CasaDomus

CMSC 447

Software Development Plan (SDP)

Date: May 13th, 2018

Version: 1.2

Authors: Cynthia Chou, Ryan Coleman, Elia Deppe

1 Scope

1.1 Identification

This Software Development Plan (SDP) is applicable to CasaDomus 1.0, a home finding program that runs on both Chrome version number 60.0+ and Firefox version number 58.0+, that searches for potential areas of housing based off a user's response entry to a questionnaire.

1.2 System overview

The purpose of this piece of software, which should run on any system that is capable of any of the two following web browsers, Chrome version 60.0+, and Firefox version 58.0+, is to enable users to easily find housing locations based off their preferences. The web-based software should perform these tasks by using user preferences obtained from a questionnaire to search through information obtained through various APIs, .csv files, and a database, which display to the user a map that is gradiented based off the comparability of the housing options in various areas to the user preferences.

1.3 Document overview

This is the Software Development Plan document that describes the plan that intended to be followed throughout the entirety of the development of the product.

2 Referenced Documents

- Software Requirements Specification (SPS)
  - Version 1.2
- Software Design Document (SDD)
  - Version 1.2
- Software Test Description (STD)
  - Version 1.2
- Software Test Report (STR)
  - Version 1.2
- Software User Manual (SUM)
  - Version 1.2

3 Overview of required work

a. Requirements and constraints on the system and software to be developed
   - Since the software is to be developed to be entirely functional on certain web browsers, including Chrome 58.0+ and Firefox 60.0+, the software should be functional on any device that supports such browsers. As consequence, the software has no known constraints on the system.

b. Requirements and constraints on project documentation
   - All project documentation should be done by 5/15/2018, signed off by out

client, Prof. Charles Nicholas, and turned into Prof. Russ Cain in a binder.

c. Position of the project in the system life cycle
    - The position of the project in the system life cycle depends on what has been completed for the project. On the completion of all the requirements for one position in the system life cycle, the project then moves onto the next step in the life cycle.

Position in Lifecycle	**Items to be Completed**
Requirements	Obtaining and documenting the requirements specified by the customer
Design	Designing the project around its requirements along with designing the GUI.
Implementation	Implementing the project, which is done simultaneously with the front end and the back end
Test	Test the functionality and performance of the code and determine whether requirements have been met
Maintenance	After the code has been tested, it should be updated, and the functionality shall be maintained based off needs.

d. Requirements and constraints on project schedules and resources
    - It shall be completed by May 15th, 2018.

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

    Since the project does not store any sensitive information, nor does it associate any potentially stored information with users in any direct manner, there are minimal concerns for project security and privacy.

4 Plans for performing general software development activities

4.1 Software development process

The Software Development Process used shall be a mix of Waterfall and Agile, predominantly following the Waterfall system, but utilizing the iterative development practices of Agile. Requirements will be obtained from the client, Prof. Nicholas and from there CasaDomus will begin designing the software. From there CasaDomus will develop the software incrementally, adding new CSU's and testing them before integrating them with the CSCI, until the software is finished. After that CasaDomus will begin testing of the product as well as debugging and redeveloping the CSCI to further match any changed requirements of the client. Once testing has finished, the software will be delivered. There will be no maintenance with this product.

4.2 General plans for software development

4.2.1 Software development methods

Throughout the course of the software development, Google Drive and Github were utilized. Github shall be used for a public repository for source code, documentation and as a configuration management system. Google Drive should be used for the purpose of storing and instantaneous collaboration on documents.

4.2.2 Standards for software products

a. Standards for format (such as indentation, spacing, capitalization, and order of information)
    - Please refer to the SDD, Section 6.1
b. 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.)
    - Please refer to the SDD, Section 6.1
c. Standards for other comments (such as required number and content expectations)
    - Please refer to the SDD, Section 6.1
d. Naming conventions for variables, parameters, packages, procedures, files, etc.
    - Please refer to the SDD, Section 6.1
e. Restrictions, if any, on the use of programming language constructs or features
    - Please refer to the SDD, Section 3.1
f. Restrictions, if any, on the complexity of code aggregates
    - N/A

4.2.3 Reusable software products

4.2.3.1 Incorporating reusable software products

Product	**License**	**Associated Benefits, Drawbacks, Restrictions**
Papa Parse 4	MIT	

Angular JS	MIT	
Google Maps Javascript API	Google Maps Platform License	

4.2.4 Handling of critical requirements

4.2.4.1 Safety assurance

The product does not have any safety concerns. The product does not store any data entered by the user in any form that correlates it with the user, the user cannot be threatened by attacks that seek to obtain information from the product. The product does not ask for any sensitive information from the user, and therefore, should any information be obtained by any potential attacks.

4.2.4.2 Privacy assurance

The product does not ask for any sensitive information, and does not store any potentially collected information from user inputs in any way that would clearly associate any entries with any particular individual.

5 Plans for performing detailed software development activities

5.1 Project planning and oversight

5.1.1 Software development planning (covering updates to this plan)

If need be the CasaDomus team will make updates to this SDP, and alert the client, Prof. Nicholas, to these updates.

