CMSC 447

Software Development Plan (SDP)

[1 Scope 6](#_Toc432620797)

[1.1 Identification 6](#_Toc432620798)

[1.2 System overview 6](#_Toc432620799)

[1.3 Document overview 6](#_Toc432620800)

[1.4 Relationship to other plans 6](#_Toc432620801)

[2 Referenced documents 6](#_Toc432620802)

[3 Overview of required work 6](#_Toc432620803)

[4 Plans for performing general software development activities 7](#_Toc432620804)

[4.1 Software development process 7](#_Toc432620805)

[4.2 General plans for software development 7](#_Toc432620806)

[4.2.1 Software development methods 7](#_Toc432620807)

[4.2.2 Standards for software products 7](#_Toc432620808)

[4.2.3 Reusable software products 8](#_Toc432620809)

[4.2.4 Handling of critical requirements 8](#_Toc432620810)

[4.2.5 Computer hardware resource utilization 8](#_Toc432620811)

[4.2.6 Recording rationale 8](#_Toc432620812)

[4.2.7 Access for acquirer review 8](#_Toc432620813)

[5 Plans for performing detailed software development activities 9](#_Toc432620814)

[5.1 Project planning and oversight 9](#_Toc432620815)

[5.1.1 Software development planning (covering updates to this plan) 9](#_Toc432620816)

[5.1.2 CSCI test planning 9](#_Toc432620817)

[5.1.3 System test planning 9](#_Toc432620818)

[5.1.4 Software installation planning 9](#_Toc432620819)

[5.1.5 Software transition planning 9](#_Toc432620820)

[5.1.6 Following and updating plans, including the intervals for management review 9](#_Toc432620821)

[5.2 Establishing a software development environment 9](#_Toc432620822)

[5.2.1 Software engineering environment 9](#_Toc432620823)

[5.2.2 Software test environment 9](#_Toc432620824)

[5.2.3 Software development library 9](#_Toc432620825)

[5.2.4 Software development files 9](#_Toc432620826)

[5.2.5 Non-deliverable software 9](#_Toc432620827)

[5.3 System requirements analysis 9](#_Toc432620828)

[5.3.1 Analysis of user input 10](#_Toc432620829)

[5.3.2 Operational concept 10](#_Toc432620830)

[5.3.3 System requirements 10](#_Toc432620831)

[5.4 System design 10](#_Toc432620832)

[5.4.1 System-wide design decisions 10](#_Toc432620833)

[5.4.2 System architectural design 10](#_Toc432620834)

[5.5 Software requirements analysis 10](#_Toc432620835)

[5.6 Software design 10](#_Toc432620836)

[5.6.1 CSCI-wide design decisions 10](#_Toc432620837)

[5.6.2 CSCI architectural design 10](#_Toc432620838)

[5.6.3 CSCI detailed design 10](#_Toc432620839)

[5.7 Software implementation and unit testing 10](#_Toc432620840)

[5.7.1 Software implementation 10](#_Toc432620841)

[5.7.2 Preparing for unit testing 10](#_Toc432620842)

[5.7.3 Performing unit testing 10](#_Toc432620843)

[5.7.4 Revision and retesting 10](#_Toc432620844)

[5.7.5 Analyzing and recording unit test results 10](#_Toc432620845)

[5.8 Unit integration and testing 11](#_Toc432620846)

[5.8.1 Preparing for unit integration and testing 11](#_Toc432620847)

[5.8.2 Performing unit integration and testing 11](#_Toc432620848)

[5.8.3 Revision and retesting 11](#_Toc432620849)

[5.8.4 Analyzing and recording unit integration and test results 11](#_Toc432620850)

[5.9 CSCI qualification testing 11](#_Toc432620851)

[5.9.1 Independence in CSCI qualification testing 11](#_Toc432620852)

[5.9.2 Testing on the target computer system 11](#_Toc432620853)

[5.9.3 Preparing for CSCI qualification testing 11](#_Toc432620854)

[5.9.4 Dry run of CSCI qualification testing 11](#_Toc432620855)

[5.9.5 Performing CSCI qualification testing 11](#_Toc432620856)

[5.9.6 Revision and retesting 11](#_Toc432620857)

[5.9.7 Analyzing and recording CSCI qualification test results 11](#_Toc432620858)

[5.10 CSCI/HWCI integration and testing 11](#_Toc432620859)

[5.10.1 Preparing for CSCI/HWCI integration and testing 11](#_Toc432620860)

[5.10.2 Performing CSCI/HWCI integration and testing 11](#_Toc432620861)

[5.10.3 Revision and retesting 11](#_Toc432620862)

[5.10.4 Analyzing and recording CSCI/HWCI integration and test results 11](#_Toc432620863)

[5.11 System qualification testing 11](#_Toc432620864)

[5.11.1 Independence in system qualification testing 12](#_Toc432620865)

[5.11.2 Testing on the target computer system 12](#_Toc432620866)

[5.11.3 Preparing for system qualification testing 12](#_Toc432620867)

[5.11.4 Dry run of system qualification testing 12](#_Toc432620868)

[5.11.5 Performing system qualification testing 12](#_Toc432620869)

[5.11.6 Revision and retesting 12](#_Toc432620870)

[5.11.7 Analyzing and recording system qualification test results 12](#_Toc432620871)

[5.12 Preparing for software use 12](#_Toc432620872)

[5.12.1 Preparing the executable software 12](#_Toc432620873)

[5.12.2 Preparing version descriptions for user sites 12](#_Toc432620874)

[5.12.3 Preparing user manuals 12](#_Toc432620875)

[5.12.4 Installation at user sites 12](#_Toc432620876)

