should later need to be changed, then many parts of the design will potentially need to be modified. These are few unlikely changes.
UC1: Input/Output devices (Input: File and/or Keyboard, Output: File, Memory, and/or Screen)

UC2: There will always be a source of input data external to the software.
UC3: Output data are displayed to the output device.
UC4: Goal of the system is to crawl websites download links and images.

4 Module Hierarchy

This section contains the module design structure of our project. Modules are summarized in a hierarchy as shown in Table 1. The modules listed below, most of which are leaves in the hierarchy tree, are the modules that will actually be implemented.

Modules

M1: Option Module
M2: Crawler Module
M3: HTML Corrector Module
M4: Download Resources Module
M5: Exact Query Search Module
M6: Similar Query Search Module
M7: Whitespace Checker Module
M8: Find Links Module
M9: Check Errors Module
M10: Parse Data Module
M11: Query Search Module
M12: Depth Setter Module
M13: Website Structure Modelling Module

Note that M5, M6, M7 are submodules of the larger Query Search Module. M2 is the highest level module and utilizes all others internally.

Hardware hiding

N/A

Behaviour Hiding Modules

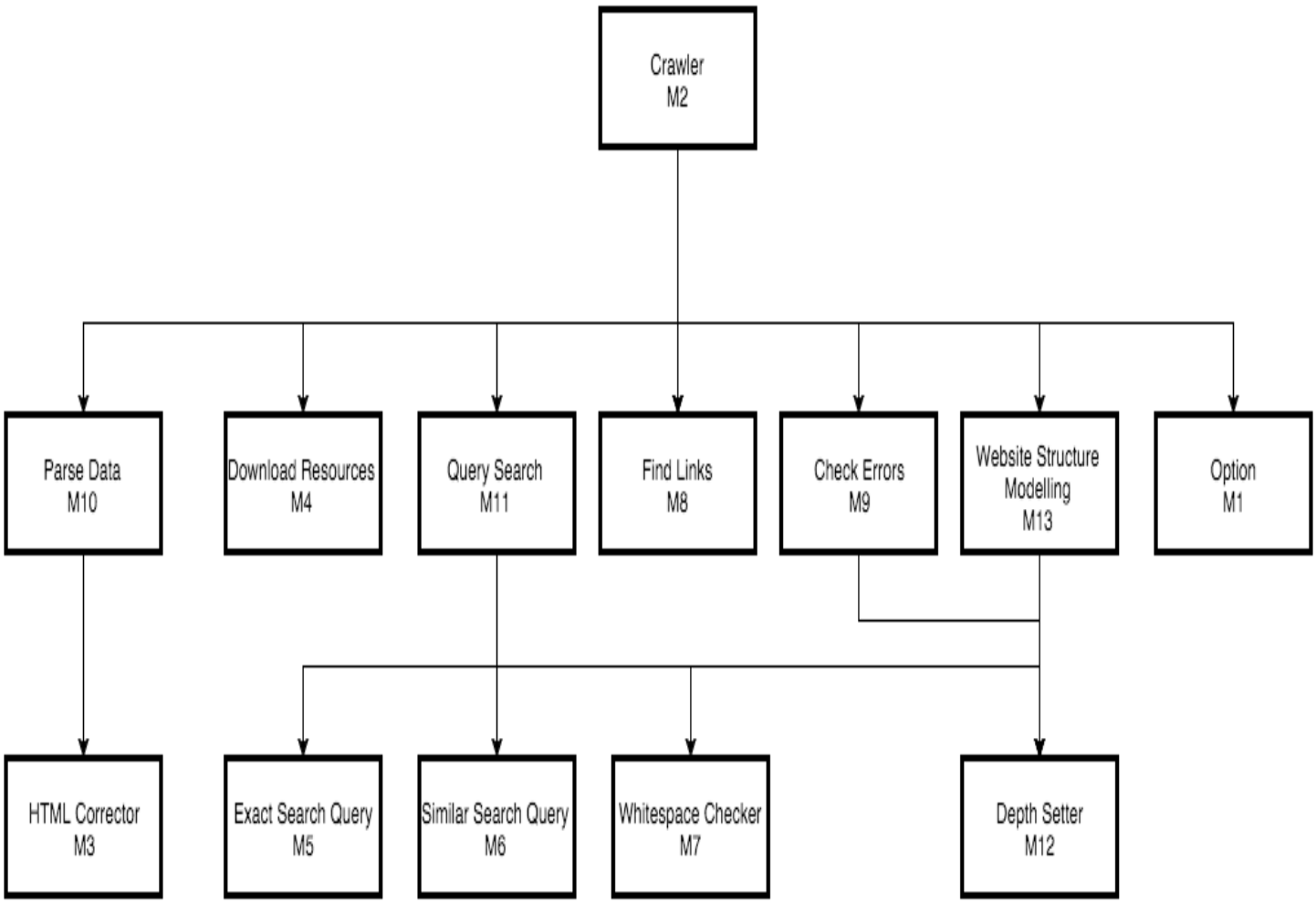
- Download Resources Module
- Exact Query Search Module
- Similar Query Search Module
- Whitespace Checker Module
- Find Links Module
- Check Errors Module
- Parse Data Module

