

Software Requirements Specification

for

C-Teaching-Website

Version 1.0

Prepared by

Group Name: codeTeachers

|  |  |  |
| --- | --- | --- |
| Christian Galvez | 11590801 | christian.galvez@wsu.edu |
| Ethan Pongon | 11639010 | ethan.pongon@wsu.edu |
| <name> | <student #> | <e-mail> |
| <name> | <student #> | <e-mail> |
| <name> | <student #> | <e-mail> |

|  |  |
| --- | --- |
|  |  |
| Date: | <place the date of submission here> |
|  |  |
|  |  |
|  |  |

Content

Revisions iii

1 Introduction 1

1.1 Document Purpose 1

1.2 Product Scope 1

1.3 Intended Audience and Document Overview 1

1.4 Definitions, Acronyms and Abbreviations 1

1.5 Document Conventions 1

1.6 References and Acknowledgments 2

2 Overall Description 3

2.1 Product Perspective 3

2.2 Product Functionality 3

2.3 Users and Characteristics 3

2.4 Operating Environment 3

2.5 Design and Implementation Constraints 4

2.6 User Documentation 4

2.7 Assumptions and Dependencies 4

3 Specific Requirements 5

3.1 External Interface Requirements 5

3.2 Functional Requirements 6

3.3 Behavior Requirements 6

4 Other Non-functional Requirements 7

4.1 Performance Requirements 7

4.2 Safety and Security Requirements 7

4.3 Software Quality Attributes 7

5 Other Requirements 8

Appendix A – Data Dictionary 9

Appendix B - Group Log 10

Revisions

| Version | Primary Author(s) | Description of Version | Date Completed |
| --- | --- | --- | --- |
| Draft Type and Number | Full Name | Information about the revision. This table does not need to be filled in whenever a document is touched, only when the version is being upgraded. | 00/00/00 |

# *<In this template you will find text bounded by the “<>” symbols. This text appears in italics and is intended to provide explanations and guide you through the document. There are two types of comments in this document. The comments that are in black are intended specifically for the course. The comments that are in blue are more general and apply to any SRS. Please make sure to delete all of the comments before submitting the documen**t.**>*

# Introduction

Our project is a website that is designed to allow users to learn about C programming and practice writing C code. In this section you will find details about the purpose of this SRS document, a brief description of the product being created and what it hopes to accomplish, and a description of who this document is intended for. In addition to that, there is also information about the conventions and jargon that this document will employ as well as a list of references for items referenced by this document.

*<TO DO: Please provide a brief introduction to your project and a brief overview of what the reader will find in this section.>*

## Document Purpose

This SRS document will cover the C-Teaching-Website v1.0 system as a whole. This document will contain all of the system requirements pertaining to the C-Teaching-Website, as well as detailing user requirements and external interface systems. The C-Teaching-Website v1.0 will be created by the codeTeachers group, who will also be responsible for this document.

<Identify the product whose software requirements are specified in this document, including the revision or release number. Describe the scope of the product that is covered by this SRS, particularly if this SRS describes only part of the system or a single subsystem.

TO DO: Write 1-2 paragraphs describing the purpose of this document as explained above.>

## Product Scope

The C-Teaching-Website final product will be a website that will provide users with an environment to learn about the C programming language and practice implementing its concepts. This usability will be delivered through tutorials and coding challenges that users will work through in a sequential manner. Users will create an account and then start the coding challenges which will involve learning about a C programming concept and then implementing that concept with correct behavior and syntax. To implement the concepts studied in the lessons, users will be able to write and run code directly on the website. Once a user has correctly implemented the concept from the lesson they are working on they will be allowed to continue to the next lesson. The goal of the product is to reduce the roadblocks someone may encounter while trying to learn C programming, for example having to learn how to compile and run code from the terminal. Another benefit of the product is that users will have access to information about concepts they’re learning on the same page they’re writing code on, making it easier to review their code as they write it.

<Provide a short description of the software being specified and its purpose, including relevant benefits, objectives, and goals.

TO DO: 1-2 paragraphs describing the scope of the product. Make sure to describe the benefits associated with the product.>

## Intended Audience and Document Overview

This document is intended for the professor and any user interested in the C-Teaching-Website project. This document contains the C-Teaching-Website project’s goals, functionality, users and characteristics, operating environment, design, user documentation, and dependencies. This document also contains external interface, functional and behavior requirements. Non-functional requirements are described as well, such as performance requirements, safety requirements and software quality. For the professor, starting with the introduction and working down the document is the suggested method for reading this document. For the user, reading the introduction in section 1 and reading section 2 may be sufficient.

