

**Interactive Statistic Mapping Application**

**Software Development Plan (SDP)**

CMSC447\_Team3\_CodePods\_SDP

Version 1.1

March 26, 2017

|  |  |
| --- | --- |
| **Course** | CMSC 447 |
| **Team** | Team 3 - Code Pods |
| **Members** | Benjamin Hazlett  Darrell Laffoon  Desiree Mercuree  Kevin Miller  Ian Moskunas  David Pan |
| **Sponsor (Customer)** | Shawn Squire |

# 

**Revision History**

|  |  |  |
| --- | --- | --- |
| **Date** | **Version** | **Description** |
| 03/24/2018 | 1.0 | First Draft Completed |
| 03/26/2018 | 1.1 | Changes to Intro and System Overview |

**Table of Contents**

[**Introduction**](#_gjdgxs) **5**

[Purpose](#_5wkzo6sbs1qg) 5

[Scope](#_at6dn5z70qz8) 5

[**References and Definitions**](#_3dy6vkm) **5**

[Table 1 - Document Artifacts](#_6dj1j7eavujw) 5

[Table 2 - Glossary of Terms](#_fjapkn1xv87d) 6

[**System Overview**](#_aj1pdb38zxvo) **7**

[Diagram 1 - System Overview](#_rbv2sdz15po) 7

[Table 3 - Application Components](#_nwsbzxy9zdi2) 8

[Table 4 - Application Actors](#_7g80tbsrr37n) 8

[**Project Management Process**](#_ndm51wj6w0s8) **9**

[Team Roles and Responsibilities](#_xyygig7zftf0) 9

[Project Plan and Deliverables](#_7eba5mj9qemw) 9

[Table 5 - Project Plan](#_wypnpf2z98gc) 9

[Table 6 - Project Deliverables](#_ipxxjeq0wy4a) 9

[**Software Development Lifecycle**](#_vb19vld19e5j) **10**

[Diagram 2 - Kanban Tracking Board (example)](#_igcmouairqsa) 10

[Requirements Analysis](#_z0904t7vfz1g) 10

[Table 7 - Requirements Analysis process](#_sip330vmikdg) 11

[Software Design](#_v6gseqxu4wok) 11

[Table 8 - Software Design Process](#_atwet59eqehn) 11

[Software Implementation](#_1goc6yh4c0r5) 12

[Table 9 - Software Implementation Process](#_3kqqxdu51g5j) 12

[Software Testing](#_xz6b3l6rts83) 12

[Table 10 - Software Testing Process](#_u9l2vmhbai2z) 13

[Delivery](#_xz6b3l6rts83) 13

[**Configuration Management**](#_stlulyojkgnq) **14**

[Table 11 - Development and Configuration Management Tools](#_vf3j4wdd9hym) 14

[Development Environment](#_c7ykf560fvc7) 15

[Version Control](#_vn7rtnw1qdka) 15

[Testing Framework](#_jwitj2ok13l) 15

[Build and Deployment](#_ait3w8ktgyt) 16

[Coding Style and Code of Conduct](#_ct3m81bjic5t) 16

[Licensing](#_5xnaeucjy6xx) 16

# Introduction

## Purpose

The purpose of this Software Development Plan (SDP) is to define the the development processes, tools, and methodologies that will be used to develop the Interactive Statistic Mapping Application.

## Scope

This SDP describes the plan that will be used by the team to develop the Application. It will describe the plan for the various phases of the Software Development Life Cycle (SDLC) that they team will follow, specifically focusing on the design, implementation, and testing phases.

The SDP contains the following information:

* The Project Management Process including:
  + Project Plan
  + Schedule of Deliverables
* The software development approach including the approach to:
  + Design
  + Implementation
  + Testing
* Change Management
  + Environment
  + Development and CM Tools
  + Process Guidance

# References and Definitions

### *Table 1 - Document Artifacts*

|  |  |  |  |
| --- | --- | --- | --- |
| **Document** | **Description** | **Version** | **Date** |
| CMSC447\_Team3\_CodePods\_SRS | Software Requirements Specification | 1.2 | 03/26/2018 |
| CMSC447\_Team3\_CodePods\_SDD | Software Development Plan | 1.1 | 05/12/2018 |
| CMSC447\_Team3\_CodePods\_STD | Software Test Description | 1.1 | 05/14/2018 |
| CMSC447\_Team3\_CodePods\_STR | Software Test Report | 1.3 | 05/15/2018 |
| CMSC447\_Team3\_CodePods\_SUM | Software Users Manual | 1.2 | 05/14/2018 |

