Java Learning Application

Software Requirements Specification

Version 1.0

2/14/2024

Max Flannick - Team Leader

Destiny Deleon - Programmer

Anthony Epault - Designer

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Description** | **Author** | **Comments** |
| <date> | <Version 1> | <Your Name> | <First Revision> |
| 2/14/2024 | Java Learning Application 1.0 | Max Flannick | initial draft |
| 2/14/2024 | Java Learning Application 1.0 | Destiny Deleon | initial draft |
| 2/14/2024 | Java Learning Application 1.0 | Anthony Epault | initial draft |

**Document Approval**

The following Software Requirements Specification has been accepted and approved by the following:

|  |  |  |  |
| --- | --- | --- | --- |
| **Signature** | **Printed Name** | **Title** | **Date** |
| Max Flannick | Max Flannick | Team Leader | 2/14/2024 |
| Destiny Deleon | Destiny Deleon | Programmer | 2/14/2024 |
| Anthony Epault | Anthony Epault | Designer | 2/14/2024 |

**Table of Contents**

1. Introduction 1

1.1 Purpose 1

1.2 Scope 1

1.3 Definitions, Acronyms, and Abbreviations 1

1.4 Overview 1

2. General Description 1

2.1 Product Perspective 1

2.2 Product Functions 2

2.3 Users and Characteristics 2

2.4 General Constraints 2

2.5 Assumptions and Dependencies 2

2.6 Operating Environment 2

3. Specific Requirements 2

3.1 External Interface Requirements 3

3.1.1 User Interfaces 3

3.1.2 Hardware Interfaces 3

3.1.3 Software Interfaces 3

3.1.4 Communications Interfaces 3

3.2 Functional Requirements 3

3.2.1 <Functional Requirement or Feature #1> 3

3.2.2 <Functional Requirement or Feature #2> 3

3.3 Use Cases 4

3.3.1 Use Case #1 4

3.3.2 Use Case #2 4

3.4 Non-Functional Requirements 4

3.5.1 Performance 5

3.5.2 Reliability 5

3.5.3 Availability 5

3.5.4 Security 5

3.5.5 Maintainability 5

3.5.6 Portability 5

3.5 Design Constraints 5

3.6 Logical Database Requirements 5

3.7 Other Requirements 5

4. Analysis Models 5

4.1 Sequence Diagrams 6

5. Change Management Process 6

A. Appendices 6

A.1 Appendix 1 7

A.2 Appendix 2 7

**List of Figures**

[Figure 1: Data Flow Diagram Example 1 6](#_Toc504395184)

[Figure 2 Data Flow Diagram Example 2 6](#_Toc504395185)

**List of Tables**

[Table 1: Use case 1 4](#_Toc504395186)

# 1. Introduction

## 1.1 Purpose

Our goal is to develop a desktop application tailored to assist beginners or students in learning Java coding, with the aim of fostering a deeper understanding of programming languages. We aspire to outpace competitors by achieving a 10 percent increase in efficiency, ensuring that users grasp concepts at a quicker pace. Moreover, we intend to enhance user engagement by making chapters less condensed, approximately reducing the content by 5-10 percent, while still providing comprehensive coverage of each topic and its core components.

## 1.2 Scope

I. Software Product(s)

A. Name: Java Learning Application

B. Description:

1. The software product will:

a. Provide a desktop application guiding users through learning Java Programming.

b. Utilize Java for coding with JavaFX or a similar tool for the user interface.

c. Comprise chapters covering various topics from basic to more complex Java concepts.

2. The software product will not:

a. Include animations for every chapter or section to avoid distractions.

b. Cover more complex topics like recursive processes due to time constraints.

c. Offer multiplayer minigames as the focus is on single-user interaction.

d. Incorporate leaderboards for minigames due to memory and time constraints.

II. Application Scope

A. Deliverables:

1. Phase 1:

a. Implementation of interactive title screen.

b. Organization of teaching materials for each section.

c. Preparation of teaching materials in a presentable format.

2. Phase 2:

a. Creation of quizzes with approximately 10-15 questions per chapter.

b. Development of interactive learning elements for each chapter.

c. Ensuring error-free functionality and readiness for integration.

3. Phase 3:

a. Design and implementation of matching and other minigames.

b. Inclusion of sound effects.

c. Final polishing and integration of all components.

B. Application Objectives:

1. Provide a structured learning experience for Java Programming.

2. Engage users through interactive elements such as quizzes and minigames.

3. Present material in an organized and user-friendly manner.

C. Consistency with Higher-Level Specifications:

1. The application aligns with the outlined chapters and deliverables.

2. Focuses on achieving the specified learning objectives and goals.

III. Application Description

A. Benefits, Objectives, and Goals:

1. Offer parameter-driven, user-definable reports with a 2-hour turnaround.

2. Provide interactive learning experiences through quizzes and minigames.

3. Organize material into structured chapters for effective learning.

B. Application Context:

1. Desktop application for individuals seeking to learn Java Programming.

2. Designed to cater to users ranging from beginners to intermediate learners.

3. Emphasizes practical understanding through interactive elements and quizzes.

## 1.3 Definitions, Acronyms, and Abbreviations

**Java** - A high-level, class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible.

**Web Application** - Software that runs in your web browser

**Function** - a block of code used to do a certain task, such as adding two values for example, which could help reduce the amount of lines of code within a program file. It is helpful for dealing with repetitive tasks.

**Arguments** - values passed to a function

**Parameters** - values defined and used within a function

**Recursive Processes** - a process used in programming where the a function calls itself within the function

**Class** - a common feature within Java used in program files. A class gives the ability to create objects to call functions within the class.

**Parent/Child Class** - a type of relationship between two (or more) classes where the child class has its own functions and is able to access functions and data within the parent class as well. The parent class, however, can access functions and data from the child(ren) classes.

## 1.4 Overview

In this project we aim to teach those who do not understand the fundamentals of Java through a fun matching desktop game. Through this game, we will teach the basics of the coding language in the simplest way possible without losing the focus of the one being taught.

