CMSC 447

Software Design Description (SDD)

[1 Scope 3](#_gjdgxs)

[1.1 Identification 3](#_2et92p0)

[1.2 System overview 3](#_tyjcwt)

[1.3 Document overview 3](#_3dy6vkm)

[2 Referenced documents 3](#_4d34og8)

[3 CSCI-wide design decisions 3](#_17dp8vu)

[4 CSCI architectural design 4](#_26in1rg)

[4.1 CSCI components 4](#_35nkun2)

[4.2 Concept of execution 5](#_z337ya)

[4.3 Interface design 5](#_4i7ojhp)

[4.3.1 Interface identification and diagrams 6](#_2xcytpi)

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

[5 CSCI detailed design 7](#_qsh70q)

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

[6 Requirements traceability 9](#_2p2csry)

[7 Notes 9](#_23ckvvd)

[A. Appendixes 10](#_ihv636)

# Scope

This section shall be divided into the following paragraphs.

## Identification

This paragraph shall contain a full identification of the system and the software to which this document applies, including, as applicable, identification number(s), title(s), abbreviation(s), version number(s), and release number(s).

The Project consists of the 4 software CSCIs listed below:

1. (DATA) Data collector: Responsible for communicating between API’s.
2. (IMD) Interactive Map Display: Responsible for displaying a map to the client-side system.
3. (FORM) Search Form: Responsible for aggregating the information given by the user containing filters and a location.
4. (WEB) Responsible for serving HTML from the server-side system to the client-side system and hosting an API to which the (FORM) sends the filters and location.

## System overview

This paragraph shall briefly state the purpose of the system and the software to which this document applies. It shall describe the general nature of the system and software; summarize the history of system development, operation, and maintenance; identify the project sponsor, acquirer, user, developer, and support agencies; identify current and planned operating sites; and list other relevant documents.

The system will be a web-based application with the ability to search for houses based off of certain search criterias in a census designated area from a map interface. The user will be any individual searching for a home in the region from Ocean Pines to the coast of Delaware. There be unit tests after each feature is added. The operating site will be the RLC at UMBC.

## Document overview

This paragraph shall summarize the purpose and contents of this document and shall describe any security or privacy considerations associated with its use.

This document describes the implementation details of The Coastal Neighborhood Finder (CNF).

# Referenced documents

This section shall list the number, title, revision, and date of all documents referenced in this document. This section shall also identify the source for all documents not available through normal Government stocking activities.

* Software Requirement Specification
* Software Design Plan
* Software Test Description
* Software Test Report
* Software User Manual

# 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.

Main software units that make up the CSCI:

**(Data) CSCI Description**

The (DATA) CSCI is composed of these software units:

* Zillow Collector (ZILL)
  + getCities()
  + getProperties()
  + getCounty()
* Yelp Collector (YELP)
  + query\_api()
  + search\_lat()
* Xml Decoder (XML)
  + makeCity()
  + getCityList()
  + getData()
* Server API (API)
  + getAmmenitiesRoute()
  + getPropertiesRoute()
  + getCitiesRoute()

The (WEB) CSCI is composed of these software units:

* Start server script (SRP)

The (IMD) CSCI is composed of these software units:

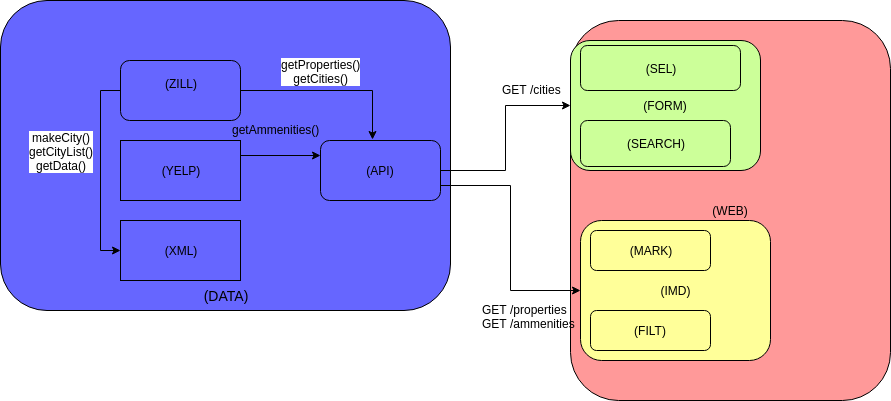
* Leaflet map (LEAF)
* Markers(MARK)
  + House Markers
    - getProperties()
    - createHouseEntry()
  + House results table
    - updateResults()
  + Amenities markers
* Filters(FILT)
  + Property filter
    - filterProperty()
  + Amenity Filter
    - filterAmmenity()