5.1.2 CSCI test planning

CSCI testing shall follow the tests listed on the Software Test Description (STD) with its results being listed on the Software Test Report (STR). Testing of the CSCI shall take place iteratively as new CSU's are implemented and attached to the CSCI, to ensure that the CSU's do not cause error to the CSCI as a whole.

5.2 Establishing a software development environment

5.2.1 Software engineering environment

The following text editing softwares and interactive development environments were used to develop the source code, Vim, NotePad++, Visual Studio Code. These various text editors will be used by preference of the developer for writing code and various tests.

5.2.2 Software test environment

The following tools will be used for the purpose of testing the CSUs, CSCIs, and the system. Chrome's built-in developer console, Firefox's built-in developer console.

5.2.3 Software development library

The following software development libraries will be used: AngularJS, PapaParse, GoogleMaps API. AngularJS will be the bootstrap used for Javascript. PapaParse is used for parsing the datafiles. GoogleMaps will be used for displaying the data on a map.

5.2.4 Software development files

The following software development files will be used: Javascript (.js), JSON (.json), GEOJSON (.geojson), CSV(.csv), HTML (.html), and CSS (.css). Javascript will be used for scripting on a webpage. JSON, GEOJSON, and CSV files will be used as containers for datasets. HTML and CSS will be used for the design and styling of the webpages.

5.2.5 Non-deliverable software

The following Non-Deliverable Software will be used: Microsoft Office Excel and LibreOffice Calc. Excel and Calc will be used for viewing and modification of data.

5.3 Software requirements analysis

Acquiring software requirements will be done by the CasaDomus team first speaking with Prof. Nicholas, the client, and retrieving main ideas and thoughts through discussions as to what the product should be capable of. After retrieving these ideas, the CasaDomus team will then write a list of only the customer's requirement. Then, the CasaDomus team will being creating derived requirements from the Prof. Nicholas' requirements. After creating the document, the CasaDomus team will deliver the document to Prof. Nicholas to discuss if there are any issues or if it can be signed off. If needed, the team will make the changes as addressed by Prof. Nicholas and resubmit it for approval. Otherwise the team will then proceed upon designing the website.

5.4 Software design

5.4.1 CSCI-wide design decisions

The CSCI will be designed using a bottom approach, split into two main parts: the design of the website's UI and the design of the website's functionality or logic. From these two pillars, CasaDomus will build a product that will integrate into one, once deemed roughly functional, and then will continue to be built up slowly.

5.4.2 CSCI architectural design

The CSCI will architecturally be designed through two components: the layout of the website itself via HTML and CSS, and its logic which controls the function of the website via Javascript.

5.4.3 CSCI detailed design

The CSCI will utilize at most 2-3 HTML source files for the website design and UI. The HTML files will also most likely contain scripting code as well. Javascript files will be used for the storage of extracted data and logic.

5.5 Software implementation and unit testing

5.5.1 Software implementation

The first step in the software implementation process is to understand what is to be coded. To do this, the team must maintain communications with the customer, and obtain a clear understanding of the customer's desires, intentions, and goals in regards to the software. Next, a clear list of requirements that are fully testable and free of any ambiguity should be made and presented to the customer. With the customer's approval of the documented requirements, the team can begin to plan the implementation of the product. After a clear plan has been developed, and agreed upon by the members of the team, coding can begin. Once all pieces of the product have been written, they should be integrated.

5.5.2 Preparing for unit testing

Before performing unit tests, the unit intended for testing should be completed in the portions that the unit test attempts to check. After the completion of the code to be subjected to the unit test, the unit test must also be completely implemented.

5.5.3 Performing unit testing

The unit testing is performed by the coders on the team that write the specific fragment of the code being tested. This enables the coders on the team to ensure that the code they have written has no problems, knowing exactly what they implemented in the code. In other words, it enables them to test the internals of the code that they have written, thus allowing them to confirm that the code functions as expected. This piece of the testing should be done as the code is written, thereby making it possible for some bugs or problems to be found early in the development process.

5.5.4 Revision and retesting

After the testing is completed, the results are reviewed. For each of the tests that are not passing or obtaining expected results, the code shall be corrected and modified, before rerunning that unit test again. This process will be repeated until no visible errors can be found from the unit testing.

5.5.5 Analyzing and recording unit test results

All Unit test results will be filled into the Software Test Report (STR).

5.6 Unit integration and testing

5.6.1 Preparing for unit integration and testing

New Computer Software Units (CSUs) will be written in a new branch, off of the master branch in Github.

5.6.2 Performing unit integration and testing

First the CSU will be tested thoroughly on it's own until the CSU is deemed working. Once this is done it will be integrated into the CSCI. The CSCI will then be run through its normal tests along with new tests for the added on CSU to ensure that it has retained its original functionality, and the new CSU is still functional with the entire CSCI.

5.6.3 Revision and retesting

Should any issues be found from testing the CSU or from testing the CSCI once the CSU is implemented, the CSU will once more be tested on its own along the CSCI in its previous state, before trying to integrate them once more and seeing where the issue stems from them being integrated.

