Quicode Final Report

Helio Dong, Sylvie Dyer, Prarthana Gajjala,
Nivita Patri, Ana Pérez Céspedes, Kim Tran

1 Introduction

1.1 Purpose

This document provides an overview of the finalized initial release of the Quicode application. It includes background information for the project, outlines the initial scope and functionality, and describes the technical specifications for the application's frontend and backend. Using these designs, it examines the results of the project, including its compliance with the requirements, and achieved functionality.

1.2 Background

As coding becomes a necessary skill for an increasing number of fields, providing access to intuitive, lightweight computer science education becomes increasingly important. This mobile application attempts to address this through short, systematic, gamified practice questions.
By providing bite-sized lessons to users, this application will cater to audiences in a few manners:

- Users who don't have large amounts of free time can practice for short periods of time, while building intuition towards computer science.
    - Commitment to the app, and learning to code, is not intimidating.
- Gamification will increase engagement, especially for users who are new to computer science
- Content will be an easy, and not intimidating to new learners

Quicode grants its users the freedom to invest as much time as they want into learning, with the flexibility of learning anywhere at any time. Furthermore, we aim to empower individuals to take control of their own learning, and mitigate the friction commonly associated with the first few steps into the world of computer science.

1.2.1 Use Cases

Anticipated use cases fall into three categories:

1. Users who have some limited exposure to programming (i.e. through a prior class) and want to maintain and build upon previous knowledge.
2. Users completely new to computer science, who want a gradual introduction to computer science without a rigorous, high-commitment curriculum.
    a. This could be especially useful for middle school aged kids.
3. Users actively enrolled in an introductory programming course who want some sort of supplemental learning.

1.3 Description

This application aims to integrate incremental learning with the computer science fundamentals to make computer science more accessible to audiences of varying skill sets. It will be gamified through the use of a point tracking system, earned after finishing short lessons. These lessons will be categorized by module, with each module having an overarching topic it is meant to teach. Within each of these modules, topics will be broken down into subcategories, called blocks, and once more into questions 3 difficulty levels: easy, medium, and hard. Each of these sections will involve a series of short, interactive questions that encapsulate important computer science concepts. Users will have targeted questions alike to ones they recently missed from other lessons, and their progress will be reflected in the UI.

1.4 Limitations

While developing Quicode for mobile platforms provides benefits for users who would like to code on-the-go, it also limits the amount of features that the application can provide for the users' learning experience. For instance, providing a code editor as a learning sandbox is not feasible for a mobile application as compared to a computer-based web application. Additionally, due to the diversity of modern programming languages, keeping content consistent across programming languages gets more complicated for users as the modules dive deeper into the specifics of these concepts. Resultantly, because most languages share the core foundations of computer science, the initial release of Quicode focuses on these basics and fundamentals (i.e. iteration, sequences, conditionals, data types, etc.). Finally, targeting this application towards beginning users as a way to practice their computer science skills limits the application's target audience. In the future, we hope to extend this application to be a comprehensive platform for computer science such that anyone, from a beginner or a computer science major, could benefit from using Quicode.

1.5 Differentiation from Similar Applications

There are many pre existing applications and platforms that focus on computer science education, such as CodeAcademy, FreeCodeCamp, and Khan Academy. Quicode stands apart from these programs by providing a hands-on learning experience through the use of short questions, gamification, and decreasing the amount of time needed to learn. These pre existing platforms tend to involve long articles with detailed descriptions, with less of an emphasis on practice questions. Similarly Quicode is accessible on the go, unlike desktop applications such as Scratch: it allows users to quickly brush up on their coding skills without having to devote time to sit down in front of a computer and run code, or read through long paragraphs of content. Focusing first on basics, fundamentals, and pseudocode before going onto language-specific examples, Quicode allows users to develop a foundation for their computer science intuition. This will then allow them to later learn multiple languages easily due to its interchangeable curriculum, rather than other applications which give users language-specific tracks and examples. In short, Quicode's accessibility to those with busy schedules, and prioritization of applying knowledge over reading content make it unique to any alternatives.

2 Requirements

2.1 Scope

2.1.1 Scope of Features

For the initial release, the project scope consists of three main categories: user login and authentication, module content, and user progress tracking.

When it comes to user login and authentication, Quicode requires users to provide credentials in order to create an account and store their information. These accounts are linked to the progress users make: modules they are completing, and the questions they need to answer.

