# Code Fury

# Software Development Plan (SDP)

I hereby accept this document as complete.

Signature: Date: 5/15/2018

## Contents

# 1 Scope

## 1.1 Identification

This document pertains to system version 1.0.0.

## 1.2 System overview

This system aims to provide users with an interactive map of the United States marked with areas that satisfy their lifestyle preferences.

## 1.3 Document overview

This document outlines logistical decisions that pertain to software design.

# 2 Referenced documents

The following is a list of documents referenced in this document.

- Software Requirements Specification (SRS)
- Software Design Document (SDD)

# 3 Overview of required work

The project will be divided into two major parts, namely front-end development and back-end development. The front end will consist of a web page with parameters used to relay search conditions to the back end. The back end will consist of a server that stores lifestyle data. This data will be pulled from various government agencies to ensure quality. When a user initiates a search, the data will be relayed to the Google Maps API, which will drop markers on the map display in areas that satisfy search conditions. Hovering over a marker will reveal more information about why that area matches the search conditions.

Potential data sources will be examined to determine what information best fits the search parameters. Front-end and back-end development will begin in parallel to meet our deadline and ensure compatibility. Then, the two will be integrated and tested. Upon completion, the system will be tested, and all documentation proof read and finalized.

# 4 Plans for performing general software development activities

## 4.1 Software development process

The project will be developed using the agile method, with weekly and biweekly sprints to accomplish goals. A Gantt chart breaking down the sprints is available below in section 6. The group will meet in person on Thursdays at 2:30 PM with Professor Marron to confirm the completed work is satisfactory and prepare for the next sprint.

## 4.2 General plans for software development

### 4.2.1 Recording rationale

At every meeting, one group member will record all comments, concerns, and confirmations given by Professor Marron. From these notes, an email confirmation reiterating what was discussed will be sent to all members and Professor Marron.

In addition, any discussions after class or outside of regular meetings will be sent as an email to all group members to distribute work as needed, confirm smaller changes, and help each other in between sprints.

# 5 Plans for performing detailed software development activities

## 5.1 Project planning and oversight

### 5.1.1 Software development planning

Planning will take place after classes on Tuesday and Thursday nights and following bi-weekly meetings with our customer, Dr. Marron on Thursday afternoons. In addition, weekend/evening work sessions and further planning will be scheduled as needed. Planning will be confirmed by an email sent by Kyle Coleman sent after the meetings to confirm work schedules.

### 5.1.2 CSCI test planning

Individual components will be tested as they are created and integrated into the system. In addition, periodic user tests will be done, with volunteers providing feedback on website usability and aesthetic. The customer, Dr. Marron, will have final say on the appearance and layout of the front end.

### 5.1.3 System test planning

As the CSCI's are integrated into the system, periodic tests will be done to ensure stability. Following these, stress tests to ensure the API's and databases can be repeatedly called without error will be given to ensure integrity.

### 5.1.4 Software installation planning

The project is a website that will be hosted on a local computer for simplicity's sake. In addition, a local SQL server will be used in conjunction with API's to keep the time and cost of data retrieval to a minimum.

## 5.2 Establishing a software development environment

### 5.2.1 Software engineering environment

We will be using Google Docs as our writing environment, using the templates provided by Dr. Cain. Code will be stored and developed on GitHub. That is, code will be developed on local machines and then pushed to GitHub, followed by pulls from group members to ensure compatibility across multiple devices.

### 5.2.2 Software test environment

Testing will be done at all stages of development (as both the CSCI's are created and as they are integrated into the system). A final test for functionality will be done when the system is functional and ready to be presented. Testing, at every level, will be done by all members of the group to ensure accuracy.

### 5.2.3 Software development library

The front-end will be a webpage in HTML with CSS wrappers for appearance, supported by a PHP framework for accessing the database/API's. In addition, the Google Maps API will be used to create the map display and overlay it with markers. MySQL will be used to construct and maintain the database.

