Destiny Deleon, Anthony Ephault, Max Flannick

DeSales University | [Company address]

**Java Learning Application**

DESIGN DOCUMENT

**Software Design Document Specification Template**

The Software Design Specification (SDS) sections provide you with guidelines related to the structure and the contents of SDS document. The Software Design Specification document includes at least these sections.

For the project, your team may have good reasons for wanting to deviate from this proposed outline. If a section is not applicable in your case, do not delete it; instead, give the topic heading and write "Not applicable".

You will note that there is some overlap in the content between different documents (i.e. the User Requirements Specification Document and the Software Design Specification Document.) This redundancy allows each document to stand on its own.

***ONLY THE SECTION TITLES COLORED IN ORANGE ARE REQUIRED TO BE COMPLETED.***

***DO NOT DELETE THE SECTIONS YOU ARE NOT COMPLETING AS THEY ARE A PART OF THE DOCUMENT***

Contents

[1 Introduction 3](#_Toc505762450)

[1.1 Purpose of this document 3](#_Toc505762451)

[1.2 Scope of the development project 3](#_Toc505762452)

[1.3 Definitions, acronyms, and abbreviations 3](#_Toc505762453)

[1.4 References 3](#_Toc505762454)

[1.5 Overview of document 3](#_Toc505762455)

[2 System architecture description 3](#_Toc505762456)

[2.1 Overview of modules / components 3](#_Toc505762457)

[2.2 Structure and relationships 3](#_Toc505762458)

[2.3 User interface 3](#_Toc505762459)

[2.4 User interface issues 3](#_Toc505762460)

[3 Detailed description of components (ONLY 2 ARE REQUIRED) 3](#_Toc505762461)

[3.1 X Component (or Class or Function ...) 3](#_Toc505762462)

[3.2 Y Component (or Class or Function ...) 4](#_Toc505762463)

[4 Reuse and relationships to other products 4](#_Toc505762464)

[5 Design decisions and tradeoffs 4](#_Toc505762465)

[6 Pseudocode for components 4](#_Toc505762466)

[7 Appendices (if any) 4](#_Toc505762467)

[Software component template for section 3 4](#_Toc505762468)

# Introduction

## Purpose of this document

Full description of the main objectives of the SDS document.

## Scope of the development project

This will be similar to what was written in the SRS.

## Definitions, acronyms, and abbreviations

Be sure to alphabetize!

## References

This section will include technical books and documents related to design issues. Be certain that the references you give are complete and in the appropriate format.

## Overview of document

A short description of how the rest of the SDS is organized and what can be found in the rest of the document. This is not simply a table of contents. Motivate and briefly describe the various parts!

# System architecture description

## Overview of modules / components

This subsection will introduce the various components and subsystems.

## Structure and relationships

Make clear the interrelationships and dependencies among the various components. Structure charts can be useful here. A simple finite state machine can be useful in demonstrating the operation of the product. Include explanatory text to help the reader understand any charts.