# 2. General Description

## 2.1 Product Perspective

This is a new, self-contained educational tool focused on helping users learn and understand the programming language Java. Our application is designed to provide a comprehensive learning experience specifically tailored towards students/beginners. These educational modules will consist of basics of programming concepts (printing, if statements, etc.) along with more complex concepts (functions/arguments, classes, and constructors, etc.). These concepts will be taught through textbook-like readings, quizzes, minigames, etc. We plan to use a database to store the answers to the quizzes and minigames.

## 2.2 Product Functions

* **Textbook Selection** - Users can view chapters within the textbook to learn different concepts, such as loops.
* **Practice Questions** - Users can access practice questions related to concepts learned, such as if statements. These questions may be in the form of multiple-choice questions or open-ended questions.
* **Matching Game** - Users can match the concepts they’ve learned with their definition from each chapter.
* **Visual Study Guide with Images** - Users can choose to run a code snippet within to get a better understanding of how outputs are generated. These visuals will only be placed in the loops chapter.
* **Flashcards for Chapters** - Users can flip through virtual flash cards with the definition of the concepts they’ve learned on one side and the concept name on the other. These flashcards will only be included in more complex chapters, such as those that go over functions and loops.
* **Coding Minigame** - Users can try to solve 5 beginner level coding problems upon completing each chapter.
* **Textbook Section Divided into Topic Sections** - Users can view subsections within each chapter that will have a more in-depth focus on specific topics. For example, the loops chapter will have subsections for while loop, for loops, etc.
* **Timer Introduced into Minigames** - Users can keep track of how fast they complete a minigame within the application. This function can be used for the user to keep track of their high score.
* **Chapter Summary Section for Chapters -** Users can view key topics discussed within the chapter at the end of each chapter.
* **Table of Contents with Hyperlinks** - Users can interact with hyperlinks to easily navigate through the application.
* **End-of-Chapter Quiz for Chapters** - Users can test their knowledge of each concept through quizzes at the end of each chapter. These questions may be in the form of multiple-choice questions or open-ended questions.

## 2.3 Users and Characteristics

There are three types of user classes that we anticipate will use our application: beginners to programming, students learning Java within a school setting, and people out in the workforce wanting a refresher on Java. Each class is predicted to wish to interact with the application in a different way.

* **Beginners to programming** - This class of users won’t have any experience with coding languages, so it’s important to ensure that the concepts taught in this application are clear for beginners to understand. Beginners may require step-by-step explanations and interactive learning tools to fully grasp the fundamentals of Java effectively.
* **Students** - This class of users are students who are learning Java within a class setting. This class differs from beginners to programming since students may have had some coding experience or exposure to other programming languages. They may benefit from the quiz functions so their instructor could get a grasp on their understanding of Java while using the application.
* **Java refreshers** - This class of users have an understanding of Java but want to either learn more about the programming language, get a refresher of the Java programming language or learn about any updates to java as newer versions come out. They may seek concise explanations and practical examples to quickly refresh their knowledge or stay updated with the latest features and best practices in Java programming. They may benefit from the table of contents with hyperlinks so they can easily navigate to anything specific they want a refresher on.

## 2.4 General Constraints

Time constraints may play a role during the duration of the project. Some sections may be longer than others so it will be important to properly organize what is required for each section to complete everything on time.

Error processing is important during the testing of the program. Some errors may be longer to debug than others so we will test each chapter multiple times to make sure everything functions properly before moving on.

## 2.5 Assumptions and Dependencies

We will assume the user’s computer will be able to handle the execution of the application. This application will not demand too much processing power so the user will most likely be able to run the application on their computer with no problems. If there are any issues with the hardware or the operating system of the user's computer, the application may not run as it intended to.

## 2.6 Operating Environment

This application will be able to operate on standard laptops and desktops as long as enough memory is available upon the installation of the application. The application should run well on operating systems such as Windows 10 and macOS Catalina along with more recent operating systems. This application may also work well on earlier operating systems, however, to ensure there are no disruptions during the use of the application, we suggest that the user have the latest version of their operating system.

# 3. Specific Requirements

## 3.1 External Interface Requirements

### 3.1.1 User Interfaces

* UI1: For each chapter within the application there will be an informational section like a textbook or slides. However, this will only cover the main points without a large word count. Each chapter will specify sections within it to help organize the information.
* UI2: The chapter will also include practice questions, so the user is able to strengthen their learning.
* UI3: Each chapter will also a set of 10-15 questions at the end of the chapter as review.
* UI4: Some chapters will include minigames such as a matching game or flashcards, such as those within Quizlet. There will be scores associated with these minigames so the user can have some entertainment while learning the material.
* UI5: There will be five total chapters (listed below) with sections within each.
  + The Basics (printing, variable types, etc.)
  + If/Else Statements
  + Loops (While, For, Do, etc.)
  + Functions, Arguments, and Parameters
  + Classes (Constructors, Parent/Child Classes, etc.)

### 3.1.2 Hardware Interfaces

* There are no hardware interfaces that are needed for the creation of this project