[5.13 Preparing for software transition 12](#_Toc432620877)

[5.13.1 Preparing the executable software 12](#_Toc432620878)

[5.13.2 Preparing source files 12](#_Toc432620879)

[5.13.3 Preparing version descriptions for the support site 12](#_Toc432620880)

[5.13.4 Preparing the "as built" CSCI design and other software support information 12](#_Toc432620881)

[5.13.5 Updating the system design description 12](#_Toc432620882)

[5.13.6 Preparing support manuals 12](#_Toc432620883)

[5.13.7 Transition to the designated support site 12](#_Toc432620884)

[5.14 Software configuration management 13](#_Toc432620885)

[5.14.1 Configuration identification 13](#_Toc432620886)

[5.14.2 Configuration control 13](#_Toc432620887)

[5.14.3 Configuration status accounting 13](#_Toc432620888)

[5.14.4 Configuration audits 13](#_Toc432620889)

[5.14.5 Packaging, storage, handling, and delivery 13](#_Toc432620890)

[5.15 Software product evaluation 13](#_Toc432620891)

[5.15.1 In-process and final software product evaluations 13](#_Toc432620892)

[5.15.2 Software product evaluation records, including items to be recorded 13](#_Toc432620893)

[5.15.3 Independence in software product evaluation 13](#_Toc432620894)

[5.16 Software quality assurance 13](#_Toc432620895)

[5.16.1 Software quality assurance evaluations 13](#_Toc432620896)

[5.16.2 Software quality assurance records, including items to be recorded 13](#_Toc432620897)

[5.16.3 Independence in software quality assurance 13](#_Toc432620898)

[5.17 Corrective action 13](#_Toc432620899)

[5.17.1 Problem/change reports, including items to be recorded (candidate items include project name, originator, problem number, problem name, software element or document affected, origination date, category and priority, description, analyst assigned to the problem, date assigned, date completed, analysis time, recom- mended solution, impacts, problem status, approval of solution, follow-up actions, corrector, correction date, version where corrected, correction time, description of solution implemented) 14](#_Toc432620900)

[5.17.2 Corrective action system 14](#_Toc432620901)

[5.18 Joint technical and management reviews 14](#_Toc432620902)

[5.18.1 Joint technical reviews, including a proposed set of reviews 14](#_Toc432620903)

[5.18.2 Joint management reviews, including a proposed set of reviews 14](#_Toc432620904)

[5.19 Other software development activities 14](#_Toc432620905)

[5.19.1 Risk management, including known risks and corresponding strategies 14](#_Toc432620906)

[5.19.2 Software management indicators, including indicators to be used 14](#_Toc432620907)

[5.19.3 Security and privacy 14](#_Toc432620908)

[5.19.4 Subcontractor management 14](#_Toc432620909)

[5.19.5 Interface with software independent verification and validation (IV&V) agents 14](#_Toc432620910)

[5.19.6 Coordination with associate developers 14](#_Toc432620911)

[5.19.7 Improvement of project processes 14](#_Toc432620912)

[5.19.8 Other activities not covered elsewhere in the plan 14](#_Toc432620913)

[6 Schedules and activity network 14](#_Toc432620914)

[7 Project organization and resources 15](#_Toc432620915)

[7.1 Project organization 15](#_Toc432620916)

[7.2 Project resources 15](#_Toc432620917)

[8 Notes 15](#_Toc432620918)

[9 A. Appendixes 15](#_Toc432620919)

# Scope

This section shall be divided into the following paragraphs.

## Identification

This paragraph shall contain a full identification of the system and the software to which this document applies, including, as applicable, identification number(s), title(s), abbreviation(s), version number(s), and release number(s).

This project is to be a web application which when implemented will allow a user to input preferences into a questionnaire and top locations will then be outputted onto a map. When first accessing the webpage, the user will begin with the “login page.” From the login page, a user can access a “create an account page” or after authentication access their “homepage.” The homepage will allow the user to modify their account information from a “modify account page,” as well as allow the user to see the “map” and access the “questionnaire,” which are defined below.

The “questionnaire” refers to the portion of the project where the user inputs these preferences. This will be accessed from the homepage into a “questionnaire page.” The “ratings” will refer to the numerical input from a user in which they determine which statistics have a higher preference. The “statistics” refer to the set of data that our project will be accessing.

The “map” refers to the portion of the project which is the visualization of data. The map is initially displayed from the homepage, but will initially be empty, and be modified as the user takes the questionnaire. After the user completes the questionnaire the map is referenced to be an “updated map” with the continuing definitions being components of the map. The map may also have markers which will be referred to as “pins.” The pins will allow the user to see an image of the location referred to as the “pin image.” In some cases where an image is not available, or if there is additional implementation, a pin may allow the user to see descriptive information on the location which will be referred to as the “pin information.”

## System overview

This paragraph shall briefly state the purpose of the system and the software to which this document applies. It shall describe the general nature of the system and software; summarize the history of system development, operation, and maintenance; identify the project sponsor, acquirer, user, developer, and support agencies; identify current and planned operating sites; and list other relevant documents.

The goal of this project is to create a web application which will allow a user to create an account, and from such account be able to evaluate different statistical values such that a list of locations within the United States will be outputted to the user which correlate to the user input. With the list, a map should be presented with pins that correlate to the location and can then be further expanded to output a location image or description.

A SQL database will be used for maintaining the user accounts. Along with this we expect to be using third party software for our output. This would include the use of GoogleMaps for our map output, and GooglePlaces for our image outputs, and GooglePlace for a description output. Further third-party software will be expanded in this section as seems fit throughout the project.