## User interface

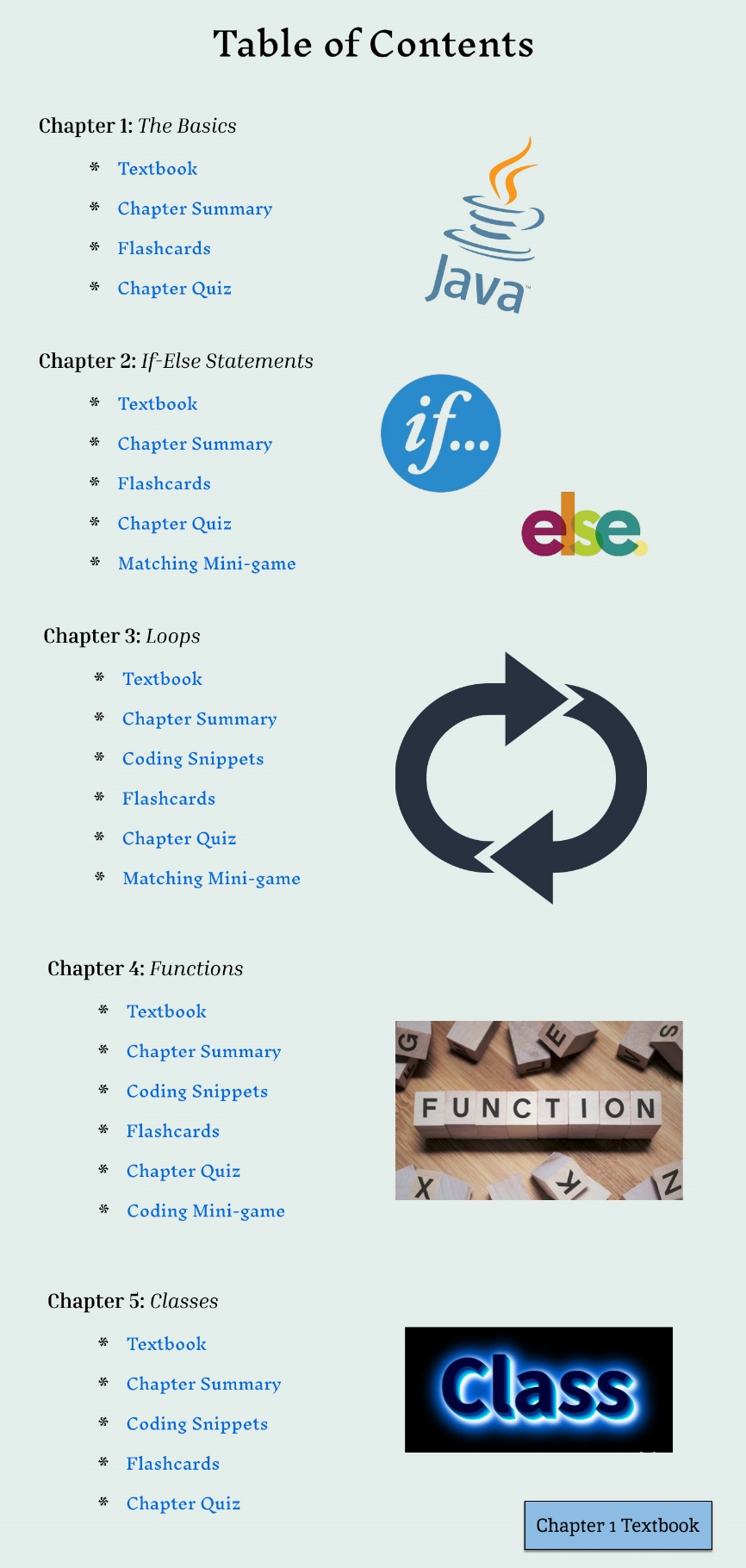


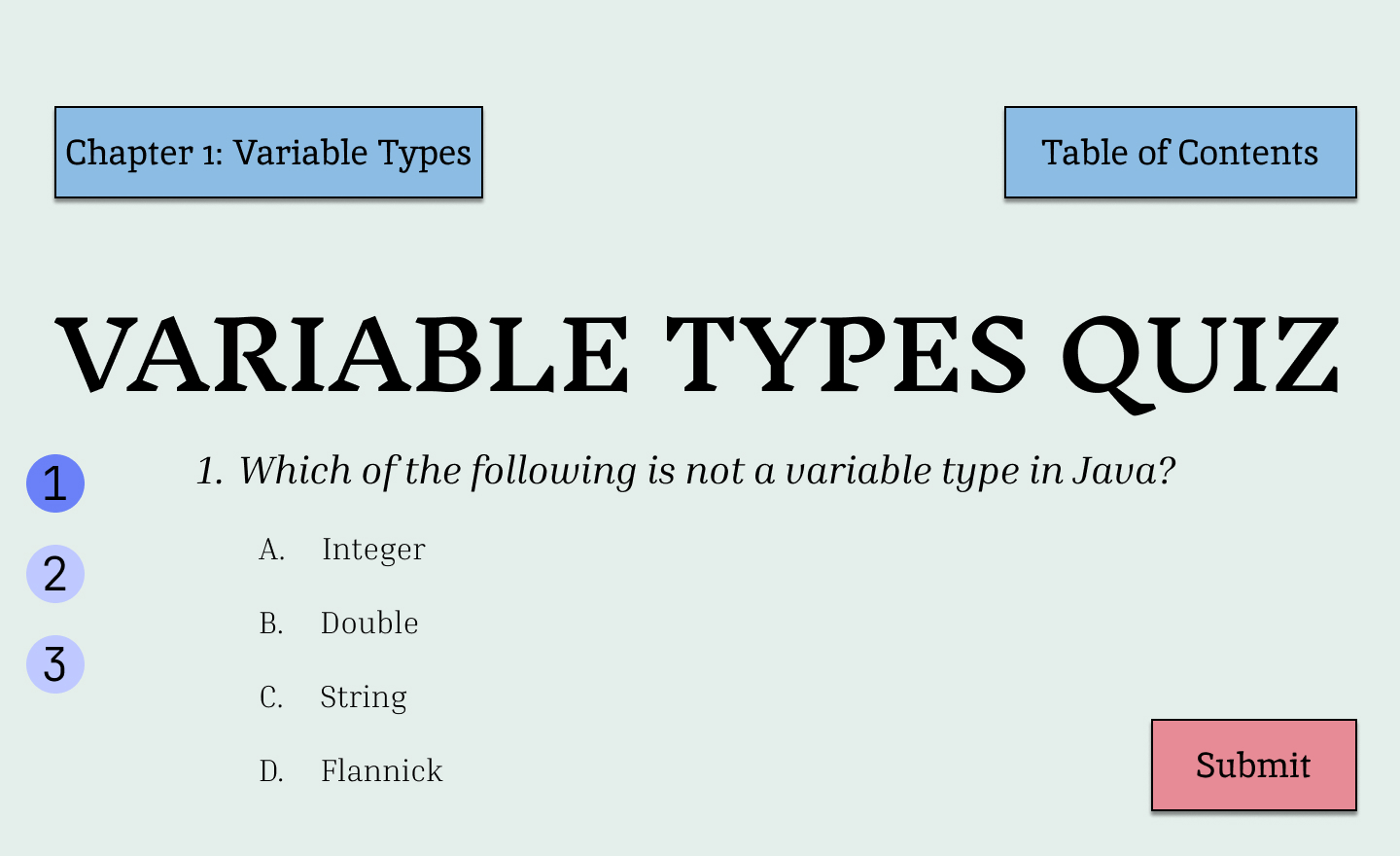
Table of Contents

This user interface allows the user to click on the respective section they would like to view. Each set of text will be hyperlinked with their correlating section, meaning there is no prerequisite needed to view the section. However, sections are organized in chronological order meaning it’s recommended that those sections are to be finished first, especially the textbook, before moving onto the next. The user is also able to click on the “Chapter 1 Textbook” button to move onto the first section chronologically which is the textbook of chapter one. One important factor when deciding the user interface for this section is the images that correlate with the respective section along with text font. [Note: For all of the following sections the user is able to click on the “Table of Contents” button move back to the table of contents at any time]