### 

### *Table 2 - Glossary of Terms*

|  |  |
| --- | --- |
| **Term** | **Meaning** |
| **The Application** | The targeted software solution - the Interactive Statistic Mapping Application |
| **The System** | The System that encompasses the application. The System and The Application could be used interchangeably in most cases |
| **The Customer** | The Customer that sponsored the project/software Application - Shawn Squire |
| **The Team** | The team of students who will build the Application |
| **SDLC** | Software Development Life Cycle - describes the steps and phases used to design, build and test the application |
| **Agile** | A set of principles that are used to define a iterative and incremental SDLC that is used to build the Application in iterations and allows for Customer feedback to guide the development of the Application |
| **Actor** | A person, external system, or other ‘actor’ who interacts with The Application or System |
| **Component** | A sub-system or part of the Application or System |
| **Map** | The selected, displayed Map, including boundaries. For example, it could be the State of Maryland, or Baltimore, or a neighborhood |
| **Map Data** | Detailed data about the selected Map |
| **Statistics Option** | One of the statistic choices such as crime, school ranking, etc. |
| **Proof of Concept** | A version of the Application that is used to test and prove aspects of the design |
| **Use Case** | Details behavioral requirements |
| **System Model** | Diagram that depicts system components and communication context |
| **Class Model** | Documents the Data entities of the System |
| **Sequence Model** | Documents the sequence of events between actors and components for particular use cases and application events |
| **State Model** | Documents state transitions for the system during particular uses casee and application events. |
| **Test Driven Development (TDD)** | A development methodology where unit tests are created before components are built. |
| **Software Repository** | A distributed data store that holds and tracks versions of the Application’s source code, documentation and other artifacts. |
| **Unit Test** | A test done on low level components as they are built |
| **Integration Test** | A test that tests the ability of system components to work together properly |
| **Regression Test** | A Test that checks that previously implemented features are still working after changes are deployed |

# System Overview

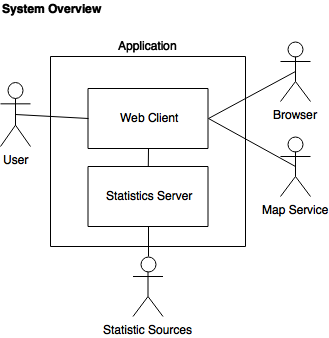
This section provides a high level System Overview for the Application. The Application’s purpose is to help a user answer a universal question:

"Where to I want to live, work, or retire?”

The Application will allow a user to select a Map and visually overlay important Statistics such as crime, income, school ranking, average commute, etc. The overlay will help the user identify areas on the Map that are more or less desirable. An expected, common use case would be a parent, who is looking to move to a safe and desirable place to raise her child, would use the Application to see and compare the crime rate and school rankings in the areas of interest.

*Diagram 1* depicts the key actors and components of the Application.

## *Diagram 1 - System Overview*

[](https://www.draw.io/#G1nqSkvUCma-uFmSX96F2KdXn3Pjgeq0vn)

The Application will consist of two components:

## *Table 3 - Application Components*

|  |  |
| --- | --- |
| **Component** | **Description** |
| **Web Client** | Frontend web application that allow a user to interactively identify a Map, and desired Statistics. It will dynamically update the Map with a visualization of the selected Map and selected Statistics Options |
| **Statistics Sources** | Backend server application that provides aggregated Statistical Data (to the Web Client), such as crime, income, commute, etc., from various trusted Statistic sources. This server encapsulates all data sources and converts them into a common, normalized format |

The Application will have of four primary actors :

## *Table 4 - Application Actors*

|  |  |
| --- | --- |
| **Actor** | **Description** |
| **User** | Will interact with the Web Client to select the Map and Statistics Option she wishes to see visualized |
| **Browser** | Will host the Application and provide input and output. Specifically, the browser will display the Map and a representation of the Statistical Data, etc. |
| **Map Service** | An external map service (Google Maps) that will provide the mapping capabilities and Map Data to the Web Client |
| **Statistic Services** | Publicly accessible Statistical Data sources that will return data for the user selected Map. *See Appendix for list of sources under consideration* |