### 5.2.4 Non-deliverable software

Software that is not part of the final project (e.g. wireframes, data visualizations, etc.) will be stored on GitHub. Older versions will remain there as reference points for later work and as a backlog of older editions.

## 5.3 System requirements analysis

### 5.3.1 Analysis of user input

After formal requirements have been written, they will be confirmed with the customer, Dr. Marron. User inputs will be based on the parameters requested by Dr. Marron. The parameters will take the form of buttons and will be used to narrow the search by conditions defined in the requirements. Front end requirements will be added to the project as specified, and the databases and website frameworks will be tailored to be consistent with back-end requirements. Should any questions arise, they will be clarified with Dr. Marron before moving forward.

### 5.3.2 Operational concept

User needs will be mapped to requirements in the Software Design Description (SDD) and Software Requirements Specification (SRS). In addition, user tests from volunteers will be used to provide feedback on how the website might be streamlined, with input and approval coming from Dr. Marron.

### 5.3.3 System requirements

To fulfill the requirements given by Dr. Marron, the project will consist of both a front-end (web page) and a back-end (using databases/API's). The front-end will require an internet browser and internet access, while the backend will require internet access for the API traffic and a local copy of an SQL database. The project will be hosted locally on the user's machine, avoiding the need for calls to an external server.

## 5.4 System design

### 5.4.1 System-wide design decisions

The site will consist of a single web page containing a map display and parameters for the user to narrow their search query. After a search is performed by the user, the results will show up on the map as markers. If a user hovers over a marker then a popup will appear with details about that location.

### 5.4.2 System architectural design

We plan to use the Google Maps JavaScript API in addition to bootstrap for the front-end. For the backend, we will use Apache and PHP. Ideally, we will be able to get all our data from APIs as we need it, but if that is too slow or if we must use CSVs as data sources, then we will create an SQL database. When a user goes to the webpage and submits the form specifying their search criteria, the JavaScript will pass the request to the backend, which will query an API and/or the database and return an array of location results. The JavaScript code in the frontend will then interact with the Google Maps API to plot the resulting locations on the map.

## 5.5 Software requirements analysis

After receiving requirements, we will research and find data sources such as APIs and CSV files that contain the data needed satisfy them. Then we will work to narrow down the data sources into the smallest set that still provides all the data needed to satisfy the requirements. If a search criteria requirement is unfeasible given the data available, then we will discuss this with our client and work to adjust the requirement.

## 5.6 Software design

### 5.6.1 CSCI-wide design decisions

The frontend will use the Google Maps JavaScript API, JQUERY, and Bootstrap. The backend will use PHP (Apache server).

### 5.6.2 CSCI architectural design

Frontend: This webpage will contain an embedded map for displaying results and a search form for specifying desired search parameters.

Search.php: This file will receive search parameters from the frontend and then return an array of locations that align with those parameters.

Database: If we decide we need a database, we will create two tables:

- States: This table will contain the name (and a unique identifier) for each state.
- Counties: This table will contain the name (and a unique identifier) for each county and the ID of the corresponding State that it belongs to. It will also contain a column for each parameter.

### 5.6.3 CSCI detailed design

Frontend: The frontend of the site will consist of a static HTML file, CSS file, and JS file. The user will be able to click a “hamburger” button to have a form appear with search options. The form will contain a search button at the bottom that will hide the form and cause the results to appear on the map. When the search form is submitted, the JavaScript will store the form’s values as JSON and then send a POST request to Search.php using the JQUERY library. When results are returned the JavaScript will use the Google Maps API to display a marker on the map for each search result.

Search.php: This file will receive parameters from a POST request made by the frontend. This file will then parse the parameters and use them to query the API and/or the database and then return the results to the POST request as a JSON array of locations (such as county names or coordinates).

## 5.7 Preparing for software use

Code Fury will release the software as open source with instructions for downloading from our GitHub account to set up on a server for personal or business use.

