ZipCompare

Software Development Plan (SDP)

[1 Scope 6](#_gjdgxs)

[1.1 Identification 6](#_30j0zll)

[1.2 System overview 6](#_1fob9te)

[1.3 Document overview 6](#_3znysh7)

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

[2 Referenced documents 6](#_3dy6vkm)

[3 Overview of required work 6](#_4d34og8)

[4 Plans for performing general software development activities 7](#_2s8eyo1)

[4.1 Software development process 7](#_3rdcrjn)

[4.2 General plans for software development 7](#_35nkun2)

[4.2.1 Software development methods 7](#_1ksv4uv)

[4.2.2 Standards for software products 7](#_2jxsxqh)

[4.2.3 Reusable software products 8](#_1y810tw)

[4.2.4 Handling of critical requirements 8](#_1ci93xb)

[4.2.5 Computer hardware resource utilization 8](#_3as4poj)

[4.2.6 Recording rationale 8](#_1pxezwc)

[4.2.7 Access for acquirer review 8](#_49x2ik5)

[5 Plans for performing detailed software development activities 9](#_147n2zr)

[5.1 Project planning and oversight 9](#_23ckvvd)

[5.1.1 Software development planning (covering updates to this plan) 9](#_ihv636)

[5.1.2 CSCI test planning 9](#_32hioqz)

[5.1.3 System test planning 9](#_1hmsyys)

[5.1.4 Software installation planning 9](#_41mghml)

[5.1.5 Software transition planning 9](#_2grqrue)

[5.1.6 Following and updating plans, including the intervals for management review 9](#_vx1227)

[5.2 Establishing a software development environment 9](#_1v1yuxt)

[5.2.1 Software engineering environment 9](#_4f1mdlm)

[5.2.2 Software test environment 9](#_2u6wntf)

[5.2.3 Software development library 9](#_19c6y18)

[5.2.4 Software development files 9](#_3tbugp1)

[5.2.5 Non-deliverable software 9](#_28h4qwu)

[5.3 System requirements analysis 9](#_37m2jsg)

[5.3.1 Analysis of user input 10](#_1mrcu09)

[5.3.2 Operational concept 10](#_46r0co2)

[5.3.3 System requirements 10](#_2lwamvv)

[5.4 System design 10](#_3l18frh)

[5.4.1 System-wide design decisions 10](#_206ipza)

[5.4.2 System architectural design 10](#_4k668n3)

[5.5 Software requirements analysis 10](#_1egqt2p)

[5.6 Software design 10](#_2dlolyb)

[5.6.1 CSCI-wide design decisions 10](#_3cqmetx)

[5.6.2 CSCI architectural design 10](#_1rvwp1q)

[5.6.3 CSCI detailed design 10](#_4bvk7pj)

[5.7 Software implementation and unit testing 10](#_2r0uhxc)

[5.7.1 Software implementation 10](#_1664s55)

[5.7.2 Preparing for unit testing 10](#_3q5sasy)

[5.7.3 Performing unit testing 10](#_25b2l0r)

[5.7.4 Revision and retesting 10](#_kgcv8k)

[5.7.5 Analyzing and recording unit test results 10](#_34g0dwd)

[5.8 Unit integration and testing 11](#_43ky6rz)

[5.8.1 Preparing for unit integration and testing 11](#_2iq8gzs)

[5.8.2 Performing unit integration and testing 11](#_xvir7l)

[5.8.3 Revision and retesting 11](#_3hv69ve)

[5.8.4 Analyzing and recording unit integration and test results 11](#_1x0gk37)

[5.9 CSCI qualification testing 11](#_2w5ecyt)

[5.9.1 Independence in CSCI qualification testing 11](#_1baon6m)

[5.9.2 Testing on the target computer system 11](#_3vac5uf)

[5.9.3 Preparing for CSCI qualification testing 11](#_2afmg28)

[5.9.4 Dry run of CSCI qualification testing 11](#_pkwqa1)

[5.9.5 Performing CSCI qualification testing 11](#_39kk8xu)

[5.9.6 Revision and retesting 11](#_1opuj5n)