- Query Search Module
- Website Structure Modelling Module

Software Decision Modules

- Option Module
- Crawler Module
- HTML Corrector Module
- Depth Setter Module

5 Module Hierarchy Diagram



6 Connection Between Requirements & Design

This design was developed using the requirements document to help guide the decomposition of the project's modules. The requirements were matched to corresponding modules which complete the various tasks. For example, Requirement #2 from the requirements document (The product shall download resources from a website) will be accomplished using module M4.

7 Module Decomposition

7.1 Hardware Hiding Modules

N/A

7.2 Behaviour Hiding Modules

7.2.1 Download Resources

void downloadResources(String: link, String: fileType, String: destination)

Secrets: Parse through the HTML code in the link provided in order to locate all files that match the specified file type. For each file, the result is downloaded into a folder specified by the user.

Services: Writes all resources matching the given file type from the page link to the file specified by destination.

Implemented By: Python

7.2.2 Exact Query Search

int[] searchForString(String: query, String: data)

Secrets: Iterates through the data provided and records every instance matching the query String that was passed in. This will be accomplished using the Knuth-Morris-Pratt String searching algorithm.

Services: Returns a list of all occurrences of a given query in the data provided.

Implemented By: Python

7.2.3 Similar Query Search

int[] searchForSimilarString(String: query, String: data, int: proximity)

Secrets: Iterates through the data provided and records every instance sufficiently close to matching the query String that was passed in. This will be accomplished using a slight deviation from the Knuth-Morris-Pratt String searching algorithm that recognizes fuzzy string searching.

Services: Returns a list of all occurrences within a certain deviation of a given query in the data provided.

Implemented By: Python

7.2.4 Whitespace Checker

boolean isWhitespace(char: character)

Secrets: Checks to see if the character passed in is a tab, a space or a new line character.

Services: Returns whether the character passed in is certain types of whitespace.

Implemented By: Python

7.2.5 Find Links

List¡Links¿ findLinks(BeautifulSoup: data, String: destination) Exceptions: No Data Found

Secrets: Parser that parsers through the HTML code in the data provided

Services: Finds all the links (<a> anchor tags on page) on a page

Implemented By: Python

7.2.6 Check Errors

List¡Errors¿ checkErrors(String: link, Array: List of links) Exceptions: Webpage Unavailable

Secrets: Algorithm to check the header in all the links provided

Services: Checks the all the links and reports the error message associated with all the links inputed

Implemented By: Python

7.2.7 Parse Data

BeautifulSoup parseData(String: link) Exceptions: Invalid Link

Secrets: Converter that converts HTML code link to BeautifulSoup object

Services: Returns Beautiful object for the link given, this will allow modules parse through pages data much faster

Implemented By: Python

7.2.8 Query Search

void querySearch(String: Query, BeautifulSoup: data, String: choice) Exceptions: Invalid Choice

Environment Variables: rawInput: Users keyboard input

Secrets: An algorithm that figures out what query search type to implement

Services: Writes all resources matching the given file type from the page link to the file specified by destination.