### 5.7.1 Preparing version descriptions for user sites

Code Fury will prepare the software for delivery to Dr. Marron. The software will be divided into batch files, command files, data files or other software files needed to install and operate the software on a server. We will also provide instructions for implementation to include command line scripts for setup.

### 5.7.2 Preparing user manuals

Our user manual will consist of instructions for operating the software. These instructions will pertain to the front-end.

## 5.8 Software configuration management

The SCM will define and establish uniform steps to control and maintain our product. This will be a best effort as directed by our task leader in managing and maintaining future changes. The following items will refer to source code, documents, and diagrams.

### 5.8.1 Configuration control

After our product becomes stable, individual software will be lumped into one version which we may title SR-1 (Software Release 1). Revisions will follow the scheme identified above.

### 5.8.2 Configuration status accounting

The status of our progress will be established and recorded during office hour visits with Dr. Marron and team meetings as directed by the task leader.

### 5.8.3 Configuration audits

Bi-weekly internal audits may be conducted by our task leader or another person who volunteers.

### 5.8.4 Packaging, storage, handling, and delivery

All maintenance will be done in our GitHub repository, which is open to anybody who wishes to track our progress.

## 5.9 Software product evaluation

### 5.9.1 In-process and final software product evaluations

May 1st, 2018 is our target date for final evaluation. May 10th, 2018 is our expected software demonstration date. Our final presentation will be conducted in Fine Arts 303 at 5:30 pm.

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

Each team member will provide commits with detailed explanations of what they are pushing, a log of these changes will be created.

### 5.10 Software quality assurance

Task leader will establish testing protocols which will utilize black box methods. Tests will search for possible errors and will ensure that we have satisfied our system requirements.

Process audits will be conducted to find deficiencies and help close them out. Task leader will assign SQA audits of the software against the SDP.

Task leader will resolve the discrepancies with SQA.

#### 5.10.1 Software quality assurance evaluations

Unit tests will be conducted as a group, walkthroughs will be conducted during our weekly team meetings so that any problems can be addressed as a team.

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

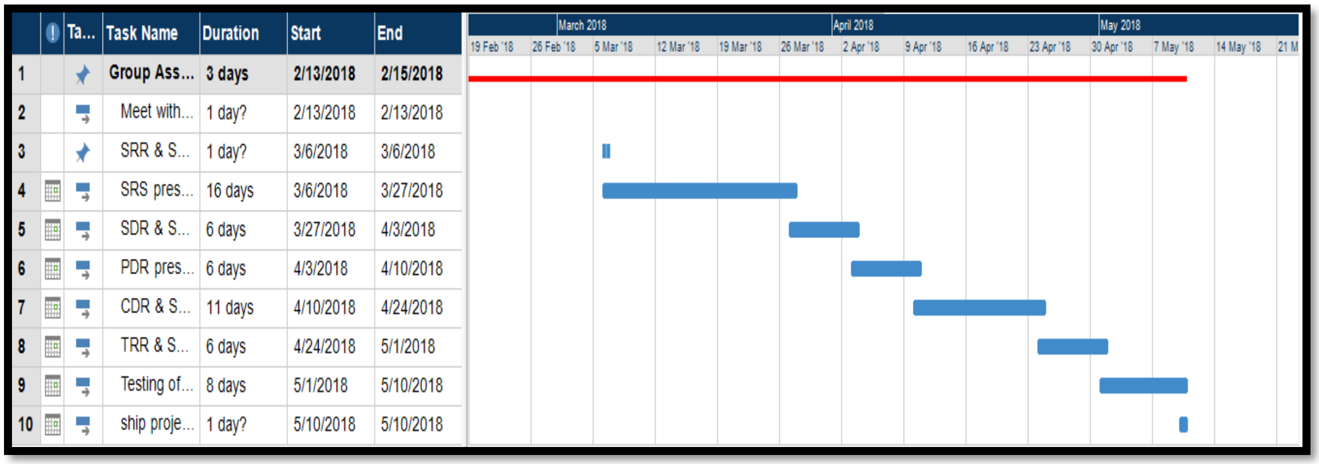
GitHub will have annotations of bugs we have found and any possible resolutions of those bugs.