[5.9.7 Analyzing and recording CSCI qualification test results 11](#_48pi1tg)

[5.10 CSCI/HWCI integration and testing 11](#_1302m92)

[5.10.1 Preparing for CSCI/HWCI integration and testing 11](#_3mzq4wv)

[5.10.2 Performing CSCI/HWCI integration and testing 11](#_2250f4o)

[5.10.3 Revision and retesting 11](#_haapch)

[5.10.4 Analyzing and recording CSCI/HWCI integration and test results 11](#_319y80a)

[5.11 System qualification testing 11](#_40ew0vw)

[5.11.1 Independence in system qualification testing 12](#_2fk6b3p)

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

[5.11.3 Preparing for system qualification testing 12](#_3ep43zb)

[5.11.4 Dry run of system qualification testing 12](#_1tuee74)

[5.11.5 Performing system qualification testing 12](#_4du1wux)

[5.11.6 Revision and retesting 12](#_2szc72q)

[5.11.7 Analyzing and recording system qualification test results 12](#_184mhaj)

[5.12 Preparing for software use 12](#_279ka65)

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

[5.12.2 Preparing version descriptions for user sites 12](#_36ei31r)

[5.12.3 Preparing user manuals 12](#_1ljsd9k)

[5.12.4 Installation at user sites 12](#_45jfvxd)

[5.13 Preparing for software transition 12](#_zu0gcz)

[5.13.1 Preparing the executable software 12](#_3jtnz0s)

[5.13.2 Preparing source files 12](#_1yyy98l)

[5.13.3 Preparing version descriptions for the support site 12](#_4iylrwe)

[5.13.4 Preparing the "as built" CSCI design and other software support information 12](#_2y3w247)

[5.13.5 Updating the system design description 12](#_1d96cc0)

[5.13.6 Preparing support manuals 12](#_3x8tuzt)

[5.13.7 Transition to the designated support site 12](#_2ce457m)

[5.14 Software configuration management 13](#_3bj1y38)

[5.14.1 Configuration identification 13](#_1qoc8b1)

[5.14.2 Configuration control 13](#_4anzqyu)

[5.14.3 Configuration status accounting 13](#_2pta16n)

[5.14.4 Configuration audits 13](#_14ykbeg)

[5.14.5 Packaging, storage, handling, and delivery 13](#_3oy7u29)

[5.15 Software product evaluation 13](#_j8sehv)

[5.15.1 In-process and final software product evaluations 13](#_1idq7dh)

[5.15.2 Software product evaluation records, including items to be recorded 13](#_42ddq1a)

[5.15.3 Independence in software product evaluation 13](#_2hio093)

[5.16 Software quality assurance 13](#_wnyagw)

[5.16.1 Software quality assurance evaluations 13](#_3gnlt4p)

[5.16.2 Software quality assurance records, including items to be recorded 13](#_1vsw3ci)

[5.16.3 Independence in software quality assurance 13](#_4fsjm0b)

[5.17 Corrective action 13](#_1a346fx)

[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](#_3u2rp3q)

[5.17.2 Corrective action system 14](#_2981zbj)

[5.18 Joint technical and management reviews 14](#_38czs75)

[5.18.1 Joint technical reviews, including a proposed set of reviews 14](#_1nia2ey)

[5.18.2 Joint management reviews, including a proposed set of reviews 14](#_47hxl2r)

[5.19 Other software development activities 14](#_11si5id)

[5.19.1 Risk management, including known risks and corresponding strategies 14](#_3ls5o66)

[5.19.2 Software management indicators, including indicators to be used 14](#_20xfydz)

[5.19.3 Security and privacy 14](#_4kx3h1s)

[5.19.4 Subcontractor management 14](#_302dr9l)

[5.19.5 Interface with software independent verification and validation (IV&V) agents 14](#_1f7o1he)

[5.19.6 Coordination with associate developers 14](#_3z7bk57)

[5.19.7 Improvement of project processes 14](#_2eclud0)

[5.19.8 Other activities not covered elsewhere in the plan 14](#_thw4kt)

[6 Schedules and activity network 14](#_1smtxgf)

[7 Project organization and resources 15](#_16x20ju)