### 3.1.3 Software Interfaces

* SI1: Informational Section
  + This section will flip through the information presented to the screen when the “Next” button is clicked or when the user correctly answers the practice question correctly.
* SI2: Practice Questions
  + This section will usually be an open-ended question (sometimes multiple choice) and will give the user the correct answer if they answered it correctly.
  + However, multiple variations of one answer, or if the user enters the word we were looking for, will be accepted.
  + Will not be case sensitive unless it must be (Example: char will be accepted but not Char because types must be lowercase)
  + Some types of open-ended questions will include definitions and code snippets (as well as the end-of-chapter quizzes)
* SI3: End-of-Chapter Quiz
  + Each question will have their own respective correct answer and will give the user what they got correct after they click the “Submit” button.
  + Most of these questions will be multiple choice questions but there will be a few open-ended as well.
  + Will follow the same grading process for open-ended questions as the practice questions.
* SI4: Minigames
  + SI4-01: Matching
    - There will be 5-8 (10-16) random sets blocks printed to the screen that the user will be able to drag across the screen. Each individual block will have a matching block associated with it.
    - If the user drags a block to its correct partner the blocks will disappear off the screen. If the answer is wrong however, the block that was dragged will go back to its starting position.
    - There will be a running timer at the top and will end when the user clears all the blocks off the screen. Following this, the user’s score will be presented to the screen, and they will be notified if they got a high score or not.
      * The timer will start when the user clicks the “Start” button.
    - The high score will be added to the screen following this until a new high score is reached.
  + SI4-02: Flashcards
    - These will be like open-ended questions presented in previous sections. The user will be able to fill in the blank for a part of a definition of a term or enter the word that is associated with the definition.
    - The processing will be like open-ended questions above, not case-sensitive but spelling will matter.
    - There will be a timer on this minigame as well that runs like the minigame timer, the time will start when the user clicks the start button and end when they finish all the flashcards. The application will tell the user if they got a high score or not at the end, depending on the previous high score.
  + SI4-03: Coding Minigame
    - Similar to the matching minigame, with a running timer and high score system.
    - The user will be prompted with 5 total coding problems and will be solving them as fast as possible.
    - To submit an answer the user will click the “submit” button and the code they typed will run and produce and output or a coding error.
    - If the output matches the correct output the user will move on to the next prompt until they finish all five. Once they submit the fifth the timer will stop.
* SI5: Chapter Summary and Visuals
  + Will summarize the material discussed in the chapter w/ definitions and important topics.
  + The visuals will be of coding snippets, applying the topics to examples.
    - The user will also be able to run these snippets and see their respective outputs as well.

### 3.1.4 Communications Interfaces

* For the textbook & the rest of the chapter the user will be able to click on “back/previous” and “next” buttons.
* For the practice questions & end-of-chapter quiz the user will be able to click on each of the multiple-choice options and be able to type in the textboxes. The user will then click a “submit” button to get their feedback on the question or set of questions.
* For the coding minigame the user will also be able to type in a textbox and click “submit” to run the coding snippets entered.
* For the matching minigame the user will be able to drag cards across the screen to match them with their respective pair.
* For the flashcards the user will be able to click on the flashcard to flip it to the other side. The user will also be able to press the arrow keys or arrows on the screen to transverse the flashcards.
* NOTE: The quiz questions, practice questions, and minigames will be communicated behind the scenes with a database that will hold the correct answers and help provide feedback. This will be discussed later.

## 3.2 Functional Requirements

This section describes specific features of the software project. If desired, some requirements may be specified in the use-case format and listed in the Use Cases Section.

### 3.2.1 Table of Contents with Hyperlinks (Implement 1st)

#### 3.2.1.1 Description and Priority

This will be the first screen the user sees after they click off the title slide and will lay out all the chapters and the sections that are within each one. The user will also be able to access all the minigames, chapter summaries, quizzes, etc. within the application to help with whatever they need to study up on. This is a 9-priority item (Very High) and should be created first as it helps organize all the material that will be touched on within the application.

**Benefit (9)** – This will show the user everything the application provides and gives them free reign and a swift way to move throughout it.

**Penalty (9)** – If this section is not implemented then it would be harder to access sections in later chapters and could be a hassle.

**Cost (2)** – This requirement is straightforward and is a good way to organize everything that will be within the application.

**Risk (1)** – The idea is not out-of-the-box and is an important piece for the application.

#### 3.2.1.2 Stimulus/Response Sequences

The user can click on any of the hyperlinks, and it will take it to its respective section.

#### 3.2.1.3 Functional Requirements

##### **3.2.1.3.1 REQ-1:** All the chapters and sections must be planned out to know what to hyperlink. However, the table of contents is subject to change based on feedback and time constraints.

### 3.2.2 Textbook Section to Learn Loops (Implement 2nd)

#### 3.2.2.1 Description and Priority

This section will be implemented at the start of the loops chapter, along with all the other chapters, and will include all the information needed to learn the Java concepts. Specifically, this chapter will go over different types of loops such as for, while, and do loops and when to use each type. This is a 9-priority item (High) as it is the most critical part needed for the user to learn that Java concept or chapter.

**Benefit (9)** – Most crucial element in teaching the user Java.

**Penalty (9)** – If the material in the textbook does not make sense to the user, then the application could be ineffective.

**Cost (9)** – This section must be planned out correctly to be able to efficiently teach the user Java, costing a lot of time in research and organization. However, if everything is organized then it helps flow smoothly with other components.

**Risk (5)** – If something is not explained correctly, something or covered too broadly, or not covered at all then the user could be lost while learning the material. However, this section is required for the application to function properly, lessening the risk.

#### 3.2.2.2 Stimulus/Response Sequences

There will be “next” and “back” buttons that the user is able to transverse throughout the chapter’s textbook portion.

#### 3.2.2.3 Functional Requirements

##### **3.2.2.3.1 REQ-1:** For the user to able to access the textbook for the loops chapter that table of contents must be hyperlinked and working.

##### **3.2.2.3.2 REQ-2:** For the user to be able to transverse onto the next part of the chapter, the user must answer all practice questions correctly. If a question is answered incorrectly the system will tell the user, they got the question wrong and give the user another chance to get it correct.

### 3.2.3 Loops Textbook Section Divided into Topic Sections (Implement 3rd)

#### 3.2.3.1 Description and Priority

This will add on to the requirement mentioned in 3.2.1.1. Once the loops section is written, bold subtitles will be added to organize the material within the chapter to make it easier for a reader to read and use. This is a 9-priority item (High) as it makes the application more organized and presentable while being an easy feature to implement.

**Benefit (9)** – By dividing the textbook section into chapters the material is more organized and easier to read.

**Penalty (7)** – The textbook section may be harder to understand, which is the core of the application.

**Cost (1)** – This requirement is easy to implement and could be done while implementing the textbook section if everything is well organized.

**Risk (1)** – The idea is not out-of-the-box and can be considered a necessity in helping the user learn Java.

#### 3.2.3.2 Stimulus/Response Sequences

There will be “next” and “back” buttons that the user is able to transverse throughout the chapter’s textbook portion like 3.2.2.2.