Implemented By: Python

7.2.9 Website Structure Modelling

String webStructureModel(String: link, int: depth)

Secrets: An algorithm that uses the seed link and structures the crawled links by depth

Services: It provides a structured model of the website and other site the initial site is connect to. It displays a hierarchy that will show users how crawled link interact with each other.

Implemented By: Python

7.3 Software Decision Modules

7.3.1 Options

void chooseOption()

Secrets: Obtains input from the user to decide whether they would like to download resources, crawl a website, search for a certain String or model a website. Uses searchDepth to allow the user to set their search depth for all but the download resources option as well as allowing the user to set their seed link.

Services: Gets user input to set various program options related to how the user would like to handle crawling a webpage.

Implemented By: Python

7.3.2 Crawler

void crawler()

Secrets: Directly calls the methods shown in the Use Hierarchy (found in section 7 of this document) and consolidates the results.
Services: Delegates various tasks of crawling a webpage to the other methods of this program.

Implemented By: Python

7.3.3 HTML Corrector

String HTMLCorrector(String: link)

Secrets: Tries to fix the given String in various ways in order to create a functioning link. This can include fixing the prefix (http://), adding www., or appending the link to the current page's path.

Services: Fixes the link passed in such that it becomes either a functioning link or is flagged as a broken link.

Implemented By: Python

7.3.4 Depth Setter

int depthSetter(int depth) Exception: Not positive int

Environment Variables
rawInput: Users keyboard input

Secrets Assignor which sets the depth variable

Services Sets the default max depth variable for the web crawler

Implemented By: Python

8 Traceability Matrix

9 Use Hierarchy Between Modules

The figure below depicts the uses relationships between all the modules in the project. It can be seen that the graph is a directed acyclic graph (DAG). The facade design pattern is being used to design this system. Higher level modules in

Requirements	Modules
R1	M1, M2, M3, M9, M10
R2	M1, M2, M3, M4, M10
R3	M12
R4	M2
R5	M1, M2, M3, M4, M5, M6, M7, M8, M9, M10, M11, M12, M13
R6	M1, M2, M3, M8, M10, M12, M13
R7	M1, M2, M3, M5, M6, M7, M8, M10, M11, M12