## 6 Schedules and activity network

The current plan is to have bi-weekly meetings with Dr. Marron to discuss progress on the project, approve work, and receive feedback and clarifications.

- Feb 13 - Group assigned and project proposed
- Feb 15 - Meet with Dr. Marron to discuss requirements, begin work on SRR & SDP
- Mar 6 - SRR & SDP presented to Dr. Marron for approval, begin work on SDR & SRS
- Mar 13 - SRS presented to Dr. Marron for approval, begin work on SDD. Begin gathering API data and research options.
- Mar 27 - SDR & SDD presented to Dr. Marron for approval, begin work on PDR. Begin work on project front end and back end.
- Apr 3 - PDR presented to Dr. Marron for approval, begin work on CDR & STD
- Apr 10 - CDR & STP presented to Dr. Marron for approval, begin work on TRR & STR
- Apr 24 - TRR & STR presented to Dr. Marron for approval, begin testing project, begin work on SAR & SUM
- May 1 - Testing of project complete, project shippable. present project, SAR, and SUM to Dr. Marron for approval. Make final corrections as needed
- May 10 - Ship project. Present project in class for peer and teacher review.

The following is a Gantt chart for the schedule described above.



| | ! | Ta... | Task Name | Duration | Start | End |
|---|---|---|---|---|---|---|
| 1 | | | Group Ass... | 3 days | 2/13/2018 | 2/15/2018 |
| 2 | | | Meet with... | 1 day? | 2/13/2018 | 2/13/2018 |
| 3 | | | SRR & S... | 1 day? | 3/6/2018 | 3/6/2018 |
| 4 | | | SRS pres... | 16 days | 3/6/2018 | 3/27/2018 |
| 5 | | | SDR & S... | 6 days | 3/27/2018 | 4/3/2018 |
| 6 | | | PDR pres... | 6 days | 4/3/2018 | 4/10/2018 |
| 7 | | | CDR & S... | 11 days | 4/10/2018 | 4/24/2018 |
| 8 | | | TRR & S... | 6 days | 4/24/2018 | 5/1/2018 |
| 9 | | | Testing of... | 8 days | 5/1/2018 | 5/10/2018 |
| 10 | | | ship proje... | 1 day? | 5/10/2018 | 5/10/2018 |

## 7 Project organization and resources

The organizational structure will consist of a task leader, design engineer, developers, customer liaison and technical writer. At any time, additional duties can be established by the task leader to help distribute the work flow amongst the group. The task leader reports directly to Prof. Cain, issues that require resolution to involve the customer will be a joint effort with customer liaison and the task leader.

### 7.1 Project organization

Task Leader - Kyle Coleman

Design Engineers: (front end) Brian Sapp, (back end) Dan Engbert

Technical Writer: Tyler Karlsen

Quality Assurance: Rushmie Kulkarni

Customer Liaison: Jon Danko

Developers: Doug Gessleman, Jon Danko, Tyler Karlsen, Rushmie Kulkarni

Other tasks as assigned by task leader.

### 7.2 Project resources

The software development will be supported by the following items:

a. Personnel resources, including:
   1) 7 persons
   2) Virtual Machines to implement servers, Microsoft Word, Microsoft Excel
   3) Dan Engbert: PHP, SQL, Python, C++, HTML, CSS, Javascript; Kyle Coleman: Python, C++, HTML, CSS, Javascript; Brian Sapp: HTML, CSS, Python, PHP, Javascript; Tyler Karlsen: C, C++, Python, Java, HTML, CSS, Rushmie Kulkarni: Javascript, SQL, Python, PHP, C, C++; Jon Danko: C, C++, Java, Python, PHP, SQL; Doug Gessleman: C, C++, Java, Python, Visual Basic .NET

b. GitHub as a repository, UMBC AOK Library for meetings, classroom for presentation.

## 8 Notes

Leaving this section for later user.