The User will interact with the Web Client, which will present a Map and Statistics Options. As the User makes changes to the Map and/or chooses a Statistics Option, the Web Client will communicate with the Statistics Server to get updated Statistical Data for the selected Map. The Web Client will then redraw the Map to include Statistical Data based on the User’s selections. The Map Service will provide Map Data. The Statistics Sources will provide Statistics Data.

A detailed look at the user requirements of the Application and the corresponding functional and non-functional system requirements can be found in the Software Requirements Specification (SRS) document (CMSC447\_Team3\_CodePods\_SRS).

# 

# Project Management Process

## Team Roles and Responsibilities

The Team will function as an Agile development team. They will organize their work effort dynamically as they execute all phases of the Application SLDC. They will participate in all phases and will define and adjust roles and responsibilities as needed based on the iteration or phase that is being worked on.

## Project Plan and Deliverables

### *Table 5 - Project Plan*

|  |  |  |  |
| --- | --- | --- | --- |
| **Phases** | **Activity** | **Deadline** | **Deliverables** |
| Kickoff | Meet with Customer and Gather Business Reqs | ASAP |  |
| Prep Environment | Build Environments & Identify Tools | Week of 3/25 |  |
| Analysis | Identify Requirements | Week of 3/25 | SRS & SDP |
| Design | Architecture and Design | Week of 4/1 | SDD |
| Implementation | Develop Application to SDD specifications | 3/25-5/6 | STD |
| Testing | Testing to plan outlined in STD | 3/25-5/6 | STR |
| Customer Demo | Present and Demo Project to Customer | 5/10 | SUM & APP |

### 

### *Table 6 - Project Deliverables*

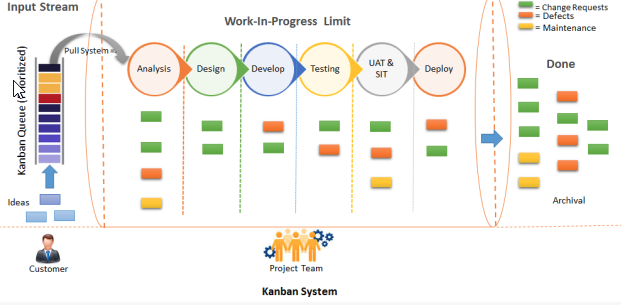
|  |
| --- |
| **Deliverables Key** |
| (SRS) Software Requirements Specification |
| (SDP) Software Development Plan |
| (SDD) Software Design Description |
| (STD) Software Test Description |
| (STR) Software Test Report |
| (SUM) Software User Manual |
| (APP) The Application |

# 

# Software Development Lifecycle

The team will follow an Agile approach based on the Kanban framework. This methodology requires real time communication and total transparency among the development team.

## Diagram 2 - Kanban Tracking Board (example)

***(https://agilegnostic.wordpress.com/2015/09/13/lean-kanban-methodology-to-application-support-and-maintenance/)***

The Kanban process allows team members to “pull” work from one phase of the lifecycle to another. Starting from analysis all the way to deployment.

The team will cycle through several iterations, with each iteration targeting certain requirements and resulting in a working Application each time that can be shared with the Customer for feedback.

The team will use JIRA, an agile based project management tool by Atlassian corporation. The tool represents the work items visually so that the team has full visibility into the status of all work items.

For a more detailed understanding of Kanban, refer to: <https://www.atlassian.com/agile/kanban>

## Requirements Analysis

Requirements Analysis will follow the Requirements Analysis Process (Table 7) and will be documented in the Software Requirements Specification (SRS) document - (CMSC447\_Team3\_CodePods\_SRS).

Following an Agile SDLC, the requirements process will be iterative and the requirements will/can evolve over the development lifecycle. The SRS will be updated as requirements change and will be used to communicate the Team’s understanding of the requirements to the Customer.