[7.1 Project organization 15](#_3qwpj7n)

[7.2 Project resources 15](#_l7a3n9)

[8 Notes 15](#_1kc7wiv)

[9 A. Appendixes 15](#_2jh5peh)

# Scope

## Identification

ZipCompare (ZC) Web Application v. 1.0 release 0 (Development)

University of Maryland Baltimore Country (UMBC)

## System overview

The purpose of ZC is to allow users to obtain a comparison between up to three zip codes that the user defines, via map clicks or direct text box input. Comparisons are made using a formula which rates quality of life within those zip codes based upon criteria such as crime rate, education, and average income [UPDATE THIS WHEN CRITERIA IS FINALIZED]. Users will be able to create accounts, make comparisons, compile a list of favorite zip codes [GET HOW MANY THEY CAN HAVE], and upvote their home zip code. To date, the map page itself and database are under development.

The sponsor for this project is Michael Neary. ZC is being developed by Dustin Cuocci, Damon Streat, Benjamin Decre, Keith McNamara, William Penas, and Mitchell Borrego. Development operations will take place on the UMBC campus and in the personal offices of each member on the development team.

Other relevant documents on this project include the Software Design Description, Requirements Specification, Test Description, Test Report, and User Manual.

## Document overview

The purpose of this document is to fully describe the processes by which ZC is being developed, and its contents should be accessible only to the development team and the sponsor.

## Relationship to other plans

The ZC Development Plan has no relation to other Software Development Plans.

# 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.

ZC Design Description v 1.0

ZC Requirements Specification

ZC Test Description v 1.0

ZC Test Report v 1.0

ZC User Manual v 1.0

# Overview of required work

Required documentation for ZC include this document, and all documents referenced in section 2 of this document. The highest level of detail shall be used where applicable to ensure clarity and transparency of the development effort.

ZC has just begun the early stages of development.

The development team is utilizing a hybrid Waterfall/AGILE strategy to ensure timely deliveries to meet the requirements documented in the ZC Requirements Specification. Updates made to the source code will be tracked and managed via Github.

There are no restrictions on project resources, but deliveries will be made to the sponsor periodically as agreed to by the sponsor.

# Plans for performing general software development activities

## Software development process

The Waterfall/AGILE hybrid process employed by the ZC development team includes weekly targeted meetings with daily check-ins via Slack to communicate progress on assigned tasks. Development has already begun on a barebones front-end implementation to serve as framework for subsequent development activities which shall proceed in the below order:

1. Front End Framework
2. Database Design and Implementation
3. Server Design and Implementation
4. Front End Development
5. Testing
6. Documentation Finalization

## General plans for software development

### Software development methods

The ZC development team will use an iterative method to progress through the development process as defined in section 4.1. All testing prior to release will be conducted manually via test cases.

### Standards for software products

The ZC development team will maintain a standard of 2 spaces for indentation between

nested lines of code, and 1 row of whitespace after brackets where applicable, for

instance after a large code block. Excessive whitespace shall be avoided.

The following example is the standard for header comments in all files:

ZipCompare v. 1.0

File Name: map.html

Description: This file contains the code used

to display the front end user interface of the

ZipCompare application. ZC uses a combination of

html, Javascript, and CSS in its functionality.

Last Updated: 4/5/2018

For comments dispersed throughout the code body of a given file, emphasis will be placed on describing reasoning over semantics of a given function. More advanced features or parts of the Google API will be commented for efficiency on the part of the reader. A standard of at least 2 lines of comments per 15 lines of code shall be maintained.

-

### Reusable software products

#### Incorporating reusable software products

ZC is being developed using Google Maps, MongoDB, and contains features such as

searching a locality, and parsing the data collected by a web crawler, which bolster the

application’s reusability.

#### Developing reusable software products

This effort is aimed at creating a reusable zip code data layer that is reusable, such that

perhaps endeavors outside this project might find it helpful. The data layer was created from government-reported Zip Code Tabulation Areas (ZCTAs), using ArcGIS for drawing and outputting it in a format suitable for the Google Maps API to display.

### Handling of critical requirements

#### Safety assurance

There are no safety considerations designated critical in the requirements for this project.