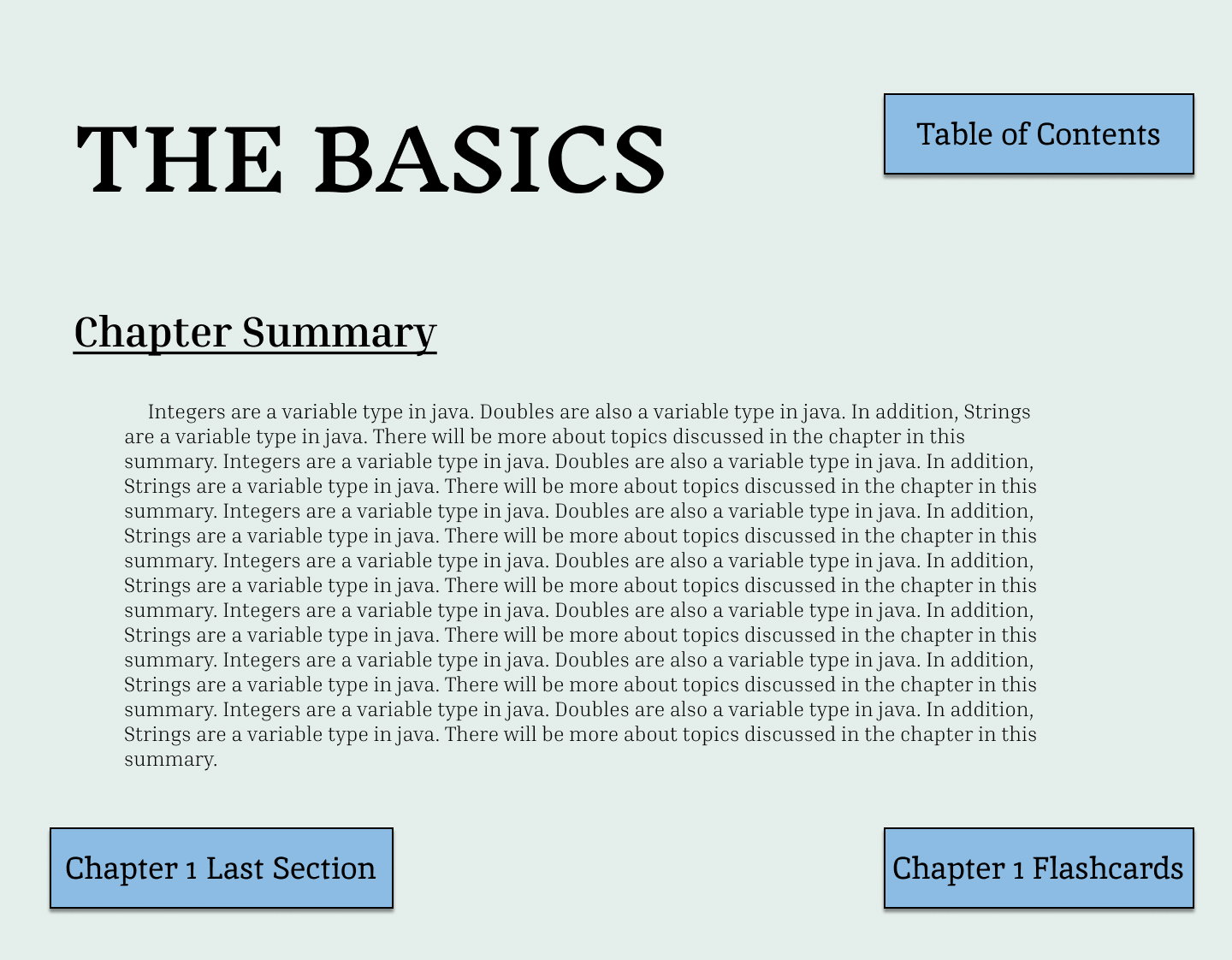
Textbook Section

This section involves a screen filled with text of important topics and information correlating with their respective chapter. This section is very important as it will relay down onto further sections and include everything the user needs to know about that subject in Java. The user is able to click on the “Section Quiz”, “Next Section” and “Chapter Summary” buttons to move throughout the chapter. Once the user is at the last section of the chapter the “Next” button will disappear and only the “Chapter # Summary” and “Section Quiz” buttons will remain. One deciding factor for this interface was formatting of the text on the screen.