The first module, called the Foundational CS Module, will contain blocks which consist of the following topics: data types and variables, operators, conditionals, arrays, and iteration. Each of these blocks will include short conceptual summaries in bite-sized lessons,interactive questions such as multiple choice, multi-select, and drag and drop, and language-specific examples, if applicable. Users will be able to choose any lessons they want to review, or start a new lesson, but must complete one module or block in order to move on to the following one.

User progress tracking will ensure that users complete any necessary lessons required for consequential lessons. This progress tracking will be associated with the user's account, and will be used to enforce the user's completion of one module or block before moving onto the next one.

2.1.2 Out of Scope (Future Releases)

For subsequent releases, granted more time for this project, the scope includes the following:

- Support for multiple coding languages
- Implementing a coding sandbox, for users to compile code and receive feedback
- Implementing incremental learning: questions building upon user's weak points (ML)
- Allow the user to interact with other users (send direct messages, share progress, etc.)

- Implementing notifications such as daily reminders, timers, and goal setting and tracking in order to improve gamification
- Give users access to more detailed and technical information about concepts rather than short overviews
- Provide support for other operating systems like Android
- Curated content with more sophisticated algorithms, potentially AI backed

2.2 Software Requirements

Quicode will be developed for use on iOS mobile devices. We chose iOS as the target because iOS offers better uniformity. The standardization, documentation, and simplicity of iOS will benefit our development. We will use iPhones with the latest iOS software for testing, and Apple's Xcode for development.
Discussed further in the following sections, our application design will involve a backend component for user authentication, question storage, and user progress tracking. We plan to use a mix of Amazon Web Services' S3 bucket, DynamoDB, and Apple's UserDefaults for these purposes.

3 Backend Design Description

This section outlines the requirements for the backend. Our backend will be implemented using Swift and AWS S3's storage solutions.

3.1 Login and Authentication

To login, users will have to use their AppleID. Quicode will leverage Apple's built-in authentication system to ensure the highest level of security. Once an account is created, the user's metadata is sent to be stored in DynamoDB. The information is also stored locally using UserDefaults.

3.2 Users and User Progress

To keep track of users, we will use S3 to store each user as a file within the bucket. On DynamoDB, the user id will be affiliated with a progress marker that will allow tracking user progress. Locally, this id will be saved through UserDefaults.

3.3 Modules, Blocks, Questions, and Content

Information regarding modules, blocks, questions, and all curriculum related content will be stored in an AWS S3 bucket. AWS S3 will provide a high-performance and cost-effective way to store and retrieve content. This choice allows us to efficiently manage and deliver curriculum-related content while ensuring its accessibility, durability, and scalability.

Similarly, requesting this information only when it is needed, we can mitigate the size of this application, and maintain high performance for Quicode.

4 Frontend Design Description

This section outlines the various requirements for the frontend, and the design for different pages the application will include.

4.1 Overview of Pages

4.1.1 Login and User Pages

These pages will be used to enable users to log into their account on the app, and view stats regarding their progress on each module.

4.1.2 Home Page

This page will be the first page users will see after they log in, displaying a list of modules they can complete lessons within. As a user progresses through modules, they will unlock more. In the future, as depicted in the diagram, there will be an option to complete a mini lesson, curated to the user's weaknesses.

4.1.3 Module Page

Within each module, there will be lesson blocks that fall under the overarching module.

4.1.4 Block Page

Each lesson block contains navigation to easy, medium, and hard questions that will test the user on the current topic. These blocks also contain an optional, brief overview of the current topic for users who are unfamiliar with the content.

4.1.5 Questions

Following the difficulty selection on the block page, the user will be faced with various types of questions about the current topic. These questions will be presented in one of the following categories: multiple choice, multi-select, or drag and drop. Users can submit their answer, and will be given an unlimited number of attempts to submit the correct answer.

5 Results

Source code can be found in our project submission, or within the following git repository.

5.1 Achieved Functionality

5.1.1 Diversified Curriculum

Curriculum was designed to target users with limited to no exposure to Computer Science. It involves multiple modules, topic blocks, and questions of varying difficulty.

5.1.2 User Progress Tracking

Users earn stars for completing modules, and curriculum is locked if the prerequisites are not met; progress tracking is stored on an AWS DynamoDB table that is organized by user ID.