#### 3.2.3.3 Functional Requirements

##### **3.2.3.3.1 REQ-1:** For a user to be able to access this requirement within the chapter the user must go through the whole chapter or through the table of contents.

##### **3.2.3.3.2 REQ-2:** The loops textbook chapter must be written for the material to be organized for the user.

### 3.2.4 End-of-Chapter Quiz for the If Statements Chapter (Implement 4th)

#### 3.2.4.1 Description and Priority

This section will be placed at the end of the chapter, right after the chapter summary and visuals. This section will include 10-15 multiple choice and open-ended questions that the user will be able to answer to review their knowledge on the chapter. Once the user clicks the “submit” button they can see their grade on the quiz, what they got right and wrong, and retake it if they want. This is an 8-priority item (High) as it is a simple way to study and an easy way for the user to test their knowledge.

**Benefit (8)** – This is a way to better test the user’s knowledge by using a staple studying method.

**Penalty (7)** – The quiz will touch on all the important topics and concepts of the chapter while giving the user a feel on what could be on a real test for that chapter.

**Cost (6)** – A database must be implemented for each quiz question and answer along with a way to grade the test, which could take a good amount of time.

**Risk (1)** – The idea is not out-of-the-box and can be considered a necessity in helping the user learn Java.

#### 3.2.4.2 Stimulus/Response Sequences

The user can click on all multiple-choice options along with typing in each text box. They also can click on the “submit” button to get the grade on their quiz even if they didn’t answer every question.

#### 3.2.4.3 Functional Requirements

##### **3.2.4.3.1 REQ-1:** For a user to be able to access this requirement within the chapter the user must go through the whole chapter or through the table of contents.

##### **3.2.4.3.2 REQ-2:** A database with each of the questions/answers must be present to be able to grade the quiz correctly.

### 3.2.5 Matching Game for Input Statements and Definitions (Implement 5th)

#### 3.2.5.1 Description and Priority

This requirement will provide an entertaining spin to learning Java. This little minigame will allow the user to drag cards to match words with their definitions. This game will first be implemented into chapter 1 about Java basics and will soon be implemented into other chapters as well. This is a 8-priority item (High) as it gives a twist into learning Java along with making the application stand out more while also being more memorable.

**Benefit (9)** – Is an entertaining way for a user to learn Java and helps keep the user interested and intrigued thought the application.

**Penalty/Risk (8)** – Since this section can take a lot of time to implement then some sections of a lower priority may not be able to be implemented during the time constraint.

**Cost (9)** – This section will take a lot of time to create. A database must be created for the questions and answers along with some basic graphic design, requiring a lot of testing for the requirement to function properly.

#### 3.2.5.2 Stimulus/Response Sequences

The user will click the “start” button to start the minigame. There will be 10-16 cards printed to the screen which will be linked with 5-8 pairs. The user will be able to drag these cards across the screen to be able to match each definition with its term. Once a correct match is found the cards will disappear from the screen and once all the cards are cleared the user wins the minigame.

#### 3.2.5.3 Functional Requirements

##### **3.2.5.3.1 REQ-1:** For a user to be able to access the matching game they must go through the whole chapter or access it from the table of contents.

##### **3.2.5.3.2 REQ-2:** For a user to be able to complete this section they must clear all the cards on the screen by finding the correct pairs. If the user answers incorrectly then the cards do not disappear and go back to where they were located before the user last dragged them.

##### **3.2.5.3.3 REQ-3:** For a pair to be correct it must match the pairing within the database for that chapter.

##### **3.2.5.3.4 REQ-4:** Pairs must be organized and planned out along with their answers in the database.

##### **3.2.5.3.5 REQ-5:** A timer must be implemented for the minigame to have a scoring section.

### 3.2.6 Flashcards for the Parent/Child Classes Chapter (Implement 6th)

#### 3.2.6.1 Description and Priority

This section is an extra study resource for the user that they can find at the end of the parent/child classes, with this feature being added for other large chapters such as the loops chapter. The user will be able to flip through definitions and their terms like real-life flashcards. However, this will be done through clicking a mouse rather than physically flipping the cards. This is a 7-priority item (Medium) as it gathers main definitions from the chapter and converts them into an easy way to study.

**Benefit (7)** – Simple, repetitive way for the user to learn the definitions within the chapter.

**Penalty/Risk (3)** – This is a safe study option as Quizlet flashcards are a common practice to learn the definitions of a chapter.

**Cost (5)** – This requirement will not require a lot of time to implement since one flashcard would not take much time to implement. However, this will be a repetitive process as there could be up to 20 definitions per chapter.

#### 3.2.6.2 Stimulus/Response Sequences

The user will be able to click different arrow keys to move through the flashcards, back and forth. The user will also use the click of a mouse to flip the flashcard to see the other side. Once the user gets to the last flashcard, they can move on to the next section through the “next” button.

#### 3.2.6.3 Functional Requirements

##### **3.2.6.3.1 REQ-1:** For a user to be able to access this section the user must go through the whole chapter or through the table of contents.

### 3.2.7 Practice Questions During If Statements Chapter (Implement 7th)

#### 3.2.7.1 Description and Priority

This section will be implemented in chapter 2, and all other chapters as well, which includes the topic of statements. These questions will be multiple choice questions that will pop up when the user is ready to go on to the next part about the chapter. However, the user is not able to proceed until they get the question right. These questions will either be multiple choice or open-ended questions. This is a 5-priority item (Medium) as it is not necessary, but it would be a good feature in helping the user learn Java, the end goal.

**Benefit (7)** – Helps the user better reinforce their knowledge while reading the chapter.

**Penalty/Risk (3)** – If the textbook section is effective then the questions can only help the reader understand the material. If the user does not get a question correct, then they can go back and read the textbook section to see why they got it wrong.

**Cost (7)** – A database must be created with the quiz questions and answers to be able to give feedback. However, the questions will be simple and not complex coding problems so validation would be easier.

#### 3.2.7.2 Stimulus/Response Sequences

For a multiple choice question the user will be able to click the radio button for each multiple-choice answer and then click submit once they have made their final decision. For an open-ended question, a user can type whatever they choose into a textbox.