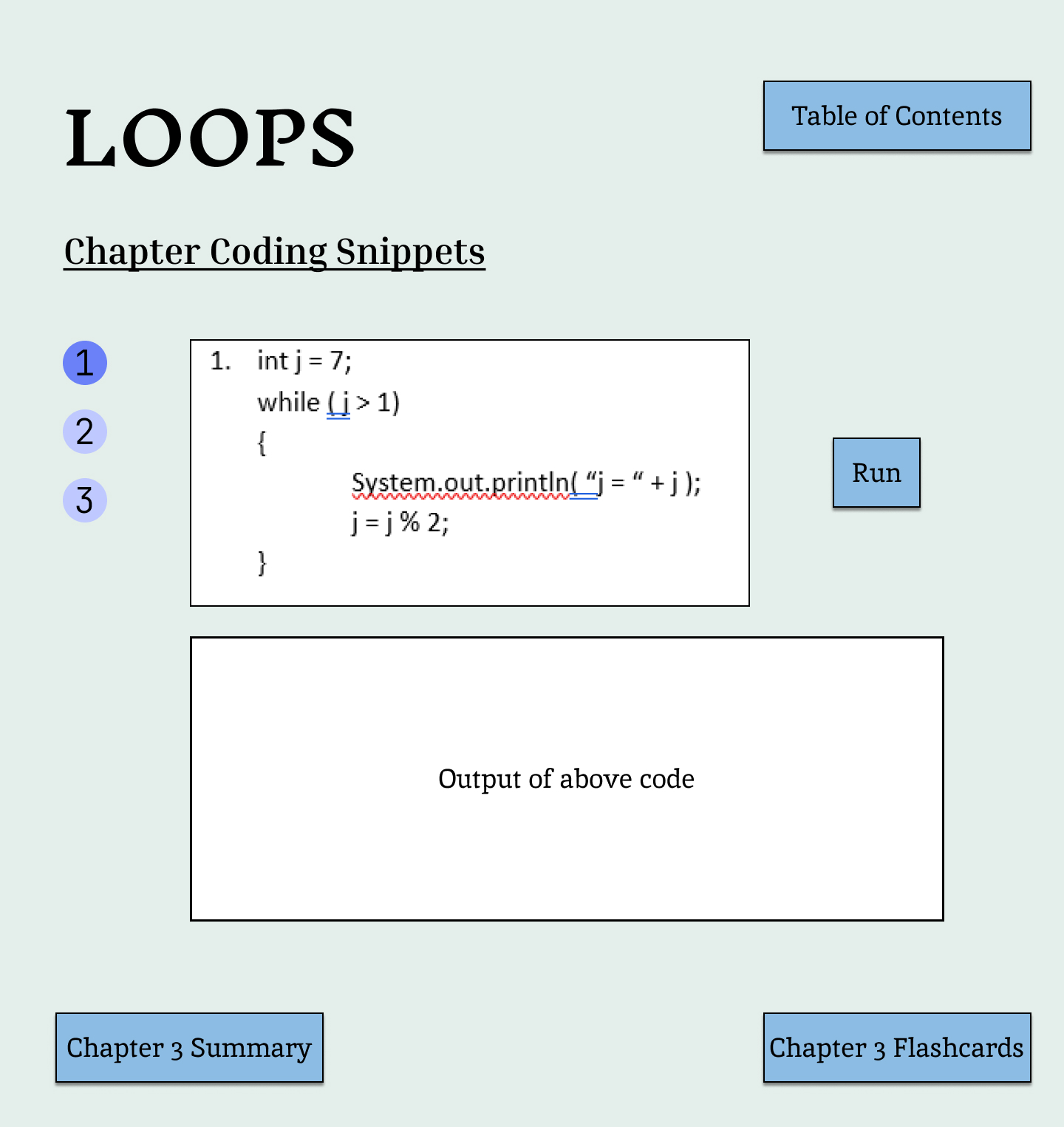
Practice Questions

This screen will appear after the user clicks on the “Section Quiz” button within the textbook. However, this screen will only appear once after clicking the “Section Quiz” button. Once the user clicks the button, a new window with a multiple choice question will open which the user will be required to answer. The user will not be able to move on in the textbook until they get the question correct, however they will be able to still view the textbook to find the correct answer. The user can navigate through the quiz questions by pressing the numbers on the left. The user is able to click on all the buttons and click “Submit” once they finalize their answer. Feedback is once provided once “Submit” is clicked and the user will be able to see if they got the question right. If the user gets the questions right the window will close and they will be able to return to the textbook. However, the user can close the practice question window at any time but they will not be able to move onto the next section of the textbook. One factor that went into deciding this interface was the layout of the screen and that it would be a separate window.