Because the nature of this project is in the scope of a classroom project, there will not be long term maintenance of the project and will be run on a local machine. Throughout this document the term sponsor will refer to John Winder who is the group’s client throughout the project. Similarly, the group refers to the group of developers working on the project including Matthew Hearn, Aaron Lewis, Alex Rochford, Cathy Poore, Ben Kittner, and Steven Heckman. The project will refer to the software and documentation created for this assignment.

All project development will be done through GitHub and then developer preference for development environments, debuggers, ect. The web portion of the application will be run using an Angular Framework on a localhost.

## Document overview

This paragraph shall summarize the purpose and contents of this document and shall describe any security or privacy considerations associated with its use.

The goal of this document is to give a brief overview of the project, as well as explain the development process of the project group. This will include coding practices including but not limited to file headers, function headers, indentation practices, file names, function names, variable names, development environments, used programming languages. Further explained practices will include design and testing processes.

These practices and to be viewed as fluid and as different concerns are raised between the development group or project client this document may be modified as necessary. For this reason, after approval has been given from the client any modifications to the document will be resubmitted for approval and require permission. In a similar manner, if the client wishes for any modifications these should be expressed either through email or verbally which are then approved upon in the form of meeting minutes from the group. With the goal of transparency in mind any modifications to this document will be explicitly noted and noted in the section below.

1.3a Version Modifications

Ver1.0: Original Document

This includes the paragraph description for each heading which will then be deleted upon approval. Additionally, any empty section is kept for both client and group approval.

Ver 1.1: Updated System Overview

Modified the system overview to correspond with the other documentation to reflect the use of Angular framework and running the application on a local host.

Ver 2.0 Updated after group meeting on 4/23.

Updated the Identification to be standard with the rest of the documentation. Updated the system overview with the use of Google Places. Updated system specifics to meet request in the group meeting on 4/23. Other changes include small grammatical changes.

## Relationship to other plans

This paragraph shall describe the relationship, if any, of the SDP to other project management plans.

None.

# Referenced documents

This section shall list the number, title, revision, and date of all documents referenced in this plan. This section shall also identify the source for all documents not available through normal Government stocking activities.

The following are referenced documents throughout this document and include the common acronym for referencing the document:

SRS- Software Requirements Specification

SDD- Software Design Description

STD- Software Test Description

SUM- Software User Manual

# Overview of required work

This section shall be divided into paragraphs as needed to establish the context for the planning described in later sections. It shall include, as applicable, an overview of:

1. Requirements and constraints on the system and software to be developed

For a list of requirements, see the SRS. There are currently no constraints on the system.

1. Requirements and constraints on project documentation

There are no constraints for the documentation.

1. Position of the project in the system life cycle

As of version 1.0 we are beginning the development life cycle. We are looking into different means of implementation of data visualization, as well as researching data for the project. We have also begun implementing the user account creation and login portion of the project. In terms of documentation we have completed the formation of project requirements and will begin the creation of the project design found in the SRS.

As of version 1.1 we have implemented the basics of the homepage and are beginning to implement the SQL server portion of the project. Also, we have begun the development of the survey/questionnaire portion of the project and will be using data from the Census Bureau. Upon completion of version 1.1 we have completed the SDD.

As of version 2.0 we have a layout for the user account portion of the system. We need to complete the implementation of user accounts with a user avatar. We also need to add the questionnaire portion of the system to the homepage. Finally, we need to implement the location pins. Upon completion of version 2.0 we have completed the STD.

1. The selected program/acquisition strategy or any requirements or constraints on it

There are no program/acquisition strategy or requirement constraints.

1. Requirements and constraints on project schedules and resources

The only defined guideline for dates is the final project presentation which is to be done May 10th. Recommended guidelines for documentation dates were given by Professor Cain. These are expected to be followed by the development team.

1. Other requirements and constraints, such as on project security, privacy, methods, standards, interdependencies in hardware and software development, etc.

There are no security, privacy, standards, or hardware/software interdependence requirements or constraints.

# Plans for performing general software development activities

This section shall be divided into the following paragraphs. Provisions corresponding to non-required activities may be satisfied by the words "Not applicable." If different builds or different software on the project require different planning, these differences shall be noted in the paragraphs. In addition to the content specified below, each paragraph shall identify applicable risks/uncertainties and plans for dealing with them.

## Software development process

This paragraph shall describe the software development process to be used. The planning shall cover all contractual clauses concerning this topic, identifying planned builds, if applicable, their objectives, and the software development activities to be performed in each build.

This project is to be viewed as using a mixture between waterfall and agile development processes. The waterfall portion of the project will best correspond to the documentation portions which are to be completed in a systematic order. However, each document is to be viewed as fluid and modified when necessary. Upon modification after client approval, the client will be notified. All documents will receive a final signature upon completion of the project.

The software coding portions of the project are to be viewed with an agile development process such that code should be developed in a manner that is flexible and compartmentalized allowing each software developer the flexibility of coding the portions where they are most experienced. Because of the software development flexibility, the documentation portions of the project will need to be updated as new tools or methods are included in the software.

## General plans for software development

This paragraph shall be divided into the following subparagraphs.

### Software development methods

This paragraph shall describe or reference the software development methods to be used. Included shall be descriptions of the manual and automated tools and procedures to be used in support of these methods. The methods shall cover all contractual clauses concerning this topic. Reference may be made to other paragraphs in this plan if the methods are better described in context with the activities to which they will be applied.