#### 3.2.7.3 Functional Requirements

##### **3.2.7.3.1 REQ-1:** For a user to be able to get to the practice question the “next” button must be pressed during after the user is done reading the textbook section for that chapter.

##### **3.2.7.3.2 REQ-2:** For a user to be able to complete this section they must answer the question correctly. This means the user will be given unlimited tries to get the practice question correct. This section will not include code statements; however, the open-ended will be based on definitions so case will not be a factor. Spelling, on the other hand, would result in an incorrect answer on the open-ended section.

##### **3.2.7.3.3 REQ-3:** For a correct answer to be found the user’s answer must be found within the database under that question for that specific chapter.

### 3.2.8 Timer Introduced into Minigames (Implement 8th)

#### 3.2.8.1 Description and Priority

This will add on to the requirements mentioned in 3.2.3.1 and 3.2.6.1. Once both types of minigames are created, a timer will be added to keep track of the user’s score. The timer will start once the user clicks the “start” button and stop once the user does all desired tasks in the minigame. The time will then be printed to the screen and along with notifying the user if they got a high score or not. The lowest time will be the user’s high score. This is a 4-priority item (Low), once the minigame is implemented, as it gives the minigame more of a purpose.

**Benefit (7)** – This requirement can add a competitive side to the minigames and keep the user entertained while learning.

**Penalty/Risk (2)** – This requirement doesn’t play a crucial role in the application and could only serve as a bonus to the application.

**Cost (4)** – A database must be implemented to store the high score and notify the user if they break the score. However, the timer itself is not that hard to implement.

#### 3.2.8.2 Stimulus/Response Sequences

The user can click the “play again” or “next” buttons once they are done viewing their results and are able to play the minigame again or move on to the next chapter or section.

#### 3.2.8.3 Functional Requirements

##### **3.2.8.3.1 REQ-1:** For a user to be able to access this requirement within the chapter the user must go through the whole chapter or through the table of contents.

##### **3.2.8.3.2 REQ-2:** At least one of the two minigames must be programmed and running for the timer to be implemented.

### 3.2.9 Coding Minigame to Understand Functions (Implement 9th)

#### 3.2.9.1 Description and Priority

This is another version of an end-of-chapter minigame, like the matching minigame. This section includes a minigame where the user is to try and solve five coding problems regarding functions as fast as possible. Since there are many ways to solve a coding problem the application will be looking for specific words and phrases to see if the user got all the main points of the question. The code will also be run to see if the user got the desired output the question asked and provide feedback otherwise. One the user enters all five prompts, they win the minigame. This is a 6-priority item (Medium) as one of these minigames, this or 3.2.5, would help make the application stand out more but both may not be necessary.

**Benefit (9)** – Is an entertaining way for a user to learn Java and helps keep the user interested and intrigued thought the application. This game is more complex, but it could be effective when it comes to learning coding.

**Penalty (6) –** Some other components may not be able to be implemented under the time constraint. However, there aren’t many components below this one on the priority list.

**Cost (9) –** This requirement will require the most amount of time out of any other requirements. There needs to be a working database with the questions and answers along with a correct grading system for the code blocks. Since there are many ways to solve a coding problem, the grading will be more complex. Also, research will need to be done to figure out how to validate the output of the coding block and provide feedback.

**Risk (9) –** Since this component is so complex it will need a lot of work to implement.

#### 3.2.9.2 Stimulus/Response Sequences

To start the minigame the user will click the “start” button. The user will be able to type in the textbox to be able to answer the prompt. Once the user enters the answer, they will click the “run” button. Once all five prompts are answered the user can click the “finish” button to win the minigame.

#### 3.2.9.3 Functional Requirements

##### **3.2.9.3.1 REQ-1:** For a user to be able to access this section the user must go through the whole chapter or through the table of contents.

##### **3.2.9.3.2 REQ-2:** For a coding prompt to be correct the user must enter an answer like the question linked in the database. Along with this, the output of the coding prompt must match the output of the one in the database as well.

##### **3.2.9.3.3 REQ-3:** Questions must be organized and planned out along with their answers in the database.

##### **3.2.9.3.4 REQ-4:** A timer must be implemented for the minigame to have a scoring/high-score section.

### 3.2.10 Chapter Summary Section for Functions Chapter (Implement 10th)

#### 3.2.10.1 Description and Priority

This section will be added right after the textbook/practice quiz questions section and gives the user a summary of the functions chapter and its main points. This will include important definitions and keywords that are important in the chapter. Will be like the end-of-chapter images but be the worded portion. This is a 4-priority item (Low) since the textbook sections will not be that large and long of a read so the chapter summary would not be a large benefit.

**Benefit (7)** –This requirement can sum up all the information in the chapter and be a good resource for the user to come back to for reviewing material.

**Penalty/Risk (2) –** If the textbook section is well written then this requirement will be a bonus. To go along with this, there will already be a lot of other study options created to help the user wrap up the chapter. Some other sections, like the flashcards for the definitions of the chapter, will include material that would be discussed in the chapter summary.

**Cost (2) –** If the material is well planned out and organized prior to writing the textbook section then this section will be quick and easy to implement.

#### 3.2.10.2 Stimulus/Response Sequences

None, except for the “next” and “back” buttons which the user can use to transverse to other sections of the chapter or the next chapter.

#### 3.2.10.3 Functional Requirements

##### **3.2.10.3.1 REQ-1:** For a user to be able to access this requirement within the chapter the user must go through the whole chapter or through the table of contents.

### 3.2.11 Visual Study Guide with Images (Implement 11th)

#### 3.2.11.1 Description and Priority

This section will be placed chronologically at the end of the loops chapter and include images of the outputs of what each loop, for and while for example, will print. These will be basic code snippets to give the user a feel of how each one works and when to use them. If the user wants to see the output of the coding snippets they will click the “run” button and will see the output printed below. This gives the user an ability to guess, in their head, what the snippet will output. This item is a 3-priority (Low) as these images will be talked about during the chapter and would not affect the flow if it was not implemented.