5.1.3 Different Styles of Questions

Multi-select, multiple choice, drag and drop with varying difficulties are utilized; each question type is developed separately within our repository and applied to the question data files.

5.1.4 Question Validation

Incorrect answers are highlighted on next attempts in order for the users to learn from their mistakes.

5.1.5 Topic Overviews

Each module and block contains a brief overview of the topic to cater to users who are new to computer science.

5.1.6 Question of the Day

daily extra practice to help users remember previous topics they have completed.

5.1.7 User Profile

users can see account information, total progress, and log out of their account.

5.1.8 Integrated AWS S3 Storage for Curriculum

module and block names, content overviews, questions are saved here on the AWS S3 bucket and pulled at appropriate locations in the application. This allows for easy modification and scaling in the future.

5.1.9 Apple UserDefaults

Saves user information between sessions to keep users logged in and pick up where they left off.

5.1.10 Sign in with Apple

Streamlined user login and authentication.

5.2 Compliance with Initial Specifications

The following details the compliance of the final app with initial specifications. Ultimately, our team was able to comply with all initial requirements, though the scope of these changed a small amount through the development process. Additionally, we were able to achieve some stretch functionality. The following section refers to the specifications outlined in the *CSDS 293 Initial Project Proposal Quicode* document.

5.2.1 User Login/Authentication

- Initial specifications required users to be able to login, and have new accounts created on the backend. Furthermore, specifications required accounts to be affiliated with app module progress.
  - **All of this functionality was achieved.**

5.2.2 Modules Consisting of Computer Science Concepts

- Initial specifications required development of a Foundational CS module, containing learning blocks: data types and variables, operators, boolean expressions, conditionals, sequences, iteration and binary numbers.
  - **All of this functionality was achieved.**
  - Furthermore, additional modules were created that delve into specific programming languages.
- Initial specifications required short conceptual summaries for each module, pseudocode examples and language-specific examples, and multiple interactive question types. Specifications called for ability to review previous lessons.
  - **All functionality was achieved.**

5.2.3 User progress tracking

- Initial specifications required a point system to gamify the application. Furthermore, specifications called for module progress tracking, where modules are locked until prerequisite modules are completed.
  - **All of this functionality was achieved.**
  - We used a star system rather than a point system. Furthermore, not just module progress was tracked. Block and sub-block (difficulty) data was tracked, locking appropriate sections, and assigning stars as necessary

5.3 Modified Specifications

While the vast majority of the scope of this application has stayed the same, we have added some technologies and changed the scope of some features. Namely, we:

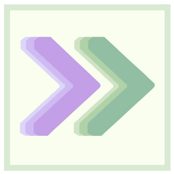
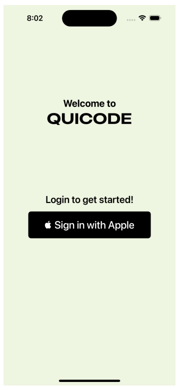
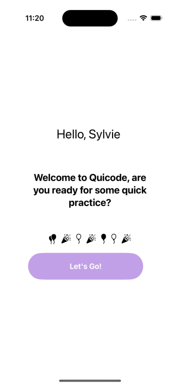
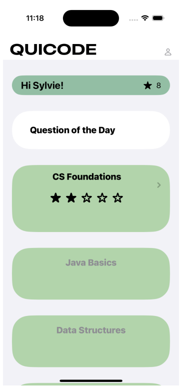
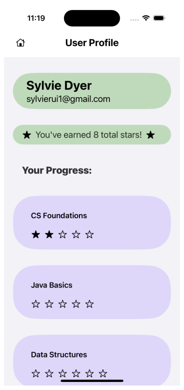
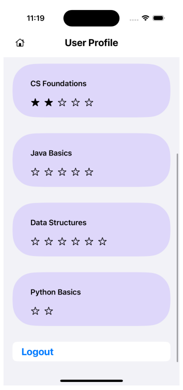
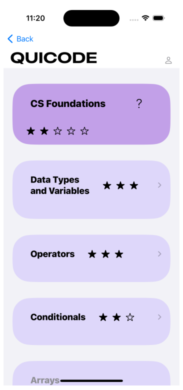
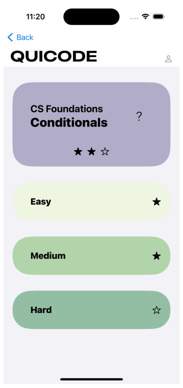
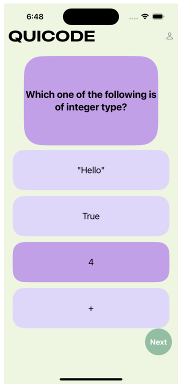
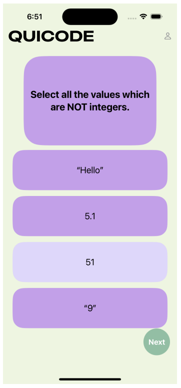
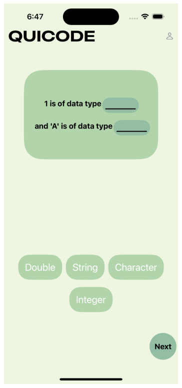
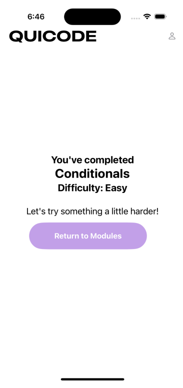
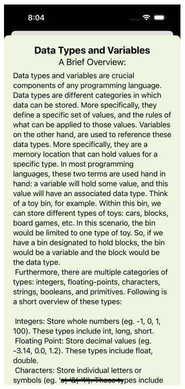
- Added use of DynamoDB for user progress tracking, due to ease of data manipulation when compared to S3
- Changed from using Apple's CoreData to Apple's UserDefaults, due to technical difficulties in accessing user data after the first login
- Modified curriculum and associated topics based on determined need and further analysis on content
- Implemented a question of the day. A similar feature was considered a stretch goal.
- Added user progress information in User view