### 

### Table 7 - Requirements Analysis process

|  |  |
| --- | --- |
| ***Step*** | ***Activity Description*** |
| 1 | Customer interview to gather user requirements |
| 2 | Develop User Stories based on Customer interview(s) |
| 3 | Develop Use Cases based on User Stories to identify key system Actors, Components, and key Behaviors |
| 4 | Develop Use Case Specifications that detali step by step requirements for the identified Use Cases, including pre/post conditions and testing criteria |
| 5 | Update the SRS and communicate changes to the Team (and the Customer as needed) |

## Software Design

Software Design for the Application will follow the The Software Design Process (Table 8) and will be documented in the Software Design Description (SDD) document - (CMSC447\_Team3\_CodePods\_SDD). The SDD will evolve with and be based on the SRS.

Following an Agile SDLC, the design process will be iterative and the design will/can evolve over the development lifecycle. The SDD will be updated when design aspects of the Application change, and the SDD will be used to communicate design decisions to the Team and to the Customer.

### Table 8 - Software Design Process

|  |  |
| --- | --- |
| ***Step*** | ***Activity Description*** |
| 1 | Finalize and refine Use Case related artifacts (created during Requirements Analysis) |
| 2 | Identify System Components and create a System Model |
| 3 | Identify The System Structure and create Class Model |
| 4 | Identify System Behaviors and Interactions and create Sequence and State Models |
| 5 | Update the SDD and communicate changes to the Team (and the Customer as needed) |

## 

## Software Implementation

Software Implementation for the Application will follow the The Software Implementation Process (Table 9). As part of Implementation a Test Plan will be created and documented in the Software Test Description (STD) document - (CMSC447\_Team3\_CodePods\_STD). The STD will evolve with and be based on the SRS.

Following an Agile SDLC, the implementation process will be iterative and the Application will be built be produced incrementally. This approach will allow the Team to build working versions of the Application that can be shared with the Customer to get feedback to validate the design and solution and allow for rapid changes. The STD will be updated as the Application is built to reflect the Test Plan. The STD will be used to communicate the Test Planto the Team and to the Customer.

### Table 9 - Software Implementation Process

|  |  |
| --- | --- |
| ***Step*** | ***Activity Description*** |
| 1 | Refine requirements and update the SRS as needed |
| 2 | Refine the software design and update the SDD as needed |
| 3 | Create a backlog and refine a backlog of features based on the SRS |
| 4 | Implement features using a 1 week iteration cadence |
| 5 | Commit changes to the repository and build the application in preparation for integration and regression testing |
| 6 | Create and execute tests, refine the test plan, and update the STD. |
| 7 | Communicate Status to the Team during routine, planned, review session, and as needed. |

## Software Testing

Software Testing for the Application will follow the The Software Testing Process (Table 10). As the Application is implemented the Test Plan will evolve and will be documented in the Software Test Description (STD) document - (CMSC447\_Team3\_CodePods\_STD). The STD will be used by the Team to define the tests needed to assure the Application is meeting defined requirements, both functional and non-functional.

As testing is performed, results will be recorded and evaluated against the STD. Results will be recorded in the Software Test Report (STR) document - (CMSC447\_Team3\_CodePods\_STR).

Following an Agile SDLC, the testing process will be iterative and done simultaneously with implementation. Prior to delivery of the Application to the customer, the Application will be tested vigorously and the results will be recorded in the final STR. The STR will be used to communicate Test Results to the Team and to the Customer.

### Table 10 - Software Testing Process

|  |  |
| --- | --- |
| ***Step*** | ***Activity Description*** |
| 1 | Refine the test plan and update the STD as requirements, design and implementation evolve |
| 2 | Refine and execute Unit tests on lower level components as they are implemented |
| 3 | Pull and build test ready versions of the Application from repository |
| 4 | Refine and execute Integration and Regression tests to ensure the Application components are integrated properly and all new and previously implemented features are working correctly as defined by the SRS and STD |
| 5 | Record test results in the STR and provide feedback to the Team (and the Customer as needed) |

## Delivery

The Team will provide routine updates to the Customer throughout the SDLC of the Application as prescribed by the Customer. Final delivery of the Application will be done at the end of the CMSC447 Spring 2018 semester and will take on the form of an Application Demo to the Customer and Professor, including a presentation by the Team.