There is great flexibility in the software development methods to be used by the group. As such, this portion will be expanded as new software tools and methods are used. Currently, we expect to be using a SQL database for holding user login data, and python for the back-end data analytics and file management. All code and documentation are to be committed and pushed to the GitHub master repository. With the continued goal of transparency any scripts and similar assisting projects are also be included within the GitHub repository for the use of other group members, as well as for viewing by the client.

### Standards for software products

This paragraph shall describe or reference the standards to be followed for representing requirements, design, code, test cases, test procedures, and test results. The standards shall cover all contractual clauses concerning this topic. Reference may be made to other paragraphs in this plan if the standards are better described in context with the activities to which they will be applied. Standards for code shall be provided for each programming language to be used. They shall include at a minimum:

* + - 1. Standards for format (such as indentation, spacing, capitalization, and order of information)

This section deals with the standard of code formatting. It is to be noted that the goal of readability is to be of greater importance than holding to these standards. If there is deviation from the standards a note within the comments should be given with explanation if necessary. In general, all indentation should be done with by using standard 8-character tabs.

In general, single line spacing is to be used for smaller sections of code. If a code portion is more than 5 lines, then there should be a line separating the beginning and end of the segment. A segment is defined as a portion of code that manipulates the same data. This would be typically occurring during a function definition and loop structures. Again, developer judgement should be used for evaluating if deviating from the defined standard would create more readable code.

* + - 1. Standards for header comments (requiring, for example, name/identifier of the code; version identification; modification history; purpose; requirements and design decisions implemented; notes on the processing (such as algorithms used, assumptions, constraints, limitations, and side effects); and notes on the data (inputs, outputs, variables, data structures, etc.)

Header comments should be included at the beginning of any software file, as well as prior to any new function or method. File headers should include the name of the file, the original creator of the file, the date the file was last modified, as well as a brief description of the functionality within the file.

Function headers should include the name of the function creator (if different than the file creator), requirements that the function intends to be satisfying, test associated with the function, inputs to the function, expected outputs of the function, any error handling being done by the function, and a brief description of the function if not satisfied by the inputs and outputs.

Furthermore, if code is being submitted to GitHub with known bugs, the bugs should be described as the ending of the file or function header. Each header must also include a section for another team member to initial, which will serve to represent a code review.

* + - 1. Standards for other comments (such as required number and content expectations)

Because all code is to include file and function headers with the name of the code creator, when modifying another group member’s code, a comment should be included either on the line of the modification, or for larger segments of code at the beginning and ending of the code segments with the modifiers’ initials. Each segment of code is expected to have a comment explaining what is occurring within the code segment. Furthermore, if a portion of code is more complex a comment should be included. This could include a URL for referenced website, or an explanation for what is being done.

* + - 1. Naming conventions for variables, parameters, packages, procedures, files, etc.

All naming for files, function, and variables should follow the following standards. A filename should be in the format of “NameOfDocument.ext” such that each word begins with a capitalized letter, including the first. Filenames should allow a not working individual to assume the role of that file from the filename. A function name should take a format of “functionName” such that the first letter is always lowercase, and all other words of the statement begin with a capital letter. Like other names, the function name should demonstrate the purpose of the function. A function name that cannot follow this should most likely be broken into multiple functions for readability. Finally, variable names should be the format of “variable\_name” such that everything is lowercase, and words are separated using an underscore. Variable names should be relevant to the scope of the function they are being implemented within and should thus be different than the name of a global or larger scope variable being passed into a function. For example, if an array is being passed into a modification function, then the global name “array” should not be used within the function, and instead use a name such as “arr” within the function scope. This intends to avoid naming errors, as well as maintain readability and obvious function scope differences within functions. Like the other portions, if a different naming scheme allows for better readability a note within the comments should be provided.

* + - 1. Restrictions, if any, on the use of programming language constructs or features

Currently there are no constraints on programming language.

* + - 1. Restrictions, if any, on the complexity of code aggregates

Currently there are no restrictions on code aggregates.

### Reusable software products

This paragraph shall be divided into the following subparagraphs.

#### Incorporating reusable software products

This paragraph shall describe the approach to be followed for identifying, evaluating, and incorporating reusable software products, including the scope of the search for such products and the criteria to be used for their evaluation. It shall cover all contractual clauses concerning this topic. Candidate or selected reusable software products known at the time this plan is prepared or updated shall be identified and described, together with benefits, drawbacks, and restrictions, as applicable, associated with their use.

We expect to be able to reuse portions of the login authentication. Logging in is a standard practice on many web platforms. When evaluating the login portions of the system, there will be checks to ensure that the procedures done in the code are standard. Also, any reused code should be free to use, and preferably opensource.

#### Developing reusable software products

This paragraph shall describe the approach to be followed for identifying, evaluating, and reporting opportunities for developing reusable software products. It shall cover all contractual clauses concerning this topic.

Currently there are no requirements for creating reusable software products. However, as a best practice, all code should be written in an easily reusable manner. This would be most applicable to practices of file, function, and variable naming. Similarly, functionality should be expanded such that each function only accomplishes a single task, such that each function could be individually reused. Finally, it is expected that functions do not manipulate an original data set, and instead work on a copy of the data set such that if a process were to be reused there would not be unintentional data manipulation. When this practice cannot be met, for example with file I/O, a note should be made within the comments simply stating that there is access to the original data set.

### Handling of critical requirements

This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for handling requirements designated critical. The planning in each subparagraph shall cover all contractual clauses concerning the identified topic.

#### Safety assurance

There are currently no safety requirements associated with this project.

#### Security assurance

There are currently no security requirements associated with this project.

#### Privacy assurance

There are currently no privacy requirements associated with this project.

#### Assurance of other critical requirements

There are no other critical assurance requirements. For a list of other requirements see the SRS.

