ZipCompare

Software Design Description (SDD)

[**Scope**](#_gjdgxs) **3**

[Identification and Revision History](#_2et92p0) 3

[System overview](#_tyjcwt) 3

[Document overview](#_3dy6vkm) 3

[Reference Documents](#_pp55inw418jb) 4

[**CSCI-wide design decisions**](#_17dp8vu) **4**

[**CSCI architectural design**](#_26in1rg) **4**

[CSCI components](#_35nkun2) 5

[Concept of execution](#_z337ya) 6

[Interface design](#_4i7ojhp) 6

[Interface identification and diagrams](#_2xcytpi) 6

[(Project-unique identifier of interface)](#_3whwml4) 6

[**CSCI detailed design**](#_qsh70q) **8**

[(Project-unique identifier of a software unit, or designator of a group of software units)](#_1pxezwc) 8

[**Requirements traceability**](#_2p2csry) **9**

[**Notes**](#_23ckvvd) **9**

[**Appendixes**](#_ihv636) **10**

# Scope

This section shall be divided into the following paragraphs.

## Identification and Revision History

ZipCompare Revision History

|  |  |  |
| --- | --- | --- |
| Date | Version | Description |
|  |  |  |
|  |  |  |

At the moment, ZipCompare has no release versions available.

This system is the product resulting from the University of Maryland, Baltimore County (UMBC) Computer Science 447: Software Engineering I course, to produce a predetermined product for an assigned customer.

## System overview

The ZipCompare system consists of software meeting the following description and fulfilling the specified requirements:

ZipCompare allows the user to use a map program or text input to find a zip code and access information about that zipcode, compare that zip code to other zip codes, and save/”star” zip codes for later information retrieval. Additionally, ZipCompare will have a user account system that allows users with accounts to “upvote” certain areas as a rating system for zip codes. Finally, ZipCompare will take the rating data of the users as well as information about the zip code to create a Top 5 listing of the most popular and highest rated zip codes.

To fufill these requirements

## Document overview

The purpose of this document is to have a manual for which we can document the design considerations, blocks, problems, and solutions so that we can better develop ZipCompare. This document will possibly contain information that exposes the underlying systems of ZipCompare, so it is a very critical security concern. There is a possibility that exposing this document could open our system up to hackers.

## 1.4 Reference Documents

The documents referenced in the rest of this Software Design Description (SDD) are:

* Software Requirements Specification (SRS)

# CSCI-wide design decisions

This section shall be divided into paragraphs as needed to present CSCI-wide design decisions, that is, decisions about the CSCI’s behavioral design (how it will behave, from a user’s point of view, in meeting its requirements, ignoring internal implementation) and other decisions affecting the selection and design of the software units that make up the CSCI. If all such decisions are explicit in the CSCI requirements or are deferred to the design of the CSCI’s software units, this section shall so state. Design decisions that respond to requirements designated critical, such as those for safety, security, or privacy, shall be placed in separate subparagraphs. If a design decision depends upon system states or modes, this dependency shall be indicated. Design conventions needed to understand the design shall be presented or referenced. Examples of CSCI-wide design decisions are the following:

1. The design will be a “point and click” application using HTML and Javascript.
   * Users will be able to choose a zip code by clicking an object on the HTML page, including general HTML objects and a google map embedded into the page.
   * Users will also be able to choose zip codes via typing the zip code into a textbox.
   * Due to the above requirements, usage of keyboard and mouse are definitely required.
2. Upon using one of the methods for choosing a zip code, the page will direct the map toward the location of the zip code, and bring up information about this zip code.
   * The information will be displayed in a sidebar alongside the map.
3. It is critical that none of the information in the database is directly visible to the user. All data must be abstracted out from the database.
   * This is to prevent exposing the underlying system to hackers. By doing this we hope to prevent hackers from being able to destroy the integrity of our data.
4. To maintain security for ZipCompare, we are going to password protect Administrator accounts on our website, and encrypt critical information within our database.
5. Other CSCI-wide design decisions made in response to requirements, such as selected approach to providing required flexibility, availability, and maintainability.

# CSCI architectural design

This section shall be divided into the following paragraphs to describe the CSCI architectural design. If part or all of the design depends upon system states or modes, this dependency shall be indicated. If design information falls into more than one paragraph, it may be presented once and referenced from the other paragraphs. Design conventions needed to understand the design shall be presented or referenced.

## CSCI components

This paragraph shall:

1. Identify the software units that make up the CSCI. Each software unit shall be assigned a project-unique identifier.

Note: A software unit is an element in the design of a CSCI; for example, a major subdivision of a CSCI, a component of that subdivision, a class, object, module, function, routine, or database. Software units may occur at different levels of a hierarchy and may consist of other software units. Software units in the design may or may not have a one- to-one relationship with the code and data entities (routines, procedures, databases, data files, etc.) that implement them or with the computer files containing those entities. A database may be treated as a CSCI or as a software unit. The SDD may refer to software units by any name(s) consistent with the design methodology being used.

