COMPSCI-320 Fall 2024 Section 2

CodeQuest

Group 2 Members: Mion Fujie, Matthew En, Liz Halpern, Zaida Le
Manager: Anushka

Summary:

CodeQuest is a gamified technical interview preparation web application that combines vetted foundational content with enriching crowdsourced interactive programming exercises.

The curriculum for the platform will be composed of a series of topics relevant to solving technical interview questions, each containing three modules:

- **Module 1:** Foundational knowledge of the topic provided through an embedded video and multiple choice questions.
- **Module 2:** A leetcode-style interactive programming exercise in python that must be completed before the user can move on to any subsequent topics.
- **Module 3:** Crowdsourced supplementary interactive exercises supporting Java, Python, JavaScript, TypeScript, Ruby, and C.

Tech Stack:

Frontend:	React, HTML/CSS/JS
Backend:	Node.js, Express
Database:	DynamoDB/Undecided
AWS:	Lambda, API Gateway, Cognito
Code Execution:	Judge0

Use Cases

Creating an account

- *Description:*
    - Users are required to register an account with CodeQuest to be able to access the application. Data will be associated with the account for the application.
- *Goal:*
    - User creates an account with CodeQuest.
- *Actors:*
    - Any user.
- *Preconditions:*
    - User opens CodeQuest.

    - User goes to sign up.
    - The user must have an email, Google, Discord, or Github account to sign up.
    - User enters information.
- *Postconditions:*
    - User now has an account with CodeQuest.
    - The user will be asked to verify their account through their email, Google, Discord, or Github accounts.
- *Exceptions:*
    - Account already exists.
- *Trigger:*
    - User signs up and creates an account by pressing the create account button with all required information.
- *Main Success Scenario:*
    - The user created an account and verified accounts though their respective Google, Discord, email, or Github account.
- *Error Scenario:*
    - Application communicates error to user that they:
        - Fill in missing required information.
        - Account already associated with email, Google, Discord, or Github account.
        - Account already exists.

Logging into the App

- *Description:*
    - Users authenticate their account so CodeQuest can verify their identity. This process prevents users from using an email, Discord, or Google account that does not belong to them.
- *Goal:*
    - Have a user authenticate their account.
- *Actors:*
    - Any user.
- *Preconditions:*
    - User created an account with CodeQuest
    - Users can use their Google, Discord, respective email, or Github account to login.
- *Postconditions:*
    - User selects a button that allows authentication.
    - User can now access their account and CodeQuest
- *Exceptions:*
    - Account was not verified.
- *Trigger:*
    - The user enters their Google, Discord, or Github account to authenticate created accounts.
    - User enters their email and password to sign in.
- *Main Success Scenario:*

    - The user now has access to their account and CodeQuest with their associated email, google, discord, or github account.
- *Error Scenarios:*
    - Application communicates to user
        - Incorrect password
        - Other sign in methods did not authenticate

Viewing Curriculum as a Graph

- *Description:*
    - Once the user opens the app, they will be directly brought to an overview of the progress graph, soon after they will then be brought to their current position on the graph, which would be the last place they left off in the course.
- *Goal:*
    - Create a visual representation of the progress that the user has made thus far in the course.
- *Actors:*
    - Any user.
- *Preconditions:*
    - Must log-in.
- *Postconditions:*
    - User can then click the continue/start lesson button to jump into the lesson.
- *Exceptions:*