### Computer hardware resource utilization

This paragraph shall describe the approach to be followed for allocating computer hardware resources and monitoring their utilization. It shall cover all contractual clauses concerning this topic.

There are currently no hardware resources that would need to be monitored for the use of the project.

### Recording rationale

This paragraph shall describe the approach to be followed for recording rationale that will be useful to the support agency for key decisions made on the project. It shall interpret the term "key decisions" for the project and state where the rationale is to be recorded. It shall cover all contractual clauses concerning this topic.

There are currently no explicit measures in place for recording rationale. It is expected that as occurrences arise they be documented with email correspondence, and when verbally expressed during a meeting, meeting minutes should be sent out within 48 hours of the meeting and reviewed by all group members and the client within the next 48 hours. If this cannot be done a simple response noting that the minutes were received with an expected timeframe of when the minutes will be read should be stated. This should be sent within the group within the scope of the group members.

### Access for acquirer review

This paragraph shall describe the approach to be followed for providing the acquirer or its authorized representative access to developer and subcontractor facilities for review of software products and activities. It shall cover all contractual clauses concerning this topic.

The client should be granted read-only access to the group’s GitHub repository with the goal of being able to review any code. Any notes on the code should be sent to the group in the form of an email message or be verbally expressed in the next group meeting which should then follow the normal meeting minute procedure.

# Plans for performing detailed software development activities

This section shall be divided into the following paragraphs. Provisions corresponding to non-required activities may be satisfied by the words "Not applicable." If different builds or different software on the project require different planning, these differences shall be noted in the paragraphs. The discussion of each activity shall include the approach (methods/procedures/tools) to be applied to: 1) the analysis or other technical tasks involved, 2) the recording of results, and 3) the preparation of associated deliverables, if applicable. The discussion shall also identify applicable risks/uncertainties and plans for dealing with them. Reference may be made to [4.2.1](#_bookmark0) if applicable methods are described there.

## Project planning and oversight

This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for project planning and oversight. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

### Software development planning (covering updates to this plan)

To plan for development a completion of SDD will be required, as well as review from the development team. This should include descriptions of the database setup, code that will be reused, as well as any continued research that needs to be done. All system decisions should be discussed within the team, and then sent to the client for approval.

### CSCI test planning

CSCI planning should be found within the SDD. This should include an outline of what the different CSCI components will be and basic descriptions of how they will be developed. CSCI decisions should be made by the developer(s) working on them and then shared with the development team for approval. This will then be sent to the client through the documentation. The CSCI components should also be designed with the testing portion of the code in mind which can be found in the STD.

### System test planning

Details of the testing of the system can be found within the STD. All CSCI components should be designed with the goal of unit and integration testing in mind. Upon completion of all CSCI components then system testing should be done. All system test should be done as a cross reference to the requirements described in the SRS.

### Software installation planning

This project is not intended to be installed on a device outside of the development group. However, to ease group development and testing of the system, there should be README files throughout each the project describing any setup to run a piece of code. These files should be included within their respective directories, with the inclusion of a system-wide README in the home directory.

### Software transition planning

This system will not need any transition planning since the system is entirely software.

### Following and updating plans, including the intervals for management review

Periodic reviews should be done with the development team and the client. The development team is expected to meet once a week to discuss current completion of the system as well as any development issues. Furthermore, the development team should include progress updates to the client. These should include notes of any completed CSCI components or documents. Finally, in person meetings should occur at least three times through the entirety of the project including at least one meeting to discuss requirement, one meeting as a prototype test, and one meeting for an acceptance review and document signing.

## Establishing a software development environment

This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for establishing, controlling, and maintaining a software development environment. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

### Software engineering environment

Currently there are no requirements focused on the development environments and usable libraries. The development team has chosen to use the Angular framework for the web development. Also, an Oracle MYSQL server will be used for holding the database. Each developer may use the development environment that they choose, assuming the code works within the testing environment.

### Software test environment

The requirements for this system state that the web application will be run on a local host. For this reason, a README file should be included for what software is needed to setup the environment. This should be included within the system’s home directory. Furthermore, all automation test should be included within the same test file, so that all test may be run in a single automated test.

### Software development library

All files should be included within the GitHub master directory. This will serve as the location to store all code and documentation. The home directory should be broken down into categories of frontend, backend, documentation, and meeting minutes.

### Software development files

All developed files should be included within the GitHub master directory, and furthermore included in there appropriate subdirectory. The frontend directory should include all code, documentation, and README files corresponding to the web application of the system. The backend directory should include all code, documentation, and README files corresponding to the data analysis portion of the application. The documentation directory should include any files which will be included in the final deliverable binder to be signed by the client. The meeting minutes directory should include any files documenting meeting that took place and should be sent to the client upon there completion.

### Non-deliverable software

All software will be delivered with the final project.

## System requirements analysis

This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for participating in system requirements analysis. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

### Analysis of user input

User input should be mostly evaluated within the user login portion of the code and the questionnaire portion of the system. This will include validation of appropriate username and password usage within the login portion of the system. The questionnaire portion of the system will need to validate that the user has entered appropriate input so that any backend portions of the system may perform their appropriate calculations. This is further expanded in the SRS.

### Operational concept

Currently there are no requirements associated with the operational concept outside of the project being a web application.

### System requirements

Further requirements can be found in the SRS.