**Benefit (7)** –With the large number of visual learners, this requirement could help the user understand how loops are structured and how the outputs are produced.

**Penalty/Risk (1)** – Since this will be the last requirement implemented, this section gives an even larger variety of study options along. This requirement does not have a lot to it either, making it easy to implement.

**Cost (1)** – This is a small requirement, meaning it will not take much time to implement.

#### 3.2.11.2 Stimulus/Response Sequences

The user will be able to click the “run” button to see the output of the snippet, along with clicking on the “next” and “back” buttons which the user can use to transverse to other sections of the chapter or the next chapter.

#### 3.2.11.3 Functional Requirements

##### **3.2.11.3.1 REQ-1:** For a user to be able to access this section the user must go through the whole chapter or through the table of contents.

## 3.3 Use Cases

### 3.3.1 Use Case #1

|  |  |
| --- | --- |
| **Use Case Name** | Table of Contents to Access Material Throughout the Application |
| **Reference** | Section 3.2.1 |
| **Trigger** | The User Starts the Application |
| **Precondition** | All the titles of the Chapters and their Components are Displayed to the screen |
| **Basic Path** | 1. The User clicks on the Chapter/Section they wish to go to 2. The application takes the user to that requested location. 3. The user reads the textbook or summaries/takes the quiz/plays the minigame/uses the flashcards. 4. The user can click the “home” button in the top left to go back to the table of contents.   OR   1. The user clicks the “next” button to go to the textbook section of chapter 1. 2. The user reads the textbook section/takes practice questions and moves throughout the application. 3. The user can click the “home” button in the top left to go back to the table of contents. |
| **Postcondition** | The user is taken to the requested section. |
| **Exception Paths** | The reader can return to the table of contents at any time. |
| **Other** | All the material that will be discussed will be planned out and organized ahead of time to make it the most user friendly. |

Table 1: Use case 1

### 3.3.2 Use Case #2

|  |  |
| --- | --- |
| **Use Case Name** | Reading the Textbook Section [at the Beginning of the Chapter] |
| **Reference** | Section 3.2.2, Section 3.2.3 |
| **Trigger** | User clicks on the textbook section from the table of contents/Accessed by finishing the previous chapter. |
| **Precondition** | All the material that will be discussed within the chapter is well organized. |
| **Basic Path** | 1. The user accesses the chapter through the table of contents or by finishing the previous chapter. 2. The user reads the chapter and moves through it with the “next” and “back” buttons. 3. The user answers the in-text practice questions to move on to the next part of the section (will be mentioned later). 4. The user finishes the chapter and moves on to other study materials. |
| **Postcondition** | The user will learn the material discussed in the chapter and will be able to use that knowledge in later study sections. |
| **Exception Paths** | The reader can return to the table of contents at any time to go to other sections. |
| **Other** | All the material that will be discussed in the chapter will be planned out and organized ahead of time to reduce confusion for the user. |

Table 2: Use case 2

### 3.3.3 Use Case #3

|  |  |
| --- | --- |
| **Use Case Name** | Organized Textbook Material with Subtitles to Easy Reference a Part of the Textbook |
| **Reference** | Section 3.2.3 |
| **Trigger** | After accessing the textbook, the sections will be found by flipping through the textbook. |
| **Precondition** | The user will be reading the textbook to view the subtitles |
| **Basic Path** | 1. The user reads the textbook section of that chapter using the “next” and “back” buttons. 2. The user reads the subtitle when they come to know what the textbook will be talking about next. |
| **Postcondition** | The user will better understand the material they are reading in the textbook since it is organized better. |
| **Exception Paths** | The reader can return to the table of contents at any time to go to other sections without finishing the whole textbook section. |

Table 3: Use case 3

### 3.3.4 Use Case #4

|  |  |
| --- | --- |
| **Use Case Name** | End-of-Chapter Quiz |
| **Reference** | Section 3.2.4 |
| **Trigger** | User clicks on the chapter quiz for that desired chapter from the table of contents/Accessed by finishing previous sections of the chapter. |
| **Precondition** | The user has read the textbook section of that respective chapter to know what the chapter is discussing (Optional). |
| **Basic Path** | 1. The user clicks the “start” button to start the quiz. 10-15 quiz questions, multiple choice and open-ended, will be displayed to the screen. 2. The user will answer each of the questions to the quiz, however they do not need to answer all the questions. The quiz will still be graded out of the number of questions regardless of if the user doesn’t answer a question. 3. The user clicks the “submit” button to finish the quiz. 4. The user will see their grade to the quiz displayed at the top left and what questions they got right or wrong. |
| **Postcondition** | The user will be able to test their knowledge on the chapter and see what they need to review or work on. |
| **Exception Paths** | The reader can return to the table of contents at any time to go to other sections without finishing reading the quiz. However, if the user does that, they will lose all progress within the quiz. |
| **Other** | All the quiz questions and their respective answers will be stored in a database to be able to provide feedback to the user after they submit the quiz. |

Table 4: Use case 4

### 3.3.5 Use Case #5

|  |  |
| --- | --- |
| **Use Case Name** | Matching Minigame to Learn Terms & Definitions |
| **Reference** | Section 3.2.5 |
| **Trigger** | User clicks on the matching minigame for that chapter from table of contents/Accessed by finishing previous sections of the chapter. |
| **Precondition** | All terms/definitions will be linked within a database so feedback can be properly provided. |
| **Basic Path** | 1. The user clicks the “start” button to start the minigame. 10-16 cards will then be presented on the screen with there being a pair for each card. 2. The user can drag any card they want across the screen with the purpose of placing either the term or the definition with their respective pair. 3. Feedback will be provided.    1. If the user gets the pair correct the two cards will disappear from the screen.    2. If the user gets the pair wrong the card will go back to its original location and the application will tell the user they got it wrong. 4. The game will continue until all the cards are cleared from the screen. |
| **Postcondition** | The user will reinforce their knowledge discussed in the chapter through a fun and entertaining way. |
| **Exception Paths** | The reader can return to the table of contents at any time to go to other sections without finishing the game. This means all the user’s progress during the minigame will be forgotten and reset. |
| **Other** | Once the minigame is finished the user’s score on the timer will be printed to the screen (discussed later). Also, all the terms and their respective definitions will be stored in a database to be able to provide feedback to the user. |

