CodePyramid

Group 7
David Snavely, Michael Mason, Parker Siemek, Thomas Boswell, Isaak Arslan

Software Requirements Specification

Document

Version: 1.0 **Date: (2/9/2018)**

Table of Contents

1. Introduction

1.1 Purpose

The purpose of this document is to give a thorough and detailed requirements description for the CodePyramid website. It will explore the intent and functions of the application. This document is intended for stakeholders of the system.

1.2 Scope

The website, CodePyramid, will be an online instructional platform aimed towards students with no prior programming experience. The website will serve students as a repository of course material, and as an interactive service where students attempt to write their own code.

1.3 Definitions, Acronyms, and Abbreviations.

HTML (or HTML5) - Refers to Hypertext Markup Language Version 5. This language is used to structure websites.
CSS - Refers to Cascading Style Sheets, used to add design and visual attributes to web pages.
JavaScript - A scripting language used to add interactivity to websites.
Parser - component of a compiler or interpreter, which parses the source code of a computer programming language to create some form of internal representation
Web Based Application - A web-based application is any program that is accessed over a network connection using HTTP, rather than existing within a device's memory.
MySQL - The database management system we are using.
C# - Coding language that will be used to manipulate database elements
FAQ - Frequently Asked Questions, a section where people can check answers to common questions to try and resolve their issues without requiring a developer to respond to them.
SRS - System Requirements Specification (this document)
Hash - A one-way function used on passwords so the original password cannot be retrieved

1.4 References

[1] IEEE Software Engineering Standards Committee, "IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications", October 20, 1998.

1.5 Overview

The remainder of the SRS describes the system's interfaces, functions, assumptions, and requirements.

2. The Overall Description

2.1 Product Perspective

2.1.1 Software Perspective

2.1.1.1 The product is a website that teaches students to code through written instructionals and interactive coding exercises.

2.1.1.2 This product is not the first of its kind, but is not identical to similar predecessors. Our system is similar to teaching sites like Codecademy in their basic premises: teaching students programming. However, CodePyramid will differ in the style of its written instruction and the design of its programming exercises.

2.1.1.3 As a website, the product is self contained.

2.1.2 System Interfaces

2.1.2.1 The system will interface with a MySQL database using C# scripts. Student account information will be stored in the database.

2.1.3 Interfaces

2.1.3.1 The system is a website, so user will interact and receive instruction via a web browser.

2.1.3.2 In regards to the Americans with Disabilities Act, we will format our website to be easily readable for all students. At the user's option, all text can be displayed in Comic Sans (a font especially easy to read for dyslexic students and others).

2.1.3.3 Sign Up and Login Interfaces

2.1.3.3.1 The website will allow new students to create an account, which keeps track of their progress.

2.1.3.3.2 If a returning student visits the website, they will be allowed to login and retain progress.

2.1.4 Hardware Interfaces

2.1.4.1 Students require an internet connection and a computer able to operate a web browser in order to receive instruction.

2.1.5 Software Interfaces

2.1.5.1 Since all instruction is delivered via a website, the only required software interface is a web browser.

2.1.6 Communications Interfaces

2.1.6.1 No communications interfaces except for the standard HTTP protocol are used.

2.1.7 Memory Constraints

2.1.7.1 As we instruct our students only through a website, very little memory is required. The exact memory consumption depends on the browser used by the student, but is generally very small.

2.1.8 Operations

2.1.8.1 Our website has two modes of operation: a student logging in, and a student learning through

the website's instructional pages.

2.1.9 Site Adaptation Requirements

2.1.9.1 Requirements for initialization are username and password to login to the website. After logging in or registering an account, it will change operational modes for the user.

2.1.9.2 While writing the CSS code for this website, specifications for alternate web browsers may be required since different browsers can read CSS code differently.

2.2 Product Functions

2.2.1 Coding Instruction

2.2.1.1 Lessons

2.2.1.1.1 Basic programming concepts will be taught through provided text and lessons.

2.2.1.1.2 Small exercises will be used to track progress, and will be placed between sections of text in the lessons.

2.2.1.1.3 At the end of each lesson an assessment based on the exercises will be given to the student, and information on the users' scores will be stored on their account.

2.2.2 Account Tracking and Management

2.2.2.1 Students will be able to track their progress and see their assessment grades on their account.

2.3 User Characteristics

2.3.1 Students

2.3.1.1 The user base is anticipated to be mostly students. As such, most will have no coding experience. To compensate, the website's text curriculum will teach users information starting with the very basics.

2.3.2 Children

2.3.2.1 Many users will be children or young adults who wish to learn programming. The website's layout will be well formatted to accommodate all links, but simplistic enough to keep students focused.

2.4 Constraints

2.4.1 Security Considerations

2.4.1.1 CodePyramid, for the sake of our safety and the safety of user information stored in cookies, will execute user scripts on a separate domain than that with which the user is interacting.

2.5 Assumptions and Dependencies

2.5.1 Browser Updates

2.5.1.1 The only assumption in our website is that the student uses a sufficiently up-to-date version of Chrome or Firefox. If they do not, their browser may not be able to interact with the current renditions of Javascript and CSS used in the website.

2.6 Apportioning of Requirements.

2.6.1 Code Execution

2.6.1.1 Performing static code analysis is one of the most rigorous and complex components of the curriculum design. As such, it may be delayed until future versions of the website. In lieu of interactive code exercises, students can read relevant sections of the text.