Please note, this may cause discrepancies between this document and previous versions.

5.4 Services and Packages

Services Used: AWS S3, AWS DynamoDB, Apple Authentication, Apple UserDefaults
Packages Used: [Soto](), [WrappingHStack]()

5.5 Visual Overview

Application Icon:					
 8:02 Welcome to QUICODE Login to get started! Sign in with Apple	 11:20 Hello, Sylvie Welcome to Quicode, are you ready for some quick practice? Let's Go!	 11:18 QUICODE Hi Sylvie! ★ 8 Question of the Day CS Foundations Java Basics Data Structures	 11:19 User Profile Sylvie Dyer sylvierui1@gmail.com ★ You've earned 8 total stars! ★ Your Progress: CS Foundations Java Basics Data Structures	 11:19 User Profile CS Foundations Java Basics Data Structures Python Basics Logout	 11:20 Back QUICODE CS Foundations ? Data Types and Variables Operators Conditionals Arrays
Login	StartUp	Home Page	User (1)	User (2)	Module Page
 11:20 Back QUICODE CS Foundations Conditionals ? Easy Medium Hard	 6:48 QUICODE Which one of the following is of integer type? "Hello" True 4 + Next	 6:51 QUICODE Select all the values which are NOT integers. "Hello" 5.1 51 "9" Next	 6:47 QUICODE 1 is of data type and 'A' is of data type Double String Character Integer Next	 6:46 QUICODE You've completed Conditionals Difficulty: Easy Let's try something a little harder! Return to Modules	
Block Page	Multiple Choice Question	Multi-Select Question	Drag and Drop Question	Block Completion Page	Brief Overview Pop-Up

8:04

Data Types and Variables
A Brief Overview:
Data types and variables are crucial components of any programming language. Data types are different categories in which data can be stored. More specifically, they define a specific set of values, and the rules of what can be applied to those values. Variables on the other hand, are used to reference these data types. More specifically, they are a memory location that can hold values for a specific type. In most programming languages, these two terms are used hand in hand: a variable will hold some value, and this value will have an associated data type. Think of a toy bin, for example. Within this bin, we can store different types of toys: cars, blocks, board games, etc. In this scenario, the bin would be limited to one type of toy. So, if we have a bin designated to hold blocks, the bin would be a variable and the block would be the data type.
Furthermore, there are multiple categories of types: integers, floating-points, characters, strings, booleans, and primitives. Following is a short overview of these types:

Integers: Store whole numbers (eg. -1, 0, 1, 100). These types include int, long, short.
Floating Point: Store decimal values (eg. -3.14, 0.0, 1.2). These types include float, double.
Characters: Store individual letters or symbols (eg. '~~a', '&', '1'). These ty~~pes include