Table 5: Use case 5

### 3.3.6 Use Case #6

|  |  |
| --- | --- |
| **Use Case Name** | End-of-Chapter Flashcards to Learn Terms & Definitions |
| **Reference** | Section 3.2.6 |
| **Trigger** | User clicks on the flashcards for that desired chapter from the table of contents/Accessed by finishing previous sections of the chapter. |
| **Precondition** | The user has read the textbook section, especially the definitions, for that respective chapter to know what the chapter is discussing (Optional). |
| **Basic Path** | 1. The user reads the term and guesses (in their head) what the definition is. 2. The user clicks the flashcard to flip it and see the term’s definition. 3. The user moves to the previous/next flashcard by clicking the arrows on the screen or using the arrow keys. 4. The user moves through all the flashcards to study all the definitions mentioned in the chapter. |
| **Postcondition** | The user will reinforce their knowledge discussed in the textbook in a repetitive and simple way. |
| **Exception Paths** | The reader can return to the textbook, or any other section, at any time to go to other sections without finishing the whole section. |

Table 6: Use case 6

### 3.3.7 Use Case #7

|  |  |
| --- | --- |
| **Use Case Name** | Practice Questions to Reinforce Knowledge from Textbook |
| **Reference** | Section 3.2.7 |
| **Trigger** | The user clicks the “next” button while reading the textbook. |
| **Precondition** | All the questions and answers are tied together within a database. |
| **Basic Path** | 1. The user reads the textbook and clicks the “next” button at the bottom. 2. A practice question appears (most likely multiple choice) which the user can freely answer. 3. The user clicks “submit” once they finalize their answer. 4. Feedback is returned if the user is right or wrong.    1. If right – User can move to the next part of the textbook.    2. If wrong – User must answer again till they are correct. They are also able to reread the textbook section to find the answer. |
| **Postcondition** | The user will better understand the material discussed within the textbook. |
| **Exception Paths** | The reader can return to the textbook section if they get the question wrong, however they cannot move on until they get the question correct. |
| **Other** | All the practice questions and their respective answers will be stored in a database to be able to provide feedback to the user. |

Table 7: Use case 7

### 3.3.8 Use Case #8

|  |  |
| --- | --- |
| **Use Case Name** | Timer Used to Score Minigames |
| **Reference** | Section 3.2.8 |
| **Trigger** | The user clicks the “start” button on either the matching or coding minigames. |
| **Precondition** | The user has read the textbook section of that respective chapter to know what the chapter is discussing (Optional). |
| **Basic Path** | 1. The timer will be displayed to the screen and set to 00:00.00. The timer will then start running when the user clicks the “start” button. 2. The user plays the minigame, with the timer running in the background, until they finish the minigame. 3. The timer will then stop once the user wins the minigame. 4. The user’s score on the timer will be displayed on the screen. 5. The user is told if they got a “high score” or not. This is done if the user has a lower time than their current best. If this is the user’s first time completing the minigame, however, that will be their high score. |
| **Postcondition** | The user will then have a competitive aspect and an incentive twist added to the minigame. |
| **Exception Paths** | The reader can return to the table of contents at any time to go to other sections without finishing the minigame. This means the timer will be voided and will be reset for when the user enters the minigame again. |

Table 8: Use case 8

### 3.3.9 Use Case #9

|  |  |
| --- | --- |
| **Use Case Name** | Coding Minigame |
| **Reference** | Section 3.2.9 |
| **Trigger** | User clicks on the coding minigame for that chapter from the table of contents/Accessed by finishing previous sections of the chapter. |
| **Precondition** | The user has read the textbook section of the chapter to understand what the coding minigame will cover (Optional). |
| **Basic Path** | 1. The user clicks the “start” button to start the minigame. 2. One of five coding prompts and the desired output will be presented along with a textbook to type the answer. 3. The user will type the code into the textbox. 4. The user clicks the “run” button. 5. Feedback will be provided.    1. If correct – the user will see the matching outputs.    2. If incorrect – the user will see what they did wrong and will be able to update their code until they get it right. 6. The user will answer the rest of the prompts correctly, enabling the “submit” button. 7. The user clicks the “submit” button to win the minigame. |
| **Postcondition** | The user gets coding experience of important Java concepts and will be able to reinforce their knowledge from the textbook in an entertaining manner. |
| **Exception Paths** | The reader can return to the textbook, or any other section, at any time to go to other sections without completing the minigame. This means all the user’s progress during the minigame will be forgotten and reset. |
| **Other** | The user is not able to skip a question and must get it right before moving onto the next one. Also, once the minigame is finished the user’s score on the timer will be printed on the screen (discussed later).  To be able to give the user feedback and tell them if they are right or wrong, all the prompts and their answers and outputs will be stored within a database. |

Table 9: Use case 9

### 3.3.10 Use Case #10

|  |  |
| --- | --- |
| **Use Case Name** | End-of-Chapter Summary |
| **Reference** | Section 3.2.10 |
| **Trigger** | User clicks on the chapter summary for that desired chapter from the table of contents/Accessed by finishing previous sections of the chapter. |
| **Precondition** | The user has read the textbook section of that respective chapter to know what the chapter is discussing (Optional). |
| **Basic Path** | 1. The user reads the chapter summary and sees all the important topics/definitions mentioned. 2. The user can choose to go back to the textbook to reread material OR move onto other sections |
| **Postcondition** | The chapter summary should wrap up all the material covered and tie up any loose ends. |
| **Exception Paths** | The reader can return to the table of contents at any time to go to other sections without/after finishing reading the chapter summary. |