As the Application is built and features are hardened, the Team will create and update a Software User Manual (SUM) - (CMSC447\_Team3\_CodePods\_STD), which will be delivered to the Customer.

See Table 5 (Project Plan) and Table 6 (Project Deliverables) for details.

# 

# Configuration Management

For detailed descriptions of all tools mentioned in this section, refer to Table 11 - Development and CM Tools.

All Tools are open source and/or free for public use.

## 

## Table 11 - Development and Configuration Management Tools

|  |  |  |
| --- | --- | --- |
| ***Tool*** | ***Description*** | ***Reference*** |
| Ubuntu 16.04 | Open Source Linux Operating System | https://www.ubuntu.com/ |
| Git | Open Source distributed version control system | https://git-scm.com/ |
| GitHub | Free hosting platform for version control and collaboration | https://github.com/ |
| Vagrant | Tool for building and managing virtual machine environments | https://www.vagrantup.com/ |
| VirtualBox | Free desktop virtualization platform | https://www.virtualbox.org/ |
| Heroku | Cloud based Platform as a Service (PaaS) provider for hosting applications | https://www.heroku.com/ |
| Mocha | JavaScript test framework | https://mochajs.org/ |
| Chai | Behavior and Test Driven Development (BDD & TDD) frameworks | http://www.chaijs.com/ |
| Yarn | Package and Dependency Management Tool | https://yarnpkg.com/en/ |
| Node.js | JavaScript runtime | https://nodejs.org/en/ |
| Express.js | Web Application Framework | https://expressjs.com/ |
| React.js | Javascript library for building user interfaces | https://reactjs.org/ |
| Semantec UI | Application development framework | https://semantic-ui.com/ |
| Google Maps API | API enabling location-based applications | https://bit.ly/1hFiprI |
| Googe-Map-React | Google Map react component | https://bit.ly/2DReytU |
| Data.gov | API/Tools to build statical data applications | https://www.data.gov/ |
| Style Guide | Team will follow the Airbnb javascript Style Guide | https://bit.ly/1b5Jwtv |
| esLint | Source code analyzer used to flag style and other programming errors | https://eslint.org/ |
| Contributing.md | Documents additional style guidelines, change management rules, and general code of conduct for the Team | https://bit.ly/2G3NPAg |
| Apache 2.0 | A permissive license that outlines the Applications source code distribution and usage rights | https://bit.ly/29AqRwb |
| JIRA | An Agile project planning tool and issue tracker | https://bit.ly/JYWkL8 |

## Development Environment

The development and application environments will hosted on virtualized instances of Ubuntu, a flavor of Linux. There will be two instances to host the Client and Server (See Table 3 - Application Components).

## Version Control

The Team will use GIT as the software repository for all application and configuration artifacts. Team members will commit changes into the repository as features are completed so they can be distributed to other team members, can be built, and tested.

The following version semantics will be used:

1. MAJOR version when the Team makes incompatible API changes
2. MINOR version when the Team adds functionality in a backwards-compatible manner
3. PATCH version when the Team makes backwards-compatible bug fixes.

The Team will also use Google Docs as the working area to create documentation. As versions of the documents are completed, they will be committed to Git as well.

## Testing Framework

The Team will follow a Test Driven Development methodology and will use the *Mocha* and *Chai* testing frameworks. For unit and regression testing. Tests will be executed using *Yarn*.

### 

## 

## 

## Build and Deployment

The Application components will be written in *Java* and will run on the *Node.js* java runtime and will be built on the *Express.js* application framework

The Team will use *React.js* amd *Semantic UI* to build the user interface. The Application will also use *Google Maps API* and the *Google-map-reac*t component to create and implement the mapping features.

The Application will also interface with data sources from D*ata.gov* to get statistical data related to map areas.

Working versions of the Application will be deployed to *Heroku* for testing and final deployment.

## Coding Style and Code of Conduct

The Team will follow the *Airbnb Javascript Style Guide*  and it will use *esLint* to analyze code against the established style guidelines.

Additional guidelines for coding style, change management process rules and general code of conduct are documented on the *Contributing.md* document located in the Application's Git Repository.

## Licensing

This project is licensed under the Apache 2.0 License. (<http://www.apache.org/licenses/>)