3. Specific Requirements

3.1 External Interfaces

3.1.1 Primary/Surface Interface

3.1.1.1 CodePyramid will provide a standard interface to create and login to accounts, progress through lessons, and look at past records of their progress through lessons.

3.1.2 Secondary/Code Interface

3.1.2.1 Within lesson pages, there will be an interface where users can write code into for the purpose of practicing concepts taught to them.

3.1.2.2 The code interface will have a JavaScript and HTML input field on the left half of the screen.

3.1.2.3 The code interface will have a run button so that the JavaScript and HTML code users have written will execute and display on the right half of the screen.

3.1.2.4 An information console for the user to check if their solution was correct will be provided in a box on the right half of the screen.

3.2 Functions

3.2.1 User Registration

3.2.1.1 On the top of the website's homepage, there will be a link to make a new account. Upon clicking this link, the user will be prompted to enter a username, password, and email address. If the username is taken, the user will be notified to insert another username, and this process will repeat until a name that is unused is provided. If the email address is improper or used, they will be notified of either error and prompted to re enter their email. There are no restrictions on password size or format. The user's information will be loaded into the site's user database. After the user finishes the registration process, they will be returned to the site's homepage.

3.2.2 User Login

3.2.2.1 On the top of the website's homepage, next to the registration link, there will be a link to log in. Upon clicking the link, the user will be prompted to enter their username and password. If the username does not belong to any user, or if the username and password are incorrect, the user will be notified and prompted to re enter their information. Upon the 100th incorrect query, the

site will close the tab in which the user is trying to login. After a successful login, the user will be returned to the site's homepage

3.2.3 User Navigation

3.2.3.1 Whenever the user clicks a link on the site, they will be taken to the address specified by the link.

3.2.4 Code Submission by User

3.2.4.1 In the lesson pages, the site will prompt users to enter snippets of code for correctness checking. This code will be passed into a parser, which will verify the code's correctness. If the code is correct, the user will be prompted to hit the 'continue' link to save their progress and continue with the course. If the code that the user provides is incorrect, the parser will check for common coding errors, and the site will display relevant hints in red text. If the code was correct, the site will update the flag for that section of the course.

3.2.4.2 In order to submit code, the user will be required to be logged in.

3.2.5 Support Message Submission by User

3.2.5.1 The site will have a support page containing a text box for users to submit questions. These questions will be emailed to the development team.

3.2.5.2 In order to submit questions to developers, the user is required to be logged in.

3.2.5.3 The support section will contain a FAQ to help users without the need for contacting the development team.

3.3 Performance Requirements

3.3.1 Software Requirements

3.3.1.1 The website software must periodically interact with the SQL database via queries and update statements. The quantity of data is small, and thus the strain of the operation is minimal. Any SQL database interaction should complete in 1 second or less.

3.3.1.2 The parsing code used in student code submission should complete in 20 seconds or less.

3.4 Logical Database Requirements

3.4.1 Database Information

3.4.1.1 The SQL database stores only user account information. This includes a student's username, email address, password, and site progress. The student's password will be hashed before it's stored to prevent any security abuses.

3.5 Design Constraints

3.5.1 Standards Compliance

3.5.1.1 We will conform to all top level domain service requirements and naming conventions.

3.6 Software System Attributes

3.6.1 Reliability

3.6.1.1 The website will have MTBF (mean time between failures) of 1000 page loads. That is, the website is expected to successfully load 1000 pages with no more than 1 failure.

3.6.2 Availability

3.6.2.1 The website will have 99% availability at any given time.

3.6.3 Security

3.6.3.1 SQL Injection

3.6.3.1.1 All login information provided by the user will be sanitized before executed in the database.

3.6.3.2 Executable code Injection

3.6.3.2.1 Coding exercises will be sanitized before they're run in order to prevent malicious code from being executed.

3.6.3.3 Reset Password

3.6.3.3.1 If a user forgets their password, they can request a link sent to their email that will allow them to reset their password.

3.6.3.4 Data Breach

3.6.3.4.1 In the event of a data breach, all users are sent an email informing them of the incident, and the next time they log on it will be recommended that they change their password.

3.6.3.5 Support Page Code Injection

3.6.3.5.1 Information submitted as an email through the support page will be sanitized to ensure there is no code injection.

3.6.3.6. Hashed Passwords

3.6.3.6.1 Passwords will be hashed to ensure they are secure.

3.6.4 Maintainability

3.6.4.1 The website will keep a log of any and all errors.

3.6.5 Portability

3.6.5.1 The Chrome and Firefox web browsers will be supported.

3.6.6 Usability

3.6.6.1 All drop-down menus, links, tabs, and text will be organized to ensure smooth navigation of the site and user satisfaction.

4. Change Management Process

4.1 Requirements For SRS Changes

4.1.1 Changes to the SRS are required to have approval from the majority of group members before they can be updated and/or put into effect.

4.2 Submission Requirements For SRS Changes

4.2.1 A support page will be included with a link to the email of the development team. With this email the user will be able to ask questions, report bugs, or suggest new features for CodePyramid.

4.3 Logging SRS Changes

4.3.1 A secondary link will be included on the support page, which will redirect the user to a page containing all current updates to the SRS along with their corresponding version number.

5. Document Approvals

David Snavely

X_____ Date: _____

Thomas Boswell

X_____ Date: _____

Isaak Arslan

X_____ Date: _____

Parker Siemek

X_____ Date: _____

Michael Mason

X_____ Date: _____

6. Supporting Information

6.1 No supporting information is required at this time.