5.6.4 Analyzing and recording unit integration and test results

All results of testing will be recorded in the STR.

5.7 CSCI qualification testing

5.7.1 Independence in CSCI qualification testing

Due to the inability to hire Quality Assurance personnel, the CasaDomus team will be doing the qualification testing.

5.7.2 Testing on the target computer system

Testing will be done on the CasaDomus team member's computers, which are Windows and Macintosh systems.

5.7.3 Preparing for CSCI qualification testing

Qualification testing of the CSCI will begin once the all CSU's have been properly integrated and the software has gone through its Critical Design Review.

5.7.4 Performing CSCI qualification testing

Qualification testing will be done by each of the CasaDomus team members on their respective machines.

5.7.5 Revision and retesting

After the testing is completed, the results are reviewed. For each of the tests that are not passing or obtaining expected results, the code shall be corrected and modified, before rerunning that CSCI test again. This process will be repeated until no visible errors can be found from the CSCI testing.

5.8 Preparing for software use

5.8.1 Preparing the executable software

Since, the product is entirely web-based, a direct executable file for the product is not required. Instead, the product should require a user to obtain at least one of its supported web browsers, including Google Chrome version 60.0+ or Firefox 58.0+, which will be used to load, display, and execute the product. Instead, direct access to the product will be given via a .zip file containing all the files, making up the product. The entirety of the product can be obtained by opening the landing page of the site, which will give access to all of the product's functionality.

5.8.2 Preparing user manuals

The user manual will be written at the end of testing phase of the product by the software development team.

5.8.3 Installation at user sites

No installation is required for this product as it will be accessible from the web.

5.9 Software configuration management

5.9.1 Configuration identification

Configuration identification will automatically be handled by Github.

5.9.2 Configuration control

Configuration control will automatically be handled by Github.

5.9.3 Configuration status accounting

Configuration status accounting will automatically be handled by Github.

5.9.4 Configuration audits

Configuration audits will automatically be handled by Github.

5.10 Software product evaluation

Besides the initial testing that the team performs on the product, the customer and the professor will review the product in accordance to the Software Requirements Specification, which includes all of the requirements that were expected of for this project.

5.10.1 In-process and final software product evaluations

During the development lifecycle of the product, there will be weekly meetings with the client, Prof. Charles Nicholas, to show progress of the product as well as discuss any concerns and acquire any insight into additional requirements/questions that the development team or the client may have. On the date 5/15/2018, a final demo of the product will be showcased to the class for the client and Prof. Cain to see.

5.11 Other software development activities

5.11.1 Risk management, including known risks and corresponding strategies

Risk	Likelihood	Consequences	Response
Personal Emergencies	Medium	Possible Delay of Product and Documentation	Allocate personnel to cover for loss
Underestimation of Workload	High	Possible Delay of Product and Documentation	Double down on workload

Outdated Data	High	Possible Delay of Product and Documentation	Fetch new data
Broken Dependencies	Low	Broken Code and Delay of Product	Update API Keys or search for new API
DDOS	Low	Product Becomes Inaccessible	N/A
Loss/Corruption of Data	Medium	Broken Code and Delay of Product	Use of backups to retain data

6 Project organization and resources

6.1 Project organization

This project is done incorporation with the customer, Professor Charles Nicholas, of UMBC. Besides his involvement in specifying the requirements of the project, and indicating his satisfaction level, we also have to work within the expectations of Professor Russell Cain, of UMBC, who will audit the software development process of the team.. All other aspects of the project are managed by the team developing the code.

6.2 Project resources

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

a. Personnel resources, including:
   1) The estimated staff-loading for the project (number of personnel over time)
      - 6 Software Developers; no expected increase in personnel
   2) 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)

Area of Responsibility	Staff Responsible
Software Engineering and Testing	All Staff
Front end	William Gao, Elia Deppe
Map API	Ryan Coleman, Haoran Ren

Data Aggregation	Elia Deppe, James Williams, Haoran Ren
Back end	James Williams
Documentation	All Staff

3) A breakdown of the skill levels of personnel performing each responsibility

Cynthia Chou	
Ryan Coleman	Proficiency in C, C++, Python, Javascript. Basic in HTML, CSS
Elia Deppe	Proficiency in C, C++, Java, Android Development, and Python. Basic in HTML, CSS, and Javascript.
William Gao	Proficiency in C, C++, Objective C, Swift. Basic Javascript, HTML, CSS,
Haoran Ren	Proficiency in Python, C, C++. Basic in HTML and SQL.
James Williams	Proficiency in Python, C, C++, Java. Basic in HTML, Javascript, AngularJS

b. 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.
   1) All development will be done in the engineer's respective homes and on campus at UMBC. There is no need for secure areas.

7 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.

Term	**Definition**
CSCI	Computer Software Configuration Item
HWCI	Hardware Configuration Item
IV & V	Independent Verification & Validation
SQA	Software Quality Assurance

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.