1. Show the static (such as "consists of") relationship(s) of the software units. Multiple relationships may be presented, depending on the selected software design methodology (for example, in an object-oriented design, this paragraph may present the class and object structures as well as the module and process architectures of the CSCI).
2. State the purpose of each software unit and identify the CSCI requirements and CSCI- wide design decisions allocated to it. (Alternatively, the allocation of requirements may be provided in [6.a.)](#_2p2csry)
3. Identify each software unit’s development status/type (such as new development, existing design or software to be reused as is, existing design or software to be reengineered, software to be developed for reuse, software planned for Build N, etc.) For existing design or software, the description shall provide identifying information, such as name, version, documentation references, library, etc.
4. Describe the CSCI’s (and as applicable, each software unit’s) planned utilization of computer hardware resources (such as processor capacity, memory capacity, input/output device capacity, auxiliary storage capacity, and communications/network equipment capacity). The description shall cover all computer hardware resources included in resource utilization requirements for the CSCI, in system-level resource allocations affecting the CSCI, and in resource utilization measurement planning in the Software Development Plan (SDP). If all utilization data for a given computer hardware resource are presented in a single location, such as in one SDD, this paragraph may reference that source. Included for each computer hardware resource shall be:
   1. The CSCI requirements or system-level resource allocations being satisfied
   2. The assumptions and conditions on which the utilization data are based (for example, typical usage, worst-case usage, assumption of certain events)
   3. Any special considerations affecting the utilization (such as use of virtual memory, overlays, or multiprocessors or the impacts of operating system overhead, library software, or other implementation overhead)
   4. The units of measure used (such as percentage of processor capacity, cycles per second, bytes of memory, kilobytes per second)
   5. The level(s) at which the estimates or measures will be made (such as software unit, CSCI, or executable program)
5. Identify the program library in which the software that implements each software unit is to be placed.

The software components of ZipCompare are defined in our SRS.

## Concept of execution

This paragraph shall describe the concept of execution among the software units. It shall include diagrams and descriptions showing the dynamic relationship of the software units, that is, how they will interact during CSCI operation, including, as applicable, flow of execution control, data flow, dynamically controlled sequencing, state transition diagrams, timing diagrams, priorities among units, handling of interrupts, timing/sequencing relationships, exception handling, concurrent execution, dynamic allocation/deallocation, dynamic creation/deletion of objects, processes, tasks, and other aspects of dynamic behavior.

Our concept of execution is relatively simple. The main working parts to our software are our frontend systems, backend systems, and data systems. Our frontend systems are responsible for transmitting data from our system to our users. This data is retrieved from our backend systems, which store things like user data and zip code rating data. The backend system is also responsible for drawing information from our data system to come up with a rating for zip codes. The data system is where all of the information about the zip codes we have. Additionally, part of the data system is our database which will contain user account information.

## Interface design

This paragraph shall be divided into the following subparagraphs to describe the interface characteristics of the software units. It shall include both interfaces among the software units and their interfaces with external entities such as systems, configuration items, and users. If part or all of this information is contained in Interface Design Descriptions (IDDs), in section [5](#_qsh70q) of the SDD, or elsewhere, these sources may be referenced.

### Interface identification and diagrams

[Currently no diagrams]

### (Project-unique identifier of interface)

This paragraph (beginning with 4.3.2) shall identify an interface by project-unique identifier, shall briefly identify the interfacing entities, and shall be divided into subparagraphs as needed to describe the interface characteristics of one or both of the interfacing entities. If a given interfacing entity is not covered by this SDD (for example, an external system) but its interface characteristics need to be mentioned to describe interfacing entities that are, these characteristics shall be stated as assumptions or as "When [the entity not covered] does this, [the entity that is covered] will ...." This paragraph may reference other documents (such as data dictionaries, standards for protocols, and standards for user interfaces) in place of stating the information here. The design description shall include the following, as applicable, presented in any order suited to the information to be provided, and shall note any differences in these characteristics from the point of view of the interfacing entities (such as different expectations about the size, frequency, or other characteristics of data elements):

* + - 1. Priority assigned to the interface by the interfacing entity(ies)
      2. Type of interface (such as real-time data transfer, storage-and-retrieval of data, etc.) to be implemented
      3. Characteristics of individual data elements that the interfacing entity(ies) will provide, store, send, access, receive, etc., such as:
         1. Names/identifiers

Project-unique identifier

Non-technical (natural-language) name

DoD standard data element name

Technical name (e.g., variable or field name in code or database)

Abbreviation or synonymous names

* + - * 1. Data type (alphanumeric, integer, etc.)
        2. Size and format (such as length and punctuation of a character string)
        3. Units of measurement (such as meters, dollars, nanoseconds)
        4. Range or enumeration of possible values (such as 0-99)
        5. Accuracy (how correct) and precision (number of significant digits)
        6. Priority, timing, frequency, volume, sequencing, and other constraints, such as whether the data element may be updated and whether business rules apply
        7. Security and privacy constraints
        8. Sources (setting/sending entities) and recipients (using/receiving entities)
      1. Characteristics of data element assemblies (records, messages, files, arrays, displays, reports, etc.) that the interfacing entity(ies) will provide, store, send, access, receive, etc., such as:
         1. Names/identifiers