## System design

This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for participating in system design. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

### System-wide design decisions

There are currently no requirements or contractual clauses related to system-wide design decisions. This has been left to the development team to do as they choose fit. Current system wide design system is primarily related to the use of a database for the user login portion of the system. The user account is shared through multiple CSCI components and will need to be accessible throughout the system. Further information related to the system wide design can be found within the SDD

### System architectural design

There are currently no requirements or contractual clauses related to system architectural design. Each CSCI component will have a description found within the SDD describing the components implementation. The design decisions for each CSCI component are left to the developer(s) of that component, which will then be approved by the development team.

## Software requirements analysis

This paragraph shall describe the approach to be followed for software requirements analysis. The approach shall cover all contractual clauses concerning this topic.

The software requirements may be found within the SRS. All requirements will be reviewed by the development team to ensure feasibility and then the SRS will be sent to the client for approval. If the client does not provide explicit requirements that the development team will create the set of requirements for the system prior to writing the SRS and will seek approval for the requirement list. These will then serve as the set of requirements for the SRS.

## Software design

This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for software design. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

### CSCI-wide design decisions

CSCI wide decisions will be discussed by the development team prior to implementation. After the development team has decided, then these decisions will be documented within the SDD which will be sent to the client for review.

### CSCI architectural design

CSCI architectural decisions will be discussed by the developer(s) of the CSCI component prior to implementation. These decisions will be delivered to the development team, and if approval is given implementation of the component may begin. These decisions will be documented within the SDD which will be sent to the client for review.

### CSCI detailed design

CSCI detailed decisions will be determined by the developer(s) of the CSCI component and documented within the SDD if the decision affects any architectural system. Documentation of details within a CSCI component should be documented within the comments of the code. If modifications are made to the SDD this will be sent to the client for review. If the documentation is limited to the comments, then review will be done during code review within the development team.

## Software implementation and unit testing

This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for software implementation and unit testing. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

### Software implementation

All software is intended to be developed in a manner that will allow for unit testing, and as such each function should be limited, and complete a single testable task. Furthermore, a unit test should be written for each developed function. Unit test should be automated.

### Preparing for unit testing

Following the above procedure, functions should be easily testable. Similarly, each developer is expected to create their own unit test as they develop code segments. These tests should be included within a test file in their respective directory. The test should be callable from a single command line argument found within the directories README file.

### Performing unit testing

Upon completion of each code segment it is expected that the portion of code is to be unit tested. The unit test should be included within a test file found in each directory. Using the command line arguments found within the README, the test should be performed. Upon successful unit tests, the command to run the directories unit test should be included in above directories tests. This should continue up until the system unit test file is reached.

### Revision and retesting

When a code segment is modified, unit testing shall be performed again. Similarly, if there is failed integration testing, a test for the failure should be developed.

### Analyzing and recording unit test results

All unit testing should present some type of visualization for the developer demonstrating that the code portion has “passed” the unit test. As the code segment becomes more well developed then a single output can be used for the accumulation of unit tests for an extended portion of code.

## Unit integration and testing

This paragraph shall be divided into the following sub- paragraphs to describe the approach to be followed for unit integration and testing. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

### Preparing for unit integration and testing

All code should be developed within the context of what they will be integrated into, and as though of a portion of the entire system. Integration testing serves to test the macrocode portions. Ignoration testing should be done if all unit tests for the code segment are passed. Furthermore, it is expected that all code units will then be a part of the CSCI and should be developed within the CSCI context.

### Performing unit integration and testing

Integration testing should begin once all the components of a code segment pass the individual unit test. The integration test should test all possible “routes” of the code and would be most associated with the loop structures of the code. In the context of a web application the integration testing should check the links between pages as well.

### Revision and retesting

Integration testing should check that previous unit tests are not being failed, as well as test the overall system. Failed unit tests should be documented with the developer of the failed portion. If a unit test for a different system is being broken by a different CSCI component, then the new component should be inspected. In the case of a poor unit test for the previously developed component, the development team will collaborate on which component should be redesigned.

### Analyzing and recording unit integration and test results

All integration testing should present some type of visualization for the developer demonstrating that the code portion has “passed” the integration test. As the code segment becomes more well developed then a single output can be used for the accumulation of integration tests for an extended portion of code

## CSCI qualification testing

This paragraph shall be divided into the following sub- paragraphs to describe the approach to be followed for CSCI qualification testing. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

### Independence in CSCI qualification testing

CSCI testing is intended to be primarily done by the development team in a manner that states overall success or failure of the CSCI component. Successful CSCI testing would be able to be publicly released and used in a manner that would demonstrate new usability within the web application. For the release, acceptance testing with the client would be required.

### Testing on the target computer system

Since the application will be locally run, all tests will be done in a local web environment. There is no “target computer” for this system.

### Preparing for CSCI qualification testing

CSCI qualification testing should begin to be approached in integration testing and should be the overall goal of the software component. Successful tests should test all possible code paths, as well as handle user input errors.

### Dry run of CSCI qualification testing

All CSCI qualification testing is expected to be done by the software development group prior to testing with the client. These test results should be known prior to product demonstrations with the intention of limiting portions of the code that are not currently implemented.

### Performing CSCI qualification testing

CSCI qualification testing should be done upon completion of an individual CSCI component. Upon a successful dry-run, a meeting should be planned with the client to demonstrate the functionality which should then receive approval or disapproval. Completion of a CSCI allows for the next development cycle for creating new functionality within the application.

### Revision and retesting

Upon the development of a new CSCI component, the previous test should be checked to ensure that previous tests do not fail. This would similarly be tested within the dry-run as well as client demonstrations.

### Analyzing and recording CSCI qualification test results

Each CSCI should be associated with at least one requirement on the SRS. The specific requirement to CSCI association will be recorded in the requirements table of the SDD document.