#### Security assurance

Every effort to protect user and application data is being taken, including account and database

encryption throughout the development of ZC.

#### Privacy assurance

ZC developers personally guarantee that no personal information offered by users or

administrators will be shared without their express permission.

#### Assurance of other critical requirements

There are no further critical requirements.

### Access for acquirer review

All acquirer reviews will periodically take place on the UMBC campus at locations

agreed upon by the acquirer and development team, likely in the ITE building.

# Plans for performing detailed software development activities

## Project planning and oversight

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

At each weekly meeting, recently completed and upcoming tasks will be evaluated and

meeting minutes recorded and deposited on the development team share drive for review.

Preparation of deliverables will also be conducted during this meeting, as well as bug reporting.

Bugs that require action items will also be recorded and assigned.

### CSCI test planning

As new features which meet CSCI requirements are implemented, a test of that feature will be

planned.

### System test planning

System-wide testing is scheduled to begin after the primary development phase has concluded,

on approximately 5/1/2018.

### Software installation planning

Not applicable.

### Software transition planning

Not applicable.

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

All documentation of the ZC project is living, but management will hold official progress reviews

approximately every two weeks.

## 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

All members of the ZC development team are responsible for cloning the github repository

([git@github.com](mailto:git@github.com):jatmdm/cmsc447project.git), and creating their own instance of the

environment. This will consist of not only cloning the repository, but also activating their own

UMBC student webspace (<https://wiki.umbc.edu/pages/viewpage.action?pageId=67994741>).

### Software test environment

The software test environment is the UMBC student webspace, controlled and maintained by UMBC.

### Software development library

No libraries are currently used by the ZC application.

### Software development files

Github will be used to maintain all files related to ZC development.

### Non-deliverable software

Github will be used to maintain the files for all non-deliverable software, including database and

webcrawler code.

## System requirements analysis

### Analysis of user input

Initial sponsor meetings to discuss the requirements have been concluded, with the

requirements compiled in the ZC Requirements Specification. The user input of hard and soft

requirements has guided the initial development phase.

### Operational concept

Iterations upon the operational concept are ongoing as requirements are updated, added, and

removed. Any requirements changes are addressed at the weekly meeting, and tasked to an

available developer.

### System requirements

The development team is constantly evaluating features which meet or that are still needed to

meet system requirements. Meeting the requirements and changing or removing vague or

unrealistic requirements is a top priority.

## System design

### System-wide design decisions

Any decisions to be made at a system-wide level require a reasonable level of discussion, which

will extend beyond the scope of the weekly meeting and into daily Slack conversations. Once a

consensus is reached about a design decision, implementation can resume.

### System architectural design

The approach to system-wide design decisions (Section 5.4.1) also applies to architectural

design.

## Software requirements analysis

The development team is constantly evaluating features which meet or that are still needed to

meet software requirements. Meeting the requirements and changing or removing vague or

unrealistic requirements is a top priority.

## Software design

### CSCI-wide design decisions

Any decisions to be made at a CSCI-wide level require a reasonable level of discussion, which

will extend beyond the scope of the weekly meeting and into daily Slack conversations. Once a

consensus is reached about a design decision, implementation can resume.

### CSCI architectural design

The approach to CSCI-wide design decisions (Section 5.6.1) also applies to architectural

design.

### CSCI detailed design

The approach to CSCI-wide design decisions (Section 5.6.1) also applies to architectural

design.

## 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

ZC is being developed in an AGILE environment, with weekly updates to the framework in phase

1 and nearly daily updates planned during primary development phases (2-4).

### Preparing for unit testing

Before unit testing begins, a review of all features (with emphasis on new ones) will be

conducted. Once this review is complete, existing test cases will be updated or new cases

developed to to cover the new features and updates to existing features. Unit testing has been

expanded for ZC to include unit integration testing.

### Performing unit testing

All features will be fully tested per the unit testing specification.

### Revision and retesting

Any vagueness,inconsistencies or errors in the unit test will be discussed among the

development team to avoid any issues for the next round of unit testing.

### Analyzing and recording unit test results

Bugs or results encountered during unit testing will be documented and incorporated into the

Github repository for the project.