The (FORM) CSCI is composed of these software units:

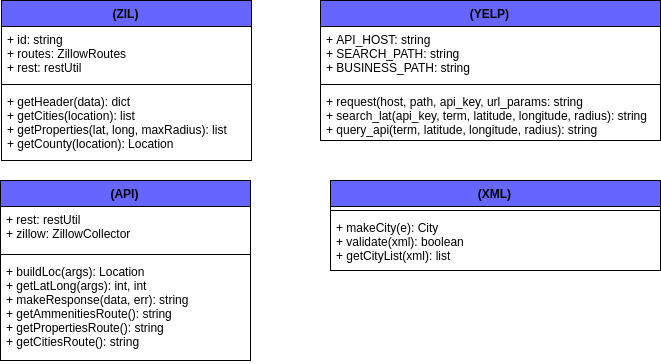
* Selectors(SEL)
  + State selectors
    - changeState()
  + County selectors
    - changeCounties()
  + City selectors
    - getCitiesRequest()
* Search (SEARCH)
  + Search button
    - searchMap()

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).

This is an overview of how the whole system works with each CSCI and their corresponding software units. Each colored box corresponds to a CSCI and will use the same coloring for future diagrams of each CSCI.

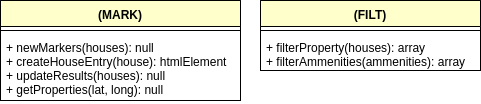


The (DATA) CSCI consists of the (ZIL), (YELP), (XML), and the (API) software units.

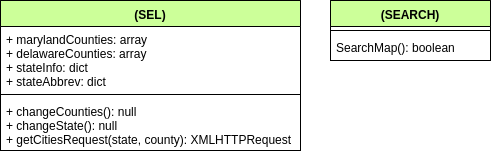


The (WEB) CSCI consists of only itself as a software unit.

The (IMD) CSCI consists of the (MARK) and the (FILT) software units.



The (FORM) CSCI consists of the (SEL) and (SEARCH) software units.



1. 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)

**Property Data Generator** - The Property Data Generator uses Zillow API to pull all necessary information about properties from Zillow website to Microsoft Excel file.

Microsoft Excel file saved as comma separated file cvs will be used as source file for generating fusion table.

Technologies Used: Visual Basic for Applications, Microsoft Excel, Zillow API.

**Fusion Table Application** (Google Drive application).

Fusion Tables is an experimental data visualization web application to gather, visualize, and share data tables. Fusion Table will use our source file (comma separated cvs) and generate the fusion table, card, and map.

* Data Management Tool - allows to see all data as a table, any row or multiple rows can be deleted, added, or edited.
* Card Editor - helps to edit layout of cards, change design, add information
* map representation - creates functional map preview, different filters can be tested

**Website builder** - utilities and customizes results of Fusion Table application to create real estate web application.

1. **Map Editor-** javascript file that specifies additional parameters and features for map that was generated with Fusion Table Application.

* initializeMap function - processes all necessary API keys, initializes location column, initializes location scope, initializes some basic graphic features
* getGeoInformation function - initializes search scope,initializes initial starting point of search, initializes radius of search, adds some event listeners that necessary for the map
* initializeFilters function - initializes all filters that will be used for filtering map data
* reset function - this function will need for implementation of new search
* getFusionInfo function - connects to Fusion Tables Application and pulls all necessary information for the map creation

**2. Page Editor**  - html file that will specify location and properties of elements that will be shown on the main html page.