<Describe the different types of reader that the document is intended for, such as developers, project managers, marketing staff, users, testers, and documentation writers (In your case it would probably be the “client” and the professor). Describe what the rest of this SRS contains and how it is organized. Suggest a sequence for reading the document, beginning with the overview sections and proceeding through the sections that are most pertinent to each reader type.>

## Definitions, Acronyms and Abbreviations

**a.out**: Generic name for executables created from the programming lessons.

**C**: A lower level human readable compiled computer programming language.

**C-Teaching-Website v1.0**: C-Teaching-Website is the name of the project this SRS document describes, and the v1.0 describes the version number of the website this document describes.

**encryptor**: A tool created in C used to encrypt local files on the host machine (server).

**Executable**: Machine code generated from feeding a script into a compiler.

**Express**: A Node.js framework used for creating different web applications.

**GCC**: A compiler for the C programming language.

**JavaScript**: Programming language interpreted by web browsers. Used for the backend of the C-Teaching-Website.

**lesson#\_tests.c**: lesson#\_tests.c files are test files written in C to be used for testing user entered code.

**Node.js**: An asynchronous JavaScript runtime used to run the C-Teaching-Website.

**//#B**: A symbol used to split a lesson#\_tests.c file into two strings.

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.

TO DO: Please provide a list of all abbreviations and acronyms used in this document sorted in alphabetical order.>

## Document Conventions

This document uses IEEE formatting for citing external sources used. Document uses the Arial font in size 11 and is single spaced. This SRS document uses 1” margins. Text in this document is justified. Bold words inside of a paragraph are used when listing a sequence of elements, such as the definitions described in section 1.4. When this SRS document references specific files or programs used by the C-Teaching-Website, all proper names will be capitalized as defined by that program.

<In general this document follows the IEEE formatting requirements. Use Arial font size 11, or 12 throughout the document for text. Use italics for comments. Document text should be single spaced and maintain the 1” margins found in this template. For Section and Subsection titles please follow the template.

TO DO: Describe any standards or typographical conventions that were followed when writing this SRS, such as fonts or highlighting that have special significance. Sometimes, it is useful to divide this section to several sections, e.g., Formatting Conventions, Naming Conventions, etc.>

## References and Acknowledgments

<List any other documents or Web addresses to which this SRS refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document.

TO DO: Use the standard IEEE citation guide (attached) for this section.> IEEE

# Overall Description

## Product Perspective

<Describe the context and origin of the product being specified in this SRS. For example, state whether this product is a follow-on member of a product family, a replacement for certain existing systems, or a new, self-contained product. If the SRS defines a component of a larger system, relate the requirements of the larger system to the functionality of this software and identify interfaces between the two. In this part, make sure to include a simple diagram that shows the major components of the overall system, subsystem interconnections, and external interface. In this section it is crucial that you will be creative and provide as much information as possible.

TO DO: Provide at least one paragraph describing product perspective. Provide a general diagram that will illustrate how your product interacts with the environment and in what context it is being used, i.e., context diagram.>

## Product Functionality

<Summarize the major functions the product must perform or must let the user perform. Details will be provided in Section 3, so only a high level summary is needed here. Organize the functions to make them understandable to any reader of the SRS. A picture of the major groups of related requirements and how they relate, such as a top level data flow diagram or object class diagram, will be effective.

TO DO:

1. Provide a bulleted list of all the major functions of the system

2. **(Optional)** Provide a Data Flow Diagram of the system to show how these functions relate to each other. This is useful when there is a clear sequence for the functions being performed.>

## Users and Characteristics

<Identify the various users that you anticipate will use this product. Users may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience.

TO DO:

1. Describe the pertinent characteristics of each user. Certain requirements may pertain only to certain users.

3. Distinguish the most important users for this product from those who are less important to satisfy.>

## Operating Environment

<Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist. In this part, make sure to include a simple diagram that shows the major components of the overall system, subsystem interconnections, and external interface

TO DO: As stated above, in at least one paragraph, describe the environment your system will have to operate in. Make sure to include the minimum platform requirements for your system. >

## Design and Implementation Constraints

<Describe any items or issues that will limit the options available to the developers. These might include: hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer’s organization will be responsible for maintaining the delivered software).