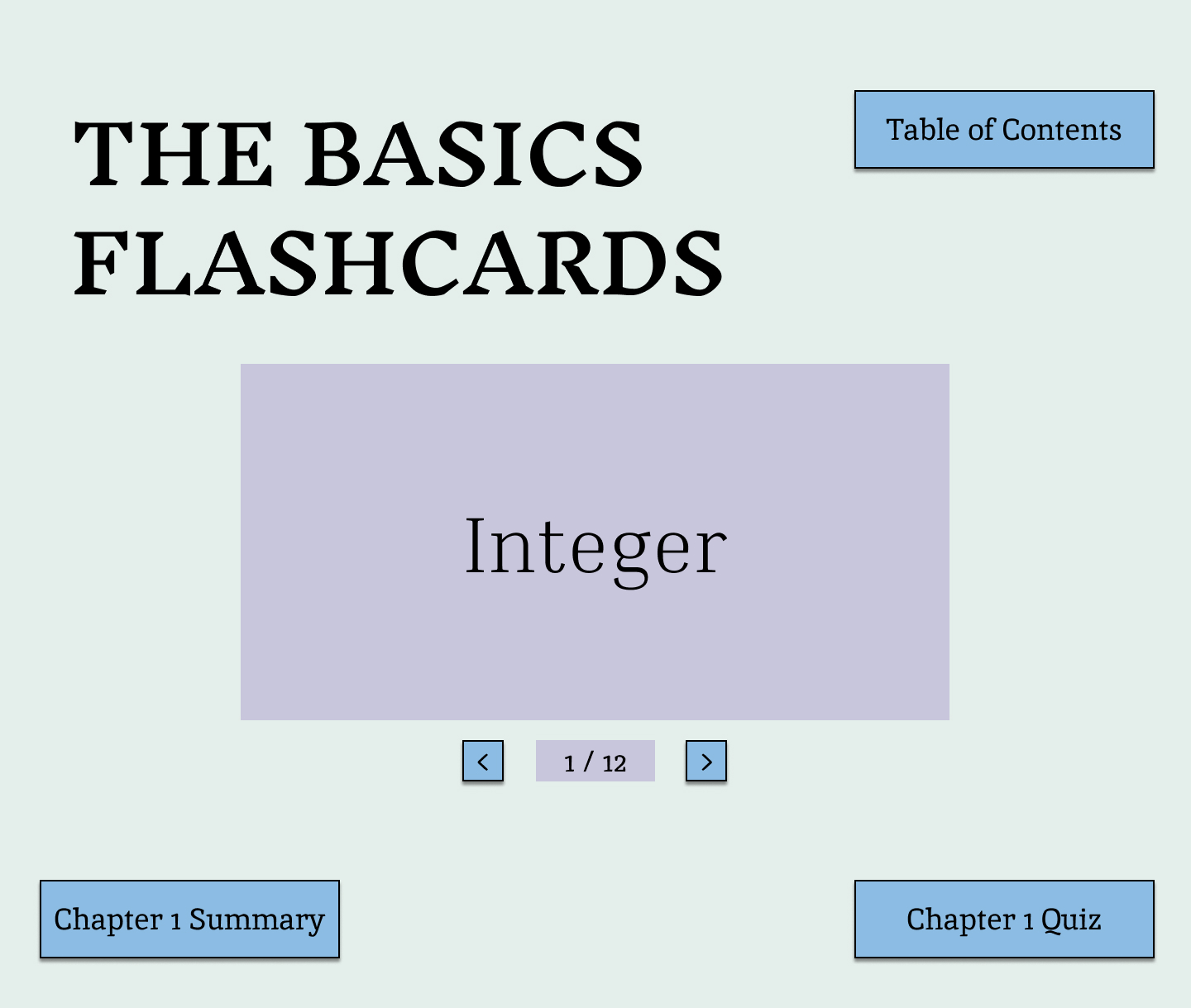
End-of-Chapter Summary

This section is similar to the textbook section but all of the information is on one screen. The user will be able to click on the “Last Section” and “Chapter # Flashcards” buttons to move back to the last part of the textbook or onto the chapter flashcards. Since this interface is similar to the textbook, we tried to make these sections look alike for the best user experience.