Table 2: Traceback to Requirements

Requirements	Modules
AC1	
AC2	

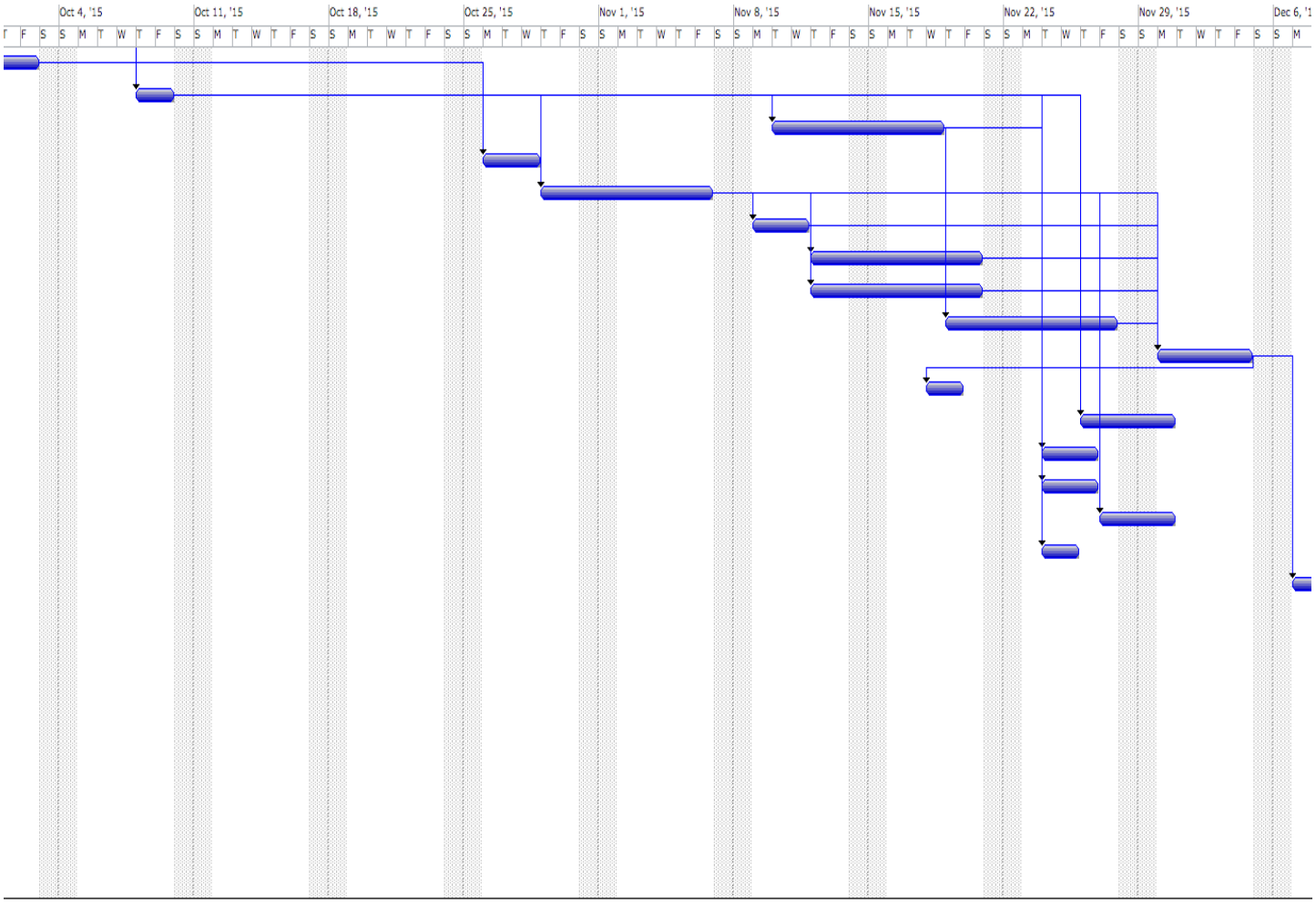
Table 3: Traceback to Anticipated Changes