## CSCI/HWCI integration and testing

This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for participating in CSCI/HWCI integration and testing. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

### Preparing for CSCI/HWCI integration and testing

CSCI integration testing should be done upon completion of the unit testing of a CSCI component. Integration testing should include the unit test of the other CSCI components. To do this each CSCI component should include a single automated test file. Instructions for how to run the file from the command line should be included in a separate README file.

### Performing CSCI/HWCI integration and testing

CSCI integration testing should require simply calling the overall systems test script. This should be found within the home directory of the system. As CSCI components complete their CSCI testing, then the system test script should be updated with what script to call to test the CSCI component. Overall system integration test should be run from the home directory.

### Revision and retesting

Upon failure of a CSCI component or system test within the CSCI integration test, there should be a review on failing CSCI component. The development group will then collaborate on how to redesign the CSCI component.

### Analyzing and recording CSCI/HWCI integration and test results

Assuming the test for the smaller component of the system have been correctly added to the CSCI integration test, then the CSCI integration results should just be pass/or fail print statements. The integration test should simply be the done via checking these print statements.

## System qualification testing

This paragraph shall be divided into the following sub- paragraphs to describe the approach to be followed for participating in system qualification testing. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

### Independence in system qualification testing

Upon completion of each CSCI component, previous testing should be done to ensure the overall system is not being modified. To ensure independence, all requirements will be written in a format for ease in evaluation. Meeting the given requirements would be equivalent to passing the system qualification testing.

### Testing on the target computer system

This will be run on a local environment.

### Preparing for system qualification testing

The system test should be formatted such that each test meets a specific requirement listed within the SRS. A successful system qualification test would represent a successful system. This would include meeting all requirements.

### Dry run of system qualification testing

All system tests are expected to be run by the group prior to a client meeting with documentation on what requirements are currently being met and which requirements currently need to be met.

### Performing system qualification testing

System qualification tests are expected to be done with the client present, and upon successful tests, the final signatures should be completed on the SRS documentation.

### Revision and retesting

If a portion of the system is modified after system testing, then the SRS document should be modified with the new testing date as well as a description of what modifications are being made to the system.

### Analyzing and recording system qualification test results

All system tests should be recorded within the SRS table.

## Preparing for software use

This paragraph shall be divided into the following sub- paragraphs to describe the approach to be followed for preparing for software use. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

### Preparing the executable software

All software will be locally run and therefore executable software will be locally owned and executed. There should be no need to prepare executable software outside of the CSCI and system testing.

### Preparing version descriptions for user sites

Version descriptions will be maintained within the SUM. System version number will be kept in a format of system version primary(#).secondary(#). A new primary index will indicate a new CSCI component. A secondary index will indicate a modification to a CSCI component.

### Preparing user manuals

The user manual will be maintained in the SUM. The SUM should be made using the README files which should be documented as the development team use new outside software. This will include descriptions for how to setup the environment and a dependency list.

### Installation at user sites

There should be no need to install at a user site since the project will be run locally.

## Preparing for software transition

This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for preparing for software transition. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

### Preparing the executable software

### Preparing source files

### Preparing version descriptions for the support site

### Preparing the "as built" CSCI design and other software support information

### Updating the system design description

### Preparing support manuals

### Transition to the designated support site

No further testing outside of the CSCI and system testing since the project will be locally run.

## Software configuration management

This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for software configuration management. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

### Configuration identification

### Configuration control

### Configuration status accounting

### Configuration audits

### Packaging, storage, handling, and delivery

Each version should be associated with completion of a CSCI element. Due to the scope of the project, configuration control should not be needed.

## Software product evaluation

This paragraph shall be divided into the following sub- paragraphs to describe the approach to be followed for software product evaluation. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

### In-process and final software product evaluations

In-process product evaluation should be done upon completion of each CSCI component and receive a signature within the SRS upon successful demonstration. The final software product is to be completed on 5/10 within the classroom environment. All tests should be done against the requirements laid out in the SRS.

### Software product evaluation records, including items to be recorded

Product evaluation should be completed with CSCI and system testing, and any modification or bugs should be reported and documented within the context of each test.

### Independence in software product evaluation

The software product is to be evaluated for approval by the client to avoid development group influence.

## Software quality assurance

This paragraph shall be divided into the following sub- paragraphs to describe the approach to be followed for software quality assurance. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

### Software quality assurance evaluations

Software quality assurance should be evaluated with each CSCI and system demonstration.

### Software quality assurance records, including items to be recorded

To maintain software quality the group intends to use practices of code review as well as the use of unit, integration, CSCI, and system testing.

### Independence in software quality assurance

Quality assurance will be independent using code review, and then final approval from the client.

## Corrective action

This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for corrective action. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

### Problem/change reports, including items to be recorded (candidate items include project name, originator, problem number, problem name, software element or document affected, origination date, category and priority, description, analyst assigned to the problem, date assigned, date completed, analysis time, recom- mended solution, impacts, problem status, approval of solution, follow-up actions, corrector, correction date, version where corrected, correction time, description of solution implemented)

All product bug reports are intended to be documented within JIRA (assuming this is setup by the University.) If this is not done, then a single document will be created and kept in the GitHub repository for bug reporting.

### Corrective action system

-There is no corrective action system for this project.

## Joint technical and management reviews

This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for joint technical and management reviews. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

### Joint technical reviews, including a proposed set of reviews

### Joint management reviews, including a proposed set of reviews

The scope of technical reviews will be simple initials from code reviews. Each unit should be independently reviewed by another group member which will be kept in the header section of individual units. There will be no management reviews for this project since there is no management.