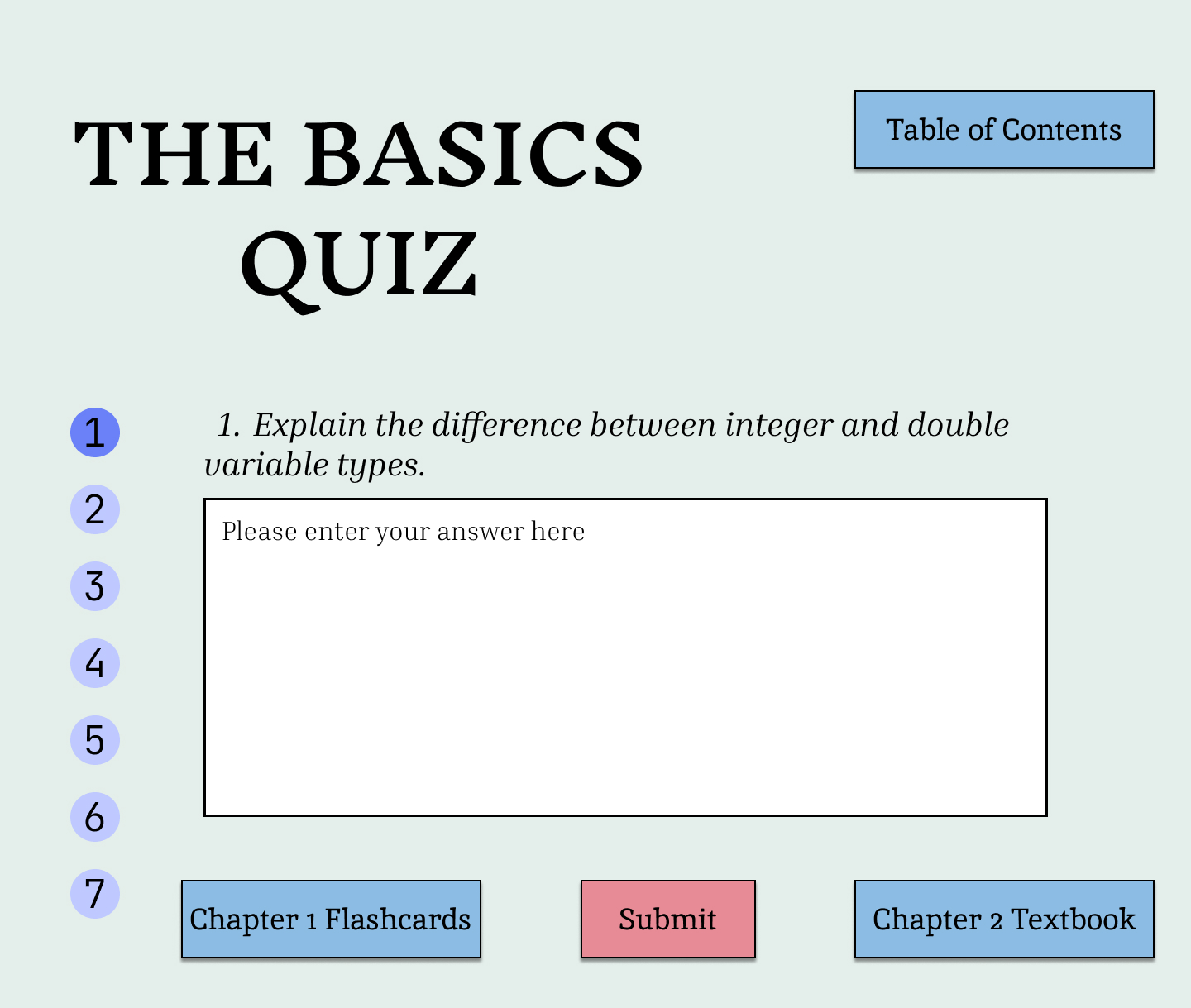
End-of-Chapter Visuals

This section is similar to other sections as the user will still be able to click on the “Chapter # Summary”, “Chapter # Flashcards” and “table of contents” buttons to traverse through the application. However, this section includes interactive material of coding snippets that were mentioned throughout the textbook that the user will be able to run. The user will click on the “Run” button to run the coding snippet displayed in the box above which will give the output if it was run by a Java compiler. The formatting of the boxes and how they will be aligned was a factor that came into deciding the user interface we would use for this section.



End-of-Chapter Flashcards

This section will consist of multiple flashcards, 10-15 cards in total, where the user will be able to refresh their knowledge on the definitions mentioned in the textbook. The user will click on the card to flip to the other side, revealing its respective term or definition. The user will also click either the arrow keys or the arrows on the screen to transverse across the flashcards. It is important to note that the “Chapter # Summary” and “Chapter # Quiz” buttons will be used for traversing sections and not the flashcards. The type of buttons that will be used to move throughout the flashcards and where they will be displayed were key factors when deciding this interface.