* header section - name of application and logo
* general information section - all necessary instructions
* filter section - this section contains multiple filters that can be used in different combinations at the same time
* map section - filtered data represented on the map as points that can be clicked to access a modal with additional information
* map control section - allows change size of the map, move map, switch between map and satellite view, use 3d simulation, enter and exit from full screen mode
* buttons section - user can choose to clear all filters or apply all chosen filters

**3. Style Editor** - css file that helps to define, apply, and manage sets of style characteristics.

1. filter style - initializes additional style elements for all filters
2. page style - initializes additional style elements for the main web page
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.

**Property Data Generator**  - development status: existing software to be re engineered

Name: Property Data Generator

Wisner, Steve. “Property Data Generator.” SW, Steve Wisner, 2018, stevewisner.com/#zillow.

**Fusion Table Application -** development status: existing software to be reused

Name: Fusion Tables

“About Fusion Tables - Fusion Tables Help.” Google, Google, 2018, support.google.com/fusiontables/answer/2571232?hl=en.

**Website Builder** - development status: existing software to be re engineered

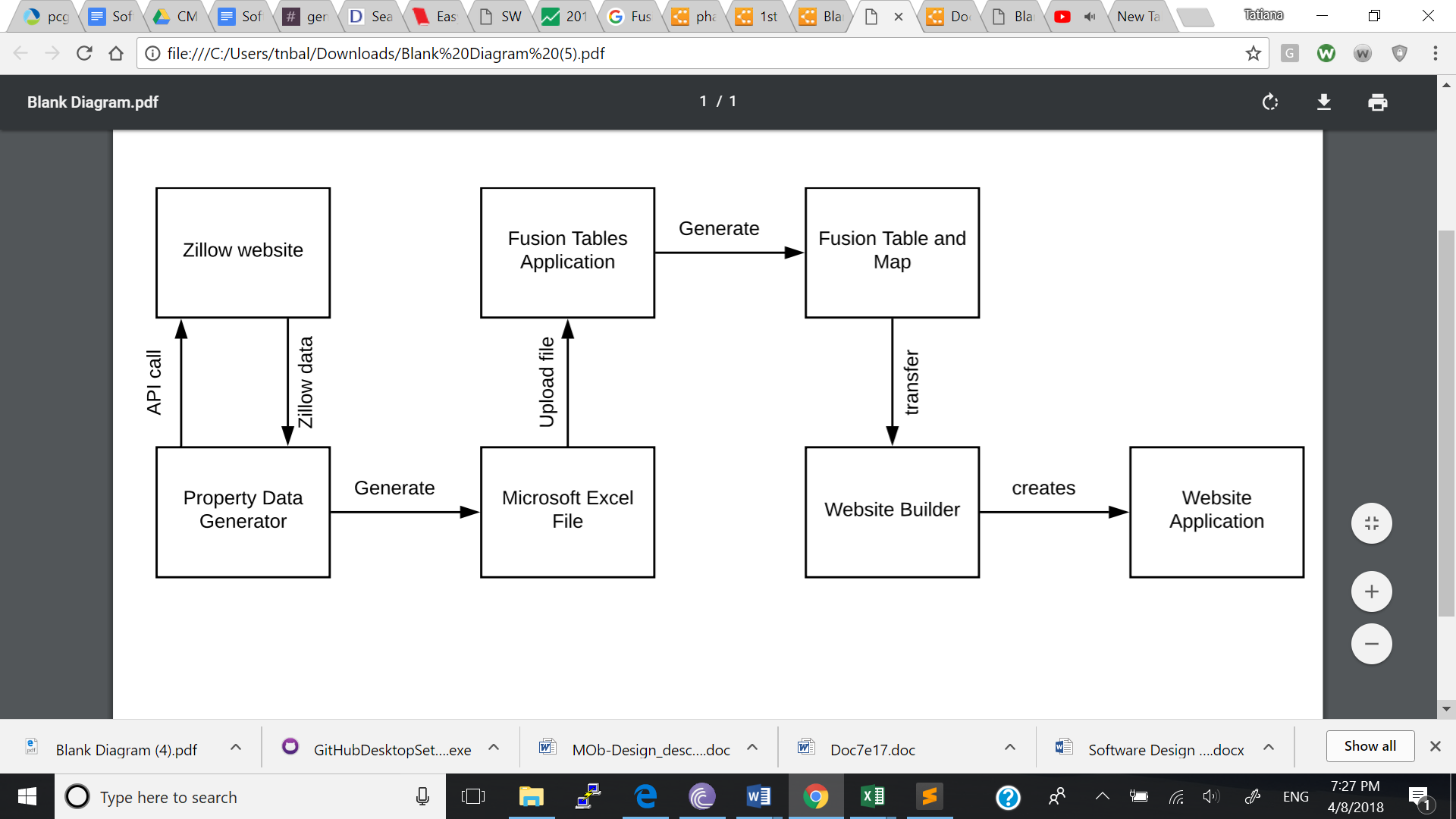
Name: Searchable Map Template with Google Fusion Tables