## Unit integration and testing

Unit integration testing will be covered in during standard unit tests. Please see section 5.75 for

further details.

### 

## CSCI qualification testing

### CSCI qualification testing will be scheduled as features meeting requirements are put

into production. Before considering a requirement satisfied, all stakeholders including the

sponsor and development team must grant their approval.

## CSCI/HWCI integration and testing

### There is no requirement for Hardware Configuration in this project.

## System qualification testing

### Independence in system qualification testing

All system units will be tested periodically and independent of one another.

### Testing on the target computer system

ZC will be tested on a low-powered notebook PC to give a reasonable performance baseline.

### Preparing for system qualification testing

Full system test cases will be developed that encapsulate a full user experience of all features in

the application.

### Dry run of system qualification testing

Immediately prior to final delivery, a dry run of the system will be conducted with only the

development team present.

### Performing system qualification testing

System qualification testing will be a component of final product delivery to the sponsor.

### Revision and retesting

A revision and retest is not a requirement of this project.

### Analyzing and recording system qualification test results

Test results will be recorded and committed to the GitHub repository for this project.

## Preparing for software use

### Preparing the executable software

An executable is not a requirement of this project.

### Preparing version descriptions for user sites

A version description will be a component of the software user manual.

### Preparing user manuals

Following the pre-delivery test phase, a user manual will be created by the development team

to assist new users with the software. However, ZC is being developed with an emphasis on

intuitive behaviour.

### Installation at user sites

User installation is not a requirement of this project.

## Preparing for software transition

### Software transitioning is not a requirement of this project.

## 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

ZC configuration management will be facilitated by GitHub.

### Configuration control

All members of the development team will coordinate to ensure that they are working with the

latest configuration, and committing regular updates to the team repository.

### Configuration status accounting

Configuration status will be addressed in each weekly meeting, as well as configuration

audits.

### Configuration audits

Audits will occur during the weekly development team meeting.

### Packaging, storage, handling, and delivery

Packaging, storage, handling, and delivery are not a requirement of this project.

## Software product evaluation

### In-process and final software product evaluations

The development team and sponsor will work collaboratively to evaluate the software

in-process. The final evaluation of ZC will be delivered by Professor Russell Cain.

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

After each weekly check-in, a record of key points in the meeting between the development

team and sponsor will be maintained.

### Independence in software product evaluation

Professor Russell Cain will perform an independent product evaluation for ZC.

## 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

QA will be performed during the Testing development phase, and may necessitate further

fixes and added Testing phase.

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

Any issues discovered during QA will be documented and committed to the GitHub repository

for ZC.

### Independence in software quality assurance

Independent QA will not be performed for ZC. Only development team members will perform

this task.

## 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)

Problem reports will be handled similarly to problems encountered during QA events (see

section 5.16.2), but no changes to the application will be made after final product delivery.

### Corrective action system

Corrective action items will be assigned to development team members as needed should

problems arise during QA and Testing events.

## Joint technical and management reviews

No joint technical and management reviews will be held as a part of this project.

## Other software development activities

### All software development activities are already covered in other areas of this document.

# Schedules and activity network

1. A schedule of deliverables for each development phase will be maintained and updated when those deliverables are added to production.
2. The front end map page will require a large amount of development time, and a

close network between front end developers Dustin Cuocci and Damon Streat will

be established to ensure timely delivery.

# Project organization and resources

## Project organization

The development team for ZC is composed of Computer Science students at UMBC,

with Professor Russell Cain as the final reviewer and Michael Neary as the sponsor.

Professor Cain is independent of the ZC developers and Michael Neary. The

development team relies upon Mr. Neary for regular feedback to guide fixes and the

addition of new features to ZC.

## Project resources

The project resources include 6 software developers each furnishing their own

equipment to facilitate the development process. Their responsibilities are as

follows:

Front End Development: Dustin Cuocci & Damon Streat

Back End Development: Mitchell Borrego, Keith McNamara, William Penas

Database and DB-Application Integration: Benjamin Decre & William Penas

# Notes

Any applicable notes to be added at a future date.Appendixes

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.