## Other software development activities

This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for other software development activities. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

### Risk management, including known risks and corresponding strategies

### Software management indicators, including indicators to be used

### Security and privacy

### Subcontractor management

### Interface with software independent verification and validation (IV&V) agents

### Coordination with associate developers

### Improvement of project processes

### Other activities not covered elsewhere in the plan

As difference risk are presented within the project this section will be further developed.

# Schedules and activity network

This section shall present:

1. Schedule(s) identifying the activities in each build and showing initiation of each activity, availability of draft and final deliverables and other milestones, and completion of each activity
2. An activity network, depicting sequential relationships and dependencies among activities and identifying those activities that impose the greatest time restrictions on the project

This section will be further expanded once builds are completed. This may further be represented on JIRA if setup.

# Project organization and resources

This section shall be divided into the following paragraphs to describe the project organization and resources to be applied in each build.

## Project organization

This paragraph shall describe the organizational structure to be used on the project, including the organizations involved, their relationships to one another, and the authority and responsibility of each organization for carrying out required activities.

## Project resources

This paragraph shall describe the resources to be applied to the project. It shall include, as applicable:

1. Personnel resources, including:
   1. The estimated staff-loading for the project (number of personnel over time)

Our group has 6 members.

* 1. The breakdown of the staff-loading numbers by responsibility (for example, management, software engineering, software testing, software configuration management, software product evaluation, software quality assurance)

Each member is expected to participate in the documentation, software development, unit testing, integration testing, system testing, and code review processes.

* 1. A breakdown of the skill levels, geographic locations, and security clearances of personnel performing each responsibility

We are all college students at UMBC with no need for security clearances for this project. Each member is expected to perform all portions of the project. For transparency, each document will include the name of the developer for that code segment.

1. Overview of developer facilities to be used, including geographic locations in which the work will be performed, facilities to be used, and secure areas and other features of the facilities as applicable to the contracted effort.

Development is expected to be done by the group where they please which could include but is not limited to the UMBC campus, personal living spaces, or any public location.

1. Acquirer-furnished equipment, software, services, documentation, data, and facilities required for the contracted effort. A schedule detailing when these items will be needed shall also be included.

None.

1. Other required resources, including a plan for obtaining the resources, dates needed, and availability of each resource item.

None.

# Notes

This section shall contain any general information that aids in understanding this document (e.g., background information, glossary, rationale). This section shall include an alphabetical listing of all acronyms, abbreviations, and their meanings as used in this document and a list of any terms and definitions needed to understand this document.

# Appendixes

Appendixes may be used to provide information published separately for convenience in document maintenance (e.g., charts, classified data). As applicable, each appendix shall be referenced in the main body of the document where the data would normally have been provided. Appendixes may be bound as separate documents for ease in handling. Appendixes shall be lettered alphabetically (A, B, etc.).

DESCRIPTION/PURPOSE

The Software Development Plan (SDP) describes a developer’s plans for conducting a software development effort. The term "software development" is meant to include new development, modification, reuse, reengineering, maintenance, and all other activities resulting in software products.

The SDP provides the acquirer insight into, and a tool for monitoring, the processes to be followed for software development, the methods to be used, the approach to be followed for each activity, and project schedules, organization, and resources.

APPLICATION/INTERRELATIONSHIP

Portions of this plan may be bound separately if this approach enhances their usability. Examples include plans for software configuration management and software quality assurance.

The Contract Data Requirements List (CDRL) should specify whether deliverable data are to be delivered on paper or electronic media; are to be in a given electronic form (such as ASCII, CALS, or compatible with a specified word processor or other support software); may be delivered in developer format rather than in the format specified herein; and may reside in a computer-aided software engineering (CASE) or other automated tool rather than in the form of a traditional document.

PREPARATION INSTRUCTIONS

General instructions.

a. Automated techniques. Use of automated techniques is encouraged. The term "document" in this means a collection of data regardless of its medium.

b. Alternate presentation styles. Diagrams, tables, matrices, and other presentation styles are acceptable substitutes for text when data required can be made more readable using these styles.

c. Title page or identifier. The document shall include a title page containing, as applicable: document number; volume number; version/revision indicator; security markings or other restrictions on the handling of the document; date; document title; name, abbreviation, and any other identifier for the system, subsystem, or item to which the document applies; contract number; CDRL item number; organization for which the document has been prepared; name and address of the preparing organization; and distribution statement. For data in a database or other alternative form, this information shall be included on external and internal labels or by equivalent identification methods.

d. Table of contents. The document shall contain a table of contents providing the number, title, and page number of each titled paragraph, figure, table, and appendix. For data in a database or other alternative form, this information shall consist of an internal or external table of contents containing pointers to, or instructions for accessing, each paragraph, figure, table, and appendix or their equivalents.

e. Page numbering/labeling. Each page shall contain a unique page number and display the document number, including version, volume, and date, as applicable. For data in a database or other alternative form, files, screens, or other entities shall be assigned names or numbers in such a way that desired data can be indexed and accessed.

f. Response to tailoring instructions. If a paragraph is tailored out of this document, the resulting document shall contain the corresponding paragraph number and title, followed by "This paragraph has been tailored out." For data in a database or other alternative form, this representation need occur only in the table of contents or equivalent.

g. Multiple paragraphs and subparagraphs. Any section, paragraph, or subparagraph in this DID may be written as multiple paragraphs or subparagraphs to enhance readability.

h. Standard data descriptions. If a data description required by this document has been published in a standard data element dictionary specified in the contract, reference to an entry in that dictionary is preferred over including the description itself.

i. Substitution of existing documents. Commercial or other existing documents, including other project plans, may be substituted for all or part of the document if they contain the required data.