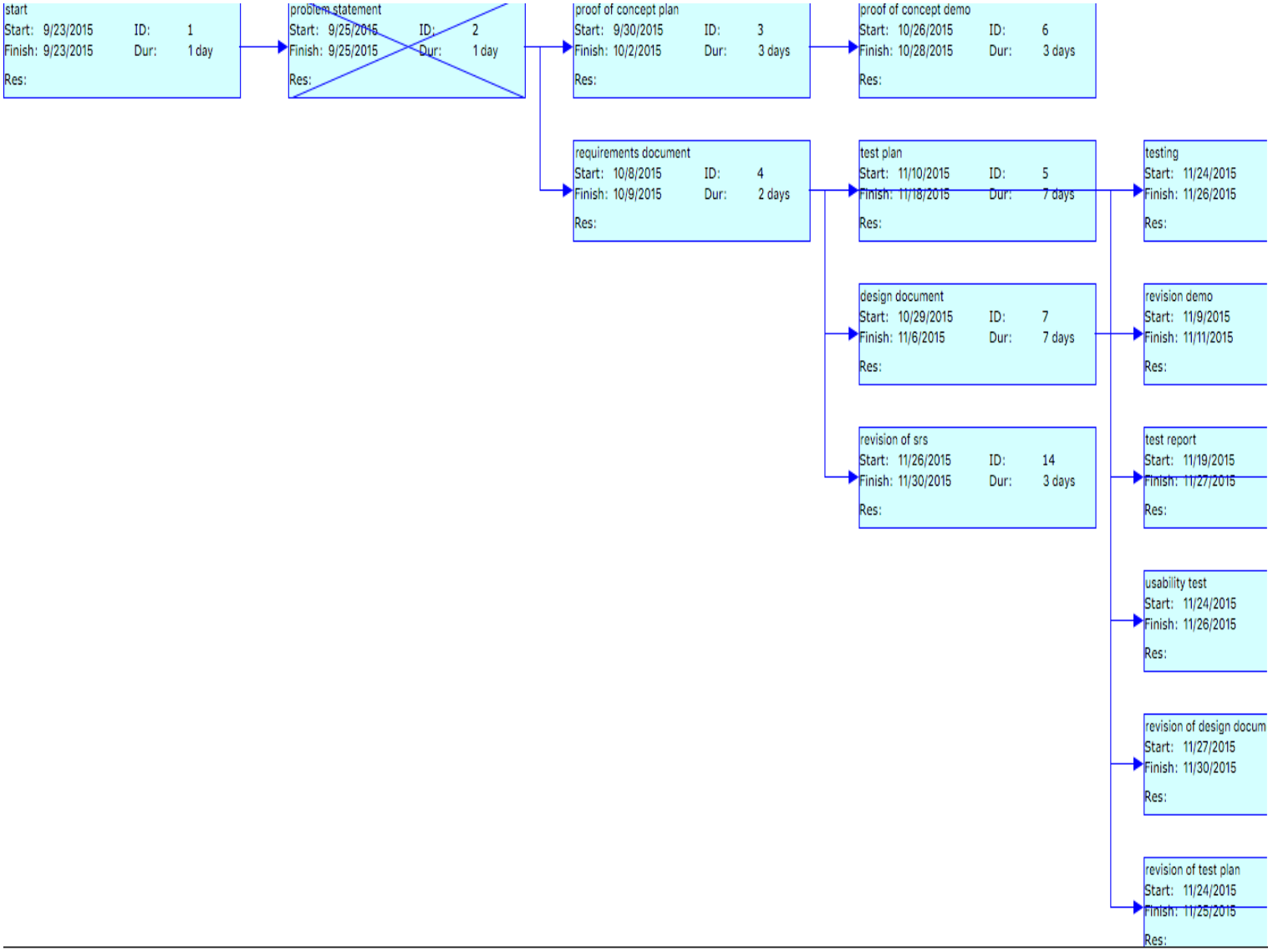
relation to the hierarchy are inherently simpler because they delegate work to modules from the lower levels.