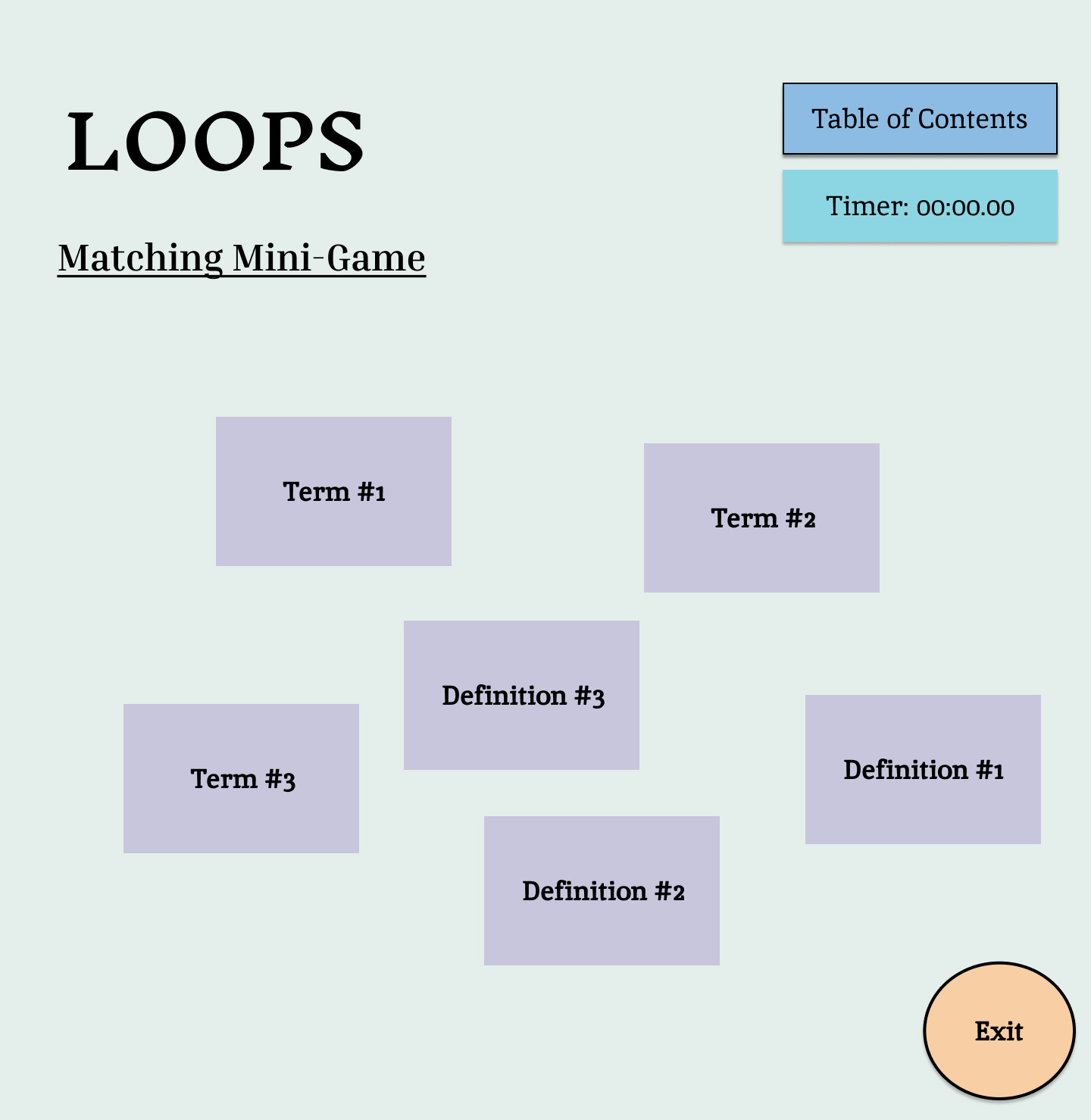
End-of-Chapter Quiz

This section includes 10-15 total questions consisting of multiple choice and open-ended sections. The user will be able to freely answer each question by clicking on the bubbles or typing in the textbox. Once the user enters each question, they will click the “submit” button to grade their quiz. An X or a checkmark will then appear next to each question showing the user what they got right and wrong. If the user got a question wrong the correct answers will appear as a way of feedback for the user. A grade will also appear on the screen based on the number of questions the user got right over the number of questions. The user is not required to answer every question, however, but they will just get that question wrong. The user is able to still click the “Chapter # Flashcards” and “Chapter # Textbook” buttons while taking the quiz but if they exit they will lose all of their progress. Some factors we looked at when considering the user interface of the quiz is the layout of all of the questions and how feedback will be provided.



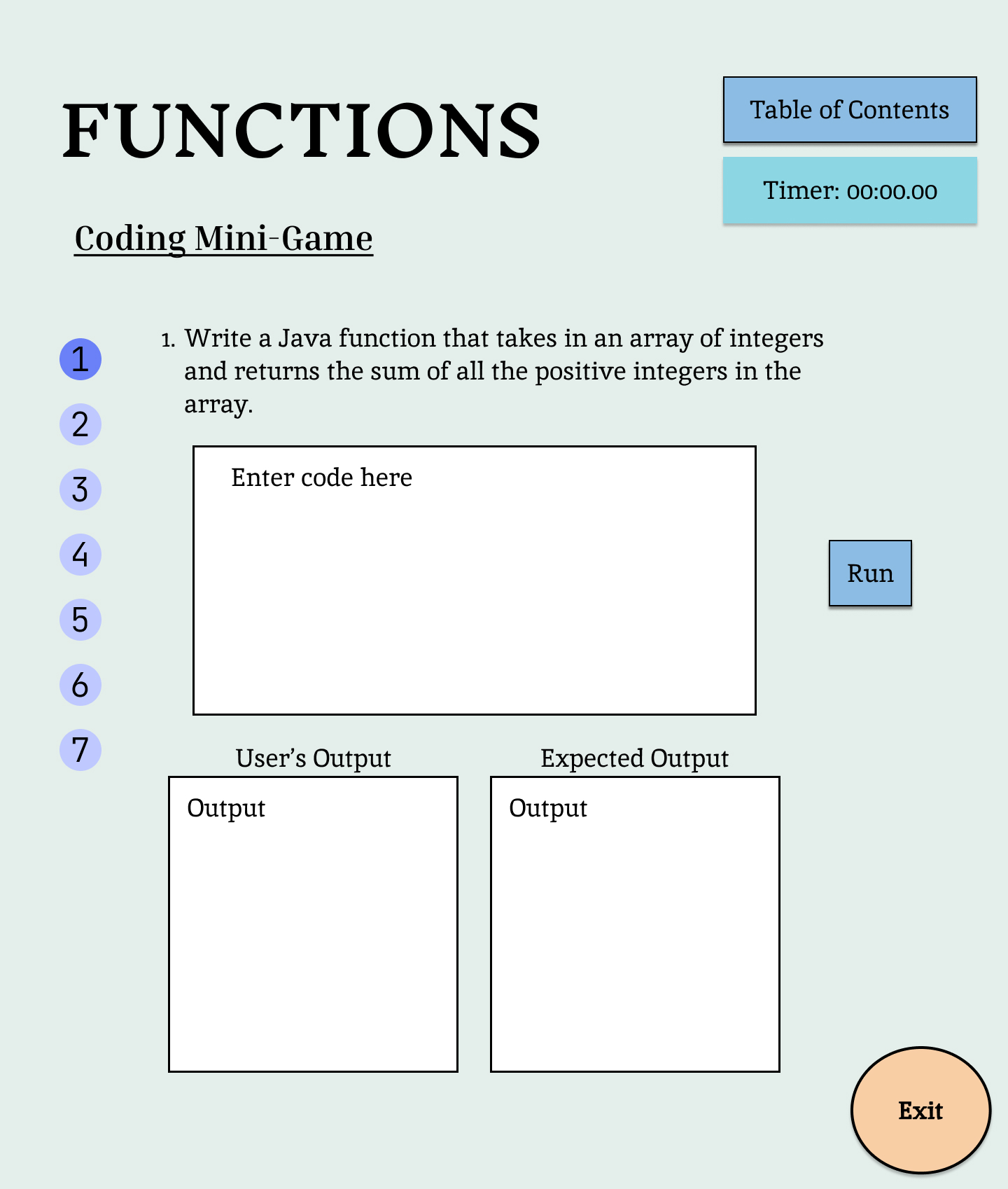
Minigame Start Screen

This user interface is very simple as the only unique button, outside of the “Chapter # Textbook”, “Chapter # Quiz”, and “table of contents” buttons, is a “Start” button. The user will click this button to start the respective minigame whether it is a coding or matching minigame. Once the user clicks “Start” a timer will also start to run which will be presented to the user once they complete the minigame. Outside of formatting and color, there were not many factors when it came to deciding the user interface for this part.