Eder, Derek. “Searchable Map Template.” Derek Eder, Derek Eder, 2017, derekeder.com/searchable\_map\_template/.

1. 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)
2. Identify the program library in which the software that implements each software unit is to be placed.

## 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.



## 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

This paragraph shall state the project-unique identifier assigned to each interface and shall identify the interfacing entities (software units, systems, configuration items, users, etc.) by name, number, version, and documentation references, as applicable. The identification shall state which entities have fixed interface characteristics (and therefore impose interface requirements on interfacing entities) and which are being developed or modified (thus having interface requirements imposed on them). One or more interface diagrams shall be provided, as appropriate, to depict the interfaces.

### (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

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. 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.)

The (WEB) CSCI will have an API that will be used to communicate with the client-side system. The API will have the following calls available from the client-side system. The data types are listed in the parameters and the results and are required to be consistent. The (WEB) CSCI will be making HTTP requests to this API, and when it does the following shall be the available responses:

GET /counties/

* params: { “state”: string }
* Result: { “err”: string “data”: { “counties”: [ {“name”: string, “latitude”: float, “longitude”: float, “id”: string }]
* Description: Will return a list of counties within the given state.

GET /counties/cities

* params: {“state”: string, “County”: string}
* Result:{ “err”: string, “data”: [ {“name”: string, “latitude” float, “longitude”: float, “id”: string} ] }
* Description: Will return an array of cities within a state and county

GET /counties/cities/neighborhoods

* params: { “state”: string, “County”: string, “city”: string }
* Results: { “err”: string, “data”: [ {“name”: string, “latitude”: float, “longitude”: float, “id”: string } ] }
* Description: Will return an array of neighborhoods within a city.

GET /counties/cities/neighborhoods/amenities

* params:
* { “longitude”: float, “latitude”: float, “radius”: float, “types”: [string], “id”: string }
* Results: { “err”: string, “data”: [ {“name”: string, “type”: string, “longitude”: float, “latitude”: float} ] }
* Description: Will return a list of amenities with the given types

The (Data) CSCI will be required to obtain the information from the external Zillow and Yelp API’s. This includes a list of amenities which shall contain

{ “name”: string, “type”: string, “longitude”: float, “latitude”: float}

a list of counties which shall contain

{ “name”: string, “longitude”: float, “latitude”: float }

and a list of cities which shall contain

{ “name”: string, “longitude”: float, “latitude”: float }.

# 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

# Requirements traceability

This section shall contain:

1. Traceability from each software unit identified in this SDD to the CSCI requirements allocated to it. (Alternatively, this traceability may be provided in [4.1.)](#_35nkun2)
2. Traceability from each CSCI requirement to the software units to which it is allocated.

# 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.).