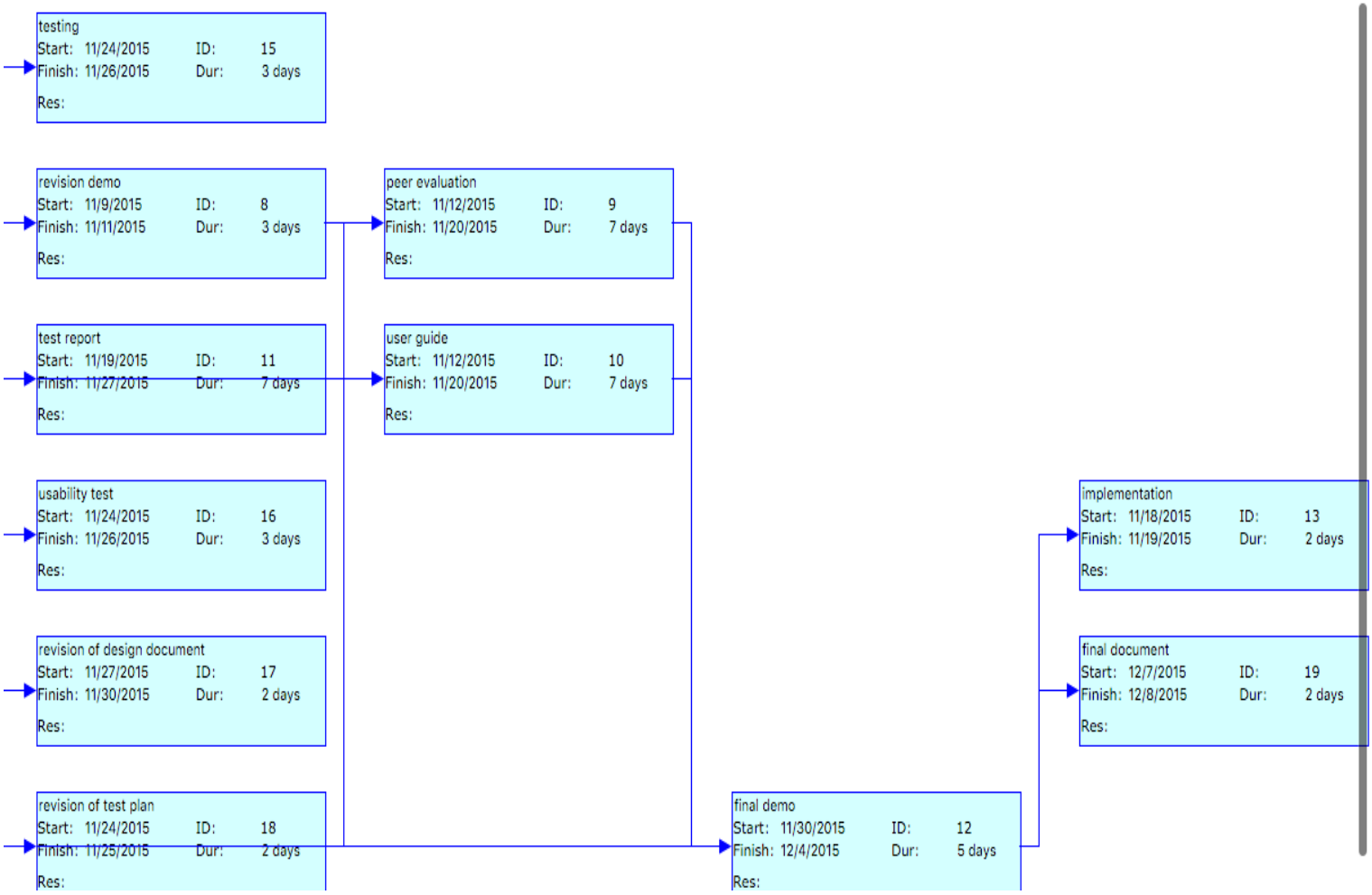
10 Detailed timeline for the rest of the course