TO DO: In this section you need to consider all of the information you gathered so far, analyze it and correctly identify relevant constraints.>

## User Documentation

The landing page for the website contains brief instructions on how to use the C-Teaching-Website. The instructions describe what user input should look like for each lesson, and what the results page means after submitting a test. The instructions given on the landing page should be sufficient for the user to understand how to use the website fully.

<List the user documentation components (such as user manuals, on-line help, and tutorials) that will be delivered along with the software. Identify any known user documentation delivery formats or standards.

TO DO: You will not actually develop any user-manuals, but you need to describe what kind of manuals and what kind of help is needed for the software you will be developing. One paragraph should be sufficient for this section.>

## Assumptions and Dependencies

<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project.

TO DO: Provide a short list of some major assumptions that might significantly affect your design. For example, you can assume that your client will have 1, 2 or at most 50 Automated Banking Machines. Every number has a significant effect on the design of your system. >

# Specific Requirements

## External Interface Requirements

### User Interfaces

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., Cancel) that will appear on every screen, error message display standards, and so on. Define the software components for which a user interface is needed.

TO DO: The least you can do for this section is to describe in words the different User Interfaces and the different screens that will be available to the user. Optional: You may also provide an initial Graphical User Interface design (does not have to be final).>

### Hardware Interfaces

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware. You are not required to specify what protocols you will be using to communicate with the hardware, but it will be usually included in this part as well.

TO DO: Please provide a short description of the different hardware interfaces. If you will be using some special libraries to communicate with your software mention them here. In case you have more than one hardware interface divide this section into subsections.>

### Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems (Windows? Linux? Etc…), tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.

TO DO: The previous part illustrates some of the information you would usually include in this part of the SRS document. To make things simpler, you are only required to describe the specific interface with the operating system.>

### Communications Interfaces

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.

TO DO: Do not go into too much detail, but provide 1-2 paragraphs were you will outline the major communication standards. For example, if you decide to use encryption there is no need to specify the exact encryption standards, but rather, specify the fact that the data will be encrypted and name what standards you consider using. >

## Functional Requirements

*< Functional requirements capture the intended behavior of the system. This behavior may be expressed as services, tasks or functions the system is required to perform. This section is the direct continuation of section 2.2 where you have specified the general functional requirements. Here, you should list in detail the different product functions with specific explanations regarding every function.*

*TO DO: Break the functional requirements to several functional areas and divide this section into subsections accordingly. Provide a detailed list of all product operations related to these functional areas.>*

## Behavior Requirements

### Use Case View

<A use case defines a goal-oriented set of interactions between external actors and the system under consideration.

TO DO: Provide a use case diagram which shows the entire system and all possible actors. Do not include detailed use case descriptions (these will be needed when you will be working on the Test Plan), but make sure to include a short description of what every use-case is, who are the actors in your diagram.>

# Other Non-functional Requirements

## Performance Requirements

<If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.

TODO: Provide relevant performance requirements based on the information you collected from the client. For example you can say “1. Any transaction will not take more than 10 seconds, etc…>

## Safety and Security Requirements

<Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. Refer to any external policies or regulations that state safety issues that affect the product’s design or use. Define any safety certifications that must be satisfied. Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements.

TODO:

* Provide relevant safety requirements based on your interview with the client or, on your expectation for the product.
* Describe briefly what level of security is expected from this product by your client and provide a bulleted (or numbered) list of the major security requirements.>

## Software Quality Attributes

<Specify any additional quality characteristics for the product that will be important to either the customers or the developers. Some to consider are: adaptability, availability, correctness, flexibility, interoperability, maintainability, portability, reliability, reusability, robustness, testability, and usability. Write these to be specific, quantitative, and verifiable when possible. At the least, clarify the relative preferences for various attributes, such as ease of use over ease of learning.

TODO: Use subsections (e.g., 4.3.1 Reliability, 4.3.2 Portability, etc…) provide requirements related to the different software quality attributes. Base the information you include in these subsections on the material you have learned in the class. Make sure, that you do not just write “This software shall be maintainable…” Indicate how you plan to achieve it, etc.>

# Other Requirements

<This section is **Optional.** Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

Appendix A – Data Dictionary

*<Data dictionary is used to track all the different variables, states and functional requirements that you described in your document. Make sure to include the complete list of all constants, state variables (and their possible states), inputs and outputs in a table. In the table, include the description of these items as well as all related operations and requirements.>*

Appendix B - Group Log

<Please include here all the minutes from your group meetings, your group activities, and any other relevant information that will assist the Teaching Assistant to determine the effort put forth to produce this document>