Table 10: Use case 10

### 3.3.11 Use Case #11

|  |  |
| --- | --- |
| **Use Case Name** | End-of-Chapter Visual Coding Snippets & Outputs |
| **Reference** | Section 3.2.11 |
| **Trigger** | User clicks on the end-of-chapter visuals for that chapter from the table of contents/Accessed by finishing previous sections of the chapter. |
| **Precondition** | The user has read the textbook section for that respective chapter to know what the chapter is discussing (Optional). |
| **Basic Path** | 1. The user reads/views the coding snippets along and can guess (in their head) about what the output is. 2. The user can click the “run” button to get the output of the snippet. 3. The user can transverse through the section with the “next” and “back” buttons on the bottom. |
| **Postcondition** | The user, especially visual learners, will reinforce their knowledge discussed in the chapter with in-depth examples. |
| **Exception Paths** | The reader can return to the table of contents at any time to go to other sections without finishing the whole section. |

Table 11: Use case 11

## 3.4 Non-Functional Requirements

### 3.5.1 Performance

* The application should take no longer than 10 seconds to respond when launched.
* The application should take no longer than 5 seconds to reveal the correct answer to quizzes when prompted to.

### 3.5.2 Reliability

* The application should perform consistently across different hardware configurations and operating systems.
* The application should effectively handle any errors, such as runtime errors, and provide the user with an informative error message describing the issue the application is dealing with.

### 3.5.3 Availability

* This application will be available on all operating systems.
* This application will be available offline, ensuring availability of the application at all times.

### 3.5.4 Security

* Only certain people within the backend team will be permitted to access the database containing answers to quiz and minigame questions.

### 3.5.5 Maintainability

* When needed this application will go through maintenance updates to ensure that everything is working properly and that there are no issues with the application.

### 3.5.6 Portability

* This application will be available across different hardware configurations and operating systems.

## 3.5 Design Constraints

One design constraint deal with the power of the user’s computer. This application should run well on most computers, but some old computers may struggle to run the minigames. The user also must have enough memory to download the application as well. Another constraint could be that a high school student may be restricted to one chapter at a time based on how their course works.

## 3.6 Logical Database Requirements

A database will be used and is a necessity for some practice sections during or at the end of the chapter. The practice questions during the textbook, the end-of chapter quiz, and minigames will require the database in some form. The database will be separated by requirements through tables with each requirement having their own storage for the questions, correct answers, and user’s answer to the questions. For the coding minigame, however, the correct answer will be separated into two sections: the correct output and keywords required in the open-ended question. This means there will be a two-part validation process when grading the user’s answer. On the other hand, the quiz questions and practice questions’ correct answer section will contain a multiple-choice letter or a small open-ended answer. Finally, the matching minigame will have storage for the term and the definition rather than question and correct answer since the user can match the term with the definition or vice versa. All these columns will be of the string type and handled like such during validation of the user’s answer. After the practice section is refreshed or finished the user’s answers to the questions will not be retained any longer. To keep data integrity, though, the questions/correct answers and term/definitions values will be private to prevent data leakage.

## 3.7 Other Requirements

There are not many other requirements that are not listed. However, there will be a title slide with a “start” button for the application which will help pique the user’s interest when first starting the application.

# 4. Analysis Models

## 4.1 Sequence Diagrams

A diagram of a application

Description automatically generated

Figure 1: Textbook Section to Learn Loops (Use Case 1)

A diagram of a book

Description automatically generated

Figure 2: Textbook Section to Learn Loops (Use Case 2)

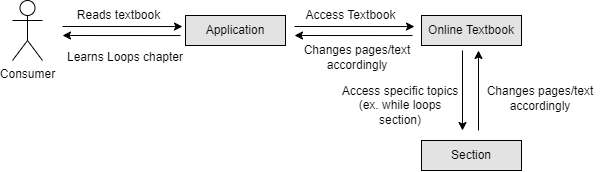


Figure 3: Loops Textbook Section Divided into Topic Sections (Use Case 3)

A black background with white text

Description automatically generated

Figure 4: End-of-Chapter Quiz for the If Statements Chapter (Use Case 4)

A diagram of a computer

Description automatically generated

Figure 5: Matching Game for Input Statements and Definitions (Use Case 5)

A diagram of a computer program

Description automatically generated with medium confidence

Figure 6: Matching Game for Input Statements and Definitions (Use Case 6)

A black background with white text

Description automatically generated

Figure 7: Practice Questions During If Statements (Use Case 7)

A black background with white text

Description automatically generated

Figure 8: Timer Introduced into Minigames (Use Case 8)

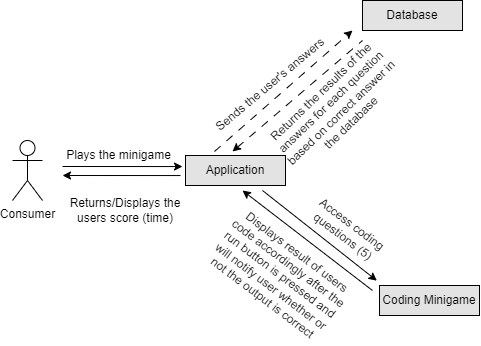


Figure 9: Coding Minigame to Understand Functions (Use Case 9)

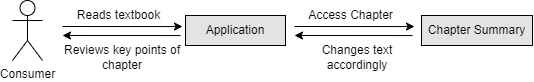


Figure 10: Chapter Summary Section for Functions Chapter (Use Case 10)

A black background with white text

Description automatically generated

Figure 11: Visual Study Guide with Images (Use Case 11)

# 5. Change Management Process

# References

# Appendices

## A.1 Appendix 1

The process that will be used to update the SRS is to go from start to finish and change the requirement piece-by-piece. When updating a requirement, once one section is completed it is important to cross-reference it after moving on to make sure everything still flows smoothly. If a requirement is being added it is important to make sure there are no redundancies during the creation of it. Also, if a requirement is being deleted it is important to check if no other requirements are dependent on it. If the project scope is being changed, however, it is also important to cross-reference every requirement and its use cases to make to see what is not required anymore and if anything needs to be added. It is important for only one person to submit the changes to prevent corruption or deletion of data and that one person can be whoever the team leader is in this case. These changes will be approved before submitting them by the boss of the company, or professor.