	Task Name	Duration	Work	Start	Finish	Predecessors	Resource Names	% Complete
1	start	1 day	0 hrs	9/23/2015	9/23/2015			0%
2	problem statement	1 day	0 hrs	9/25/2015	9/25/2015	1		0%
3	proof of concept plan	3 days	0 hrs	9/30/2015	10/2/2015	2		0%
4	requirements document	2 days	0 hrs	10/8/2015	10/9/2015	2		0%
5	test plan	7 days	0 hrs	11/10/2015	11/18/2015	4		0%
6	proof of concept demo	3 days	0 hrs	10/26/2015	10/28/2015	3		0%
7	design document	7 days	0 hrs	10/29/2015	11/6/2015	4		0%
8	revision demo	3 days	0 hrs	11/9/2015	11/11/2015	7		0%
9	peer evaluation	7 days	0 hrs	11/12/2015	11/20/2015	8		0%
10	user guide	7 days	0 hrs	11/12/2015	11/20/2015	7,8		0%
11	test report	7 days	0 hrs	11/19/2015	11/27/2015	5,7		0%
12	final demo	5 days	0 hrs	11/30/2015	12/4/2015	8,7,9,10,11		0%
13	implementation	2 days	0 hrs	11/18/2015	11/19/2015	12		0%
14	revision of srs	3 days	0 hrs	11/26/2015	11/30/2015	4		0%
15	testing	3 days	0 hrs	11/24/2015	11/26/2015	5,7,4		0%
16	usability test	3 days	0 hrs	11/24/2015	11/26/2015	5		0%
17	revision of design document	2 days	0 hrs	11/27/2015	11/30/2015	7		0%
18	revision of test plan	2 days	0 hrs	11/24/2015	11/25/2015	5		0%
19	final document	2 days	0 hrs	12/7/2015	12/8/2015	12		0%

11 Grant and pert charts for the timeline




start
Start: 9/23/2015 ID: 1
Finish: 9/23/2015 Dur: 1 day
Res:
problem statement
Start: 9/25/2015 ID: 2
Finish: 9/25/2015 Dur: 1 day
Res:
proof of concept plan
Start: 9/30/2015 ID: 3
Finish: 10/2/2015 Dur: 3 days
Res:
proof of concept demo
Start: 10/26/2015 ID: 6
Finish: 10/28/2015 Dur: 3 days
Res:
requirements document
Start: 10/8/2015 ID: 4
Finish: 10/9/2015 Dur: 2 days
Res:
test plan
Start: 11/10/2015 ID: 5
Finish: 11/18/2015 Dur: 7 days
Res:
testing
Start: 11/24/2015
Finish: 11/26/2015
Res:
design document
Start: 10/29/2015 ID: 7
Finish: 11/6/2015 Dur: 7 days
Res:
revision demo
Start: 11/9/2015
Finish: 11/11/2015
Res:
revision of srs
Start: 11/26/2015 ID: 14
Finish: 11/30/2015 Dur: 3 days
Res:
test report
Start: 11/19/2015
Finish: 11/27/2015
Res:
usability test
Start: 11/24/2015
Finish: 11/26/2015
Res:
revision of design docum
Start: 11/27/2015
Finish: 11/30/2015
Res:
revision of test plan
Start: 11/24/2015
Finish: 11/25/2015
Res:


testing
Start: 11/24/2015 ID: 15
Finish: 11/26/2015 Dur: 3 days
Res:
revision demo
Start: 11/9/2015 ID: 8
Finish: 11/11/2015 Dur: 3 days
Res:
peer evaluation
Start: 11/12/2015 ID: 9
Finish: 11/20/2015 Dur: 7 days
Res:
test report
Start: 11/19/2015 ID: 11
Finish: 11/27/2015 Dur: 7 days
Res:
user guide
Start: 11/12/2015 ID: 10
Finish: 11/20/2015 Dur: 7 days
Res:
usability test
Start: 11/24/2015 ID: 16
Finish: 11/26/2015 Dur: 3 days
Res:
implementation
Start: 11/18/2015 ID: 13
Finish: 11/19/2015 Dur: 2 days
Res:
revision of design document
Start: 11/27/2015 ID: 17
Finish: 11/30/2015 Dur: 2 days
Res:
final document
Start: 12/7/2015 ID: 19
Finish: 12/8/2015 Dur: 2 days
Res:
revision of test plan
Start: 11/24/2015 ID: 18
Finish: 11/25/2015 Dur: 2 days
Res:
final demo
Start: 11/30/2015 ID: 12
Finish: 12/4/2015 Dur: 5 days
Res: