CMSC 447
Software Development Plan (SDP)

Group 4 - */* No Comment*/*

Realty Map Filter

Christopher Banos
Joseph Graso
David Udwin
Xavier Harris
Hannah Kiesel
Federico Cifuentes-Urtubey

Table of Contents

1 Scope 3
1.1 Identification 3
1.2 System overview 3
1.3 Document overview 3
2 Referenced documents 3
3 Overview of required work 3
4 Plans for performing general software development activities 4
4.1 Software development process 4
4.2 General plans for software development 5
4.2.1 Software development methods 5
4.2.2 Reusable software products 5
4.2.3 Handling of critical requirements 5
5 Plans for performing detailed software development activities 6
5.1 Project planning and oversight 6
5.1.1 Software development planning (covering updates to this plan) 6
5.1.2 Software installation planning 6
5.2 Establishing a software development environment 6
5.2.1 Software engineering environment 6
5.2.2 Software test environment 6
5.2.3 Software development library 6
5.2.4 Software development files 7
5.3 System requirements analysis 7
5.3.1 Analysis of user input 7
5.3.2 Operational concept 7
5.3.3 System requirements 7
5.4 System design 7
5.4.1 System-wide design decisions 7
5.5 Software design 7
5.5.1 CSCI-wide design decisions 7
5.5.2 CSCI architectural design 7
5.5.3 CSCI detailed design 7
5.6 Software implementation and unit testing 8
5.6.1 Software implementation 8
5.6.2 Preparing for unit testing 8

5.6.3 Performing unit testing 8
5.6.4 Revision and retesting 8
5.6.5 Analyzing and recording unit test results 8
5.7 Software product evaluation 8
5.7.1 In-process and final software product evaluations 8
5.7.2 Software product evaluation records, including items to be recorded 8
5.7.3 Independence in software product evaluation 8
5.8 Corrective action 8
5.8.1 Problem/change reports, including items to be recorded 8
5.8.2 Corrective action system 9
6 Schedules and activity network 9
7 Project organization and resources 10
7.1 Project organization 10
7.2 Project resources 10
8 Notes 10

1 Scope

1.1 Identification

This document applies to the program identified as the Realty Map Filtering (RMF) application. The version number as of March 29, 2018 is 1.0.0.

1.2 System overview

This application's purpose is to locate houses for sale across the United States (U.S.) that meet certain specifications defined by the user. These specifications include, but are not limited to, property data (e.g., market value), nearby schools, crime rates, and community types (e.g., urban, suburban, or rural). The results will be displayed on a map within the user interface (UI). The details of a property listing will be shown along with the specifications that it met from the user's query.

The developer team of this project is */* No Comment */*. The sponsor and primary user is Yatish Joshi from Cisco Systems. The system documentation will include this Software Development Plan (SDP), and these external documents: Software Requirements Specification (SRS), Software Design Description (SDD), Software Test Description (STD), Software Test Report (STR), and a Software User Manual (SUM).

1.3 Document overview

This document contains an overview of the RMF application project. It allows the client to understand what we plan to create, how we plan to create it, and how we plan on testing it. This will allow the client to insure that we are creating what they intended and it is to their preferences.

We consider network traffic generated by this application will require security in order from being altered and returning false results. A countermeasure for this will require us to use an HTTPS connection as a query is made. Currently, we have no privacy concerns due to no intent of collecting information about the user in addition to the absence of a login account system.

2 Referenced documents

Software Requirement Specification (version 1), March 28, 2018.

Software Test Report (version 1), May 1, 2018.

3 Overview of required work

These requirements and resources shall be described in greater detail in later parts of this document as well as in the SRS document.

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

Requirements that the project must accomplish:

- Results shall be areas of interests
- Results shall be shown on a map
- Results shall be based upon a search from user inputs
- Primary return shall be houses to buy based on the specifications
- Search options will include local schools
- Local schools will include their ranking
- Search options will include states
- Search options will include counties
- Search options will include specific communities
- Search options will include local crime rates

Constraints of the Project:

- Only access to free APIs
- Each API has limited calls per day for free

b. Requirements and constraints on project documentation

Project documentation will be required to be updated as the project progresses, which will require the developers to ensure all software documentation is consistent with such changes. The documentation shall be stored on a cloud server for the developers to access.

c. Position of the project in the system life cycle

As of May 1, 2018, the system is currently in its development phase involving programming Python scripts to access the RedFin API and Google Maps API.

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

The project will be written in Python, Java, and JavaScript to 1) query APIs and 2) allow the program to execute on a local machine.

e. Requirements and constraints on project schedules and resources

The project is to be completed at the end of the current semester, May 10th, 2018.

4 Plans for performing general software development activities

4.1 Software development process

Due to the nature of a classroom setting, we are forced to use agile development. This plan comes in six phases: meet, plan, design, develop, test, and evaluate. This process will be repeated for requirement and may be repeated multiple times for any given requirement until it is satisfactory.

The meet phase involves establishing requirements, and discussing changes that need to be made from the previous evaluation. This plan phase is where the development team will decide how to meet and integrate the requirement into the current state of the project. Those working on creating the code will record the exact purpose of each part of the software to make testing accurate and easier. The development phase will be the actual coding and creation of the software. The test phase will be where the testing member of the team will ensure that the purpose recorded has been achieved and the software is free of bugs. If any issues do arise, it will logged and progress will return back to the development phase to correct the problem. Once all bugs have been removed, the client will evaluate the software to ensure that it is as they expect it to be. If it is, progress will move on to the next requirement. Otherwize, the cycle will begin again still focusing on the particular specification.

4.2 General plans for software development

4.2.1 Software development methods

Because this project will operate on a web browser, HTML, CSS, and Angular will be used to create the user interface. Javascript will be used to display map results. Python will be used to communicate with APIs to process filtering results. Lastly, GitHub will be used to synchronizing our progress on the code.

This project will be implemented as a web application. The front end with be implemented using Angular. The back end will be implemented using Python. Flask will be added to Python to run the application. GitHub will handle source control. Travis CI will handle continuous integration and automatically build the project when new commits are pushed.

4.2.2 Reusable software products

This subsection shall be divided into the following subsections.

4.2.2.1 Incorporating reusable software products

This project will use Angular, jQuery, Bootstrap, Redfin, Zillow, and the Google Maps API. Angular will be used for front end development because it enhances Javascript's abilities in manipulating HTML. This will improve the development process and results through a more friendly user interface. jQuery is necessary for Angular. Bootstrap improves the aesthetics of the web application. Redfin and Zillow are APIs that will be the data that will be filtered to provide the results to the user. Lastly, Google Maps will be used to display and provide and interactable map to visualize the location of the houses that meet the criteria of the search.

4.2.2.2 Developing reusable software products

Due to the way that this project is being developed, each part of it is separated and therefore can be used in many different ways. The code base, hosted on GitHub, will be easy to navigate and read. When possible, existing software packages that have extensive documentation will be utilized. With the exception of this use, the development team, */* No Comment */*, takes no responsibility for how the code in this application is used.

4.2.3 Handling of critical requirements

This subsection shall be divided into the following subsections.

4.2.3.1 Safety assurance

Safety assurance for the RMF is provided through its inability to track user information or location. It does not require any personal information about the user to operate.

4.2.3.2 Security assurance

To provide security in the form of confidentiality, the RMF will use HTTPS connections to access databases/APIs. For integrity, the TLS protocol used in the HTTPS connection will provide such service. For availability, the user is only limited through whether or not they have Internet connectivity.

4.2.3.3 Privacy assurance

The privacy of the user of the RMF is preserved through the absence of storing search results and the absence of user accounts. In short, there is no tracking involved in the RMF.

4.2.3.4 Assurance of other critical requirements

Other critical requirements, detailed in the SRS, include many requirements that the developers label as "derived," which can be considered as subrequirements or related to the client's defined requirements.

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)

The team members communicate via GroupMe. The version control of the code base is handled over GitHub. Travis CI is used to handle continuous integration and automate testing of the web application when new commits are made. Documentation is recorded using Google Docs. If this plan is updated, we will record it using Google Docs.

5.1.2 Software installation planning

Install Flask and Python. The software will not run without both of these installed. Git also needs to be installed to access the version control.

5.2 Establishing a software development environment

5.2.1 Software engineering environment

The software is developed and ran on a Macintosh laptop with macOS High Sierra v10.13.4.

5.2.2 Software test environment

The test environment is Travis CI as well as a Macintosh laptop with macOS High Sierra v10.13.4. First, any new changes will be tested on the local machine (Macintosh laptop) they are developed on. When verified by the developer, they will be pushed to GitHub. Travis CI will then pull the code from GitHub and test it in the cloud.

5.2.3 Software development library

The libraries used are Angular, jQuery, Bootstrap, and Flask. The purpose of each of these were identified in section 4.2.3.1.

5.2.4 Software development files

- Index.html: lays out a webpage
- app.js: controls the webpage
- style.css: stylesheet for webpage
- map_of_jobs.py: backend of web application

5.3 System requirements analysis

5.3.1 Analysis of user input

The user input will consist of the following:

- State and zipcode will be entered through text
- Crime rates
- School ratings
- Housing cost
- House features

5.3.2 Operational concept

User inputs desired parameters. Based upon those parameters, results are calculated from the various APIs. The results are a collection of houses, with their location. The Google Maps API then uses these locations to display where the houses are located. Finally these homes are shown to the user to show their address and location.

5.3.3 System requirements

The system must be able to run a web browser with Internet connection.

5.4 System design

Refer to section 5.5 for details on the RMF system.

5.4.1 System-wide design decisions

Development was performed on a Macintosh Laptop and code was hosted on GitHub. The software will be tested on Travis CI.

5.5 Software design

5.5.1 CSCI-wide design decisions

Existing software packages will be used when possible along with existing APIs to process the specifications. Refer to section 4.2 for specific tool descriptions.

5.5.2 CSCI architectural design

The developer team divides the CSCI into two parts: the front end and the back end. The front end will consist of the UI, the search parameters, the user input, location display of search results, and text display of search result information. The back end will consist of API queries and organizing data for the UI to display.

5.5.3 CSCI detailed design

Refer to Section 5 of the SDD, which shall describe in detail each component of the product.

5.6 Software implementation and unit testing

5.6.1 Software implementation

We will use the PyCharm IDE to assist with Python development and Sublime text to organize other code. The tools and APIs the development team uses are described in section 4.2.

5.6.2 Preparing for unit testing

The software development members will write the program document list what the inputs and outputs are supposed to be according to the requirements listed in the SRS.

5.6.3 Performing unit testing

We will implement the unit tests using PyCharm. We will follow the documentation given by the developers who wrote the code being tested and write up meaningful tests to cover the majority of the functionality of the code.

5.6.4 Revision and retesting

Everytime we run a unit test, we will run the rest of the unit tests to make sure one section of code doesn't break another. If we make substantial changes to a section of code, we will look over its unit tests to be sure the tests are still meaningful and provide adaquest coverage.

5.6.5 Analyzing and recording unit test results

Unit tests are written with a predefined expected result in Python. If a test result does not meet the expectation, then it has failed and the developers must fix the code to meet such expectation. If the test result matches the expectation, another test may be conducted to analyze a variety of points in the product's code.

5.7 Software product evaluation

5.7.1 In-process and final software product evaluations

During development, a regular meeting between developers and client will be held every two weeks. Any documentation written or requirements coded will be reviewed by the client. The client will ensure that their expectation are met. If the client notices something unsatisfactory, the development team will make note and correct it and follow up on it during the next meeting. This cycle will repeat until all requirements have been completed. A final overall evaluation and correction cycle will occur, completing the RMF software.

5.7.2 Software product evaluation records, including items to be recorded

When evaluating the product, the developers will identify bug fixes in each commit to the GitHub repository. Meeting minutes will also be recorded in Google Docs and shall include major updates to the code that regard the product requirements.

5.7.3 Independence in software product evaluation

The developers are the ones in charge of the product's correct functionality. They will also alert the client of any changes to the requirements.

5.8 Corrective action

5.8.1 Problem/change reports, including items to be recorded

Any major problems or changes shall be recorded in documentation, specifically in the STR. The logs will include the file name, description of the problem, and description of the solution.

5.8.2 Corrective action system

The developer that encounters a problem with the code they wrote is responsible for fixing that code if it causes a problem.

6 Schedules and activity network

a. Schedule of documents

Date	Review	Document
March 9, 2018	Software Requirements Review	Software Design Plan (SDP)
March 16, 2018	Software Design Review	Software Requirements Specification (SRS)
March 30, 2018		Software Design Description (SDD)
April 13, 2018	Preliminary Design Review	
May 3, 2018	Critical Design Review	Software Test Description (STD)
May 10, 2018	Test Readiness Review	Software Test Report (STR)
May 10, 2018	Software Acceptance Review	Software User Manual (SUM)

b. Activity Network

The RMF's Search feature will include ability to search for houses and to display search results dependent on the user's search criteria. The Search activity will depend on several APIs in order to query for the variety of criteria within a user's search. Read about the APIs in use in section 4.2.2.1. The developers expect this Search activity to take the longest to implement, rating it 13 on a Fibonacci scale of difficulty.

Displaying search results will be another main activity within the RMF. API queries will return to the user's browser, and the download speed will be dependent on the performance of their Internet connection. As for the interface of results, organizing them in a clear fashion will either be a static, scrollable window or a dynamic, adaptive window that can scroll as well. In addition to the housing information, a map displaying locations of the search results will be present. The developers expect this Search Display activity to take a high level of effort, rating it 8 on a Fibonacci scale of difficulty.

7 Project organization and resources

7.1 Project organization

There is only one organization involved, which is the developer team */* No Comment */* from UMBC. The developer team will be responsible for the project, which includes implementation, testing, and software documentation.

7.2 Project resources

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

a. Personnel resources, including:
   1) Six developers for the 15-week-long project
   2) Three of the developers will be mainly responsible for the software implementation and three of the developers will be mainly responsible for the software documentation. Implementation includes designing the user interface and determining functionality of the software tools in use. Documentation includes designing the tests for the product and describing the system in whole.

b. Refer to section 4.2.1 for a description of project resources and how they will be used.

8 Notes

The Fibonacci scale is used in an Agile programming environment, and specifically in this project, in order to denote a level of difficulty for a given task. It is designed to conceptualize iterative development, known as sprints, and the scale begins at 0 and is continuous in steps as in the Fibonacci sequence (1, 2, 3, 5, 8, ...). The higher the number, the more complex the task.

Acronyms

API - Application Programming Interface

CSCI - Computer Software Configuration Item

HTTP - Hypertext Transfer Protocol

HTTPS - Hypertext Transfer Protocol Secure (HTTP over TLS)

IDE - Integrated Development Environment

RMF - Realty Map Filter

SDD - Software Design Description

SDP - Software Development Plan

SRS - Software Requirements Specification

STD - Software Test Description

STR - Software Test Report

SUM - Software User Manual

TLS - Transport Layer Security (protocol)

UI - User interface

UMBC - University of Maryland, Baltimore County