Project-unique identifier

Non-technical (natural language) name

Technical name (e.g., record or data structure name in code or database)

Abbreviations or synonymous names

* + - * 1. Data elements in the assembly and their structure (number, order, grouping)
        2. Medium (such as disk) and structure of data elements/assemblies on the medium
        3. Visual and auditory characteristics of displays and other outputs (such as colors, layouts, fonts, icons and other display elements, beeps, lights)
        4. Relationships among assemblies, such as sorting/access characteristics
        5. Priority, timing, frequency, volume, sequencing, and other constraints, such as whether the assembly may be updated and whether business rules apply
        6. Security and privacy constraints
        7. Sources (setting/sending entities) and recipients (using/receiving entities)
      1. Characteristics of communication methods that the interfacing entity(ies) will use for the interface, such as:
         1. Project-unique identifier(s)
         2. Communication links/bands/frequencies/media and their characteristics
         3. Message formatting
         4. Flow control (such as sequence numbering and buffer allocation)
         5. Data transfer rate, whether periodic/aperiodic, and interval between transfers
         6. Routing, addressing, and naming conventions
         7. Transmission services, including priority and grade
         8. Safety/security/privacy considerations, such as encryption, user authentication, compartmentalization, and auditing
      2. Characteristics of protocols the interfacing entity(ies) will use for the interface, such as:
         1. Project-unique identifier(s)
         2. Priority/layer of the protocol
         3. Packeting, including fragmentation and reassembly, routing, and addressing
         4. Legality checks, error control, and recovery procedures
         5. Synchronization, including connection establishment, maintenance, termination
         6. Status, identification, and any other reporting features
      3. Other characteristics, such as physical compatibility of the interfacing entity(ies) (dimensions, tolerances, loads, voltages, plug compatibility, etc.)

# CSCI detailed design

This section shall be divided into the following paragraphs to describe each software unit of the CSCI. If part of all of the design depends upon system states or modes, this dependency shall be indicated. If design information falls into more than one paragraph, it may be presented once and referenced from the other paragraphs. Design conventions needed to understand the design shall be presented or referenced. Interface characteristics of software units may be described here, in Section [4,](#_26in1rg) or in Interface Design Descriptions (IDDs). Software units that are databases, or that are used to access or manipulate databases, may be described here or in Database Design Descriptions (DBDDs).

## (Project-unique identifier of a software unit, or designator of a group of software units)

This paragraph shall identify a software unit by project-unique identifier and shall describe the unit. The description shall include the following information, as applicable. Alternatively, this paragraph may designate a group of software units and identify and describe the software units in subparagraphs. Software units that contain other software units may reference the descriptions of those units rather than repeating information.

* + 1. Unit design decisions, if any, such as algorithms to be used, if not previously selected
    2. Any constraints, limitations, or unusual features in the design of the software unit
    3. The programming language to be used and rationale for its use if other than the specified CSCI language
    4. If the software unit consists of or contains procedural commands (such as menu selections in a database management system (DBMS) for defining forms and reports, on- line DBMS queries for database access and manipulation, input to a graphical user interface (GUI) builder for automated code generation, commands to the operating system, or shell scripts), a list of the procedural commands and reference to user manuals or other documents that explain them
    5. If the software unit contains, receives, or outputs data, a description of its inputs, outputs, and other data elements and data element assemblies, as applicable. Paragraph [4.3.x](#_3whwml4) of this DID provides a list of topics to be covered, as applicable. Data local to the software unit shall be described separately from data input to or output from the software unit. If the software unit is a database, a corresponding Database Design Description (DBDD) shall be referenced; interface characteristics may be provided here or by referencing section [4](#_26in1rg) or the corresponding Interface Design Descriptions (IDDs).
    6. If the software unit contains logic, the logic to be used by the software unit, including, as applicable:
       1. Conditions in effect within the software unit when its execution is initiated
       2. Conditions under which control is passed to other software units
       3. Response and response time to each input, including data conversion, renaming, and data transfer operations
       4. Sequence of operations and dynamically controlled sequencing during the software unit’s operation, including:
          1. The method for sequence control
          2. The logic and input conditions of that method, such as timing variations, priority assignments
          3. Data transfer in and out of memory
          4. The sensing of discrete input signals, and timing relationships between interrupt operations within the software unit
       5. Exception and error handling

# 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.

# Appendixes

Appendixes may be used to provide information published separately for convenience in document maintenance (e.g., charts, classified data). As applicable, each appendix shall be referenced in the main body of the document where the data would normally have been provided. Appendixes may be bound as separate documents for ease in handling. Appendixes shall be lettered alphabetically (A, B, etc.).

DESCRIPTION/PURPOSE

The Software Design Description (SDD) describes the design of a Computer Software Configuration Item (CSCI). It describes the CSCI-wide design decisions, the CSCI architectural design, and the detailed design needed to implement the software. The SDD may be supplemented by Interface Design Descriptions (IDDs) and Database Design Descriptions (DBDDs).

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.