Matching Minigame

This is the first of two minigames within the application. Once the “start” button is clicked the user is presented with 5 pairs of flashcards, 10 in total, where the user’s goal is to match the term with its respective definition. These flashcards will correspond to the material learned within the respective textbook and flashcards section. The user will be able to drag the flashcards over the screen to match them with their respective partner which will be done by dragging the card over another card. Once the user stops dragging the card feedback will be provided based on if the user matched the two cards correctly. If the cards are correctly matched then the two cards will disappear from the screen, but if the user gets the match wrong then the two cards will go back to their respective positions before the user starts dragging them. The user will not be able to click on any chapter related buttons during play but will be able to go back to the table of contents at the expense of their progress. However, if the user completes the game then the user’s score, the time on the timer, will be displayed to the screen and if they get a high score or not. The color of the cards and layout of the screen were important factors when deciding which user interface to implement.



Coding Minigame

This section is similar to the matching minigame based on the start and end of the minigame. However, the minigame is way different as the user will be prompted with a coding prompt and will be directed to type the code to solve the respective problem. There will also be a desired output textbox, that the user will not be able to edit, that will show what output we are looking for. Once the user types the answer they think will work, they will click the “run” button and an output will be displayed in the “user output” textbox showing either the output or an error message. A checkmark will also appear on the screen if the user got the question right and be able to move onto the next one. There will be five coding prompts in total which the user will be required to answer to beat the minigame but they will not be able to move on to the next prompt without answering the previous one. Just like the matching minigame the user will also not be allowed to click on any chapter related buttons but can go back to the table of contents at the expense of their progress. The layout of all the buttons and textboxes were important in deciding this section in order to make it the most user friendly.

## User interface issues

There are many issues that may occur when creating the user interface for our Java Learning Application. One thing we will have to look out for is that there is seamless flipping between the front and back slides of the flashcards. We will also need to make sure the links on the table of contents page properly brings the user to the links designated page. We will also need to ensure that the buttons interact properly with the pages they should lead the user to. Finally, we must make sure that the grading for the open-ended questions is accurate. We need to make sure that the program will pick up on keywords the user may use in order to determine the correct grade for the user’s answer.

# Detailed description of components (ONLY 2 ARE REQUIRED)

## X Component (or Class or Function ...)

Use exactly the template shown at the end of the document.

## Y Component (or Class or Function ...)

...

Examples of a component are:

1. Database
2. Server
3. Client application
4. Search Application
5. Etc.

# Reuse and relationships to other products

For teams doing enhancement work, reuse is an important issue. Most enhancement work should focus on extending, rather than replacing, the design and product development from earlier semesters. For teams doing new development, reuse can also be an important strategy. In some cases, there is freeware that could be incorporated. In other cases, there are existing modules or classes that could be adapted. Another possibility is the use of special tools that produce open source results and thus permissible under the terms of this course.

This section should include the following subsections as appropriate:

* How reuse is playing a role in your product design
* How reuse is playing a role in your product implementation (and the motivation for changes)
* If you are not reusing material that is available, then give motivation for why it is being thrown out.

# Design decisions and tradeoffs

This project aims to support individuals unfamiliar with Java programming or those seeking a refresher. The application offers a comprehensive textbook guiding users through the step-by-step process of coding in Java. Additionally, users can engage with minigames designed to reinforce coding concepts, along with a matching game aiding in the retention of textbook sections. Notably, descriptions were added for each of the 11 sections created, enhancing user understanding and navigation. To enrich the learning experience, two minigames were implemented instead of solely relying on the matching game, and a flashcards section was added to complement the textbook content. Furthermore, formatting issues were addressed, resulting in improved visual aesthetics of the user interface, including a brighter color screen, the changing of the font, and the creation of a next button on the table of contents. The addition of images to the table of contents further enhances user engagement and navigation within the application.

# Pseudocode for components

Utilize the use cases to create pseudocode for components.

# Appendices (if any)

# Software component template for section 3

The template given below suggests a reasonable structure for giving a thorough description of each component described in Part 3 of the SDS. The specific information depends in part on the design approach. Your team must adapt this template to your needs and describe it in section 3.1 of your SDS.

|  |  |
| --- | --- |
| Identification | The unique name for the component and the location of the component in the system. |
| Type | A module, a subprogram, a data file, a control procedure, a class, etc. |
| Purpose | Function and performance requirements implemented by the design component, including derived requirements. Derived requirements are not explicitly stated in the SRS, but are implied or adjunct to formally stated SDS requirements. |
| Function | What the component does, the transformation process, the specific inputs that are processed, the algorithms that are used, the outputs that are produced, where the data items are stored, and which data items are modified. |
| Subordinates | The internal structure of the component, the constituents of the component, and the functional requirements satisfied by each part. |
| Dependencies | How the component's function and performance relate to other components. How this component is used by other components. The other components that use this component. Interaction details such as timing, interaction conditions (such as order of execution and data sharing), and responsibility for creation, duplication, use, storage, and elimination of components. |
| Interfaces | Detailed descriptions of all external and internal interfaces as well as of any mechanisms for communicating through messages, parameters, or common data areas. All error messages and error codes should be identified. All screen formats, interactive messages, and other user interface components (originally defined in the SRS) should be given here. |
| Resources | A complete description of all resources (hardware or software) external to the component but required to carry out its functions. Some examples are CPU execution time, memory (primary, secondary, or archival), buffers, I/O channels, plotters, printers, math libraries, hardware registers, interrupt structures, and system services. |
| Processing | The full description of the functions presented in the Function subsection. Pseudocode can be used to document algorithms, equations, and logic. |
| Data | For the data internal to the component, describes the representation method, initial values, use, semantics, and format. This information will probably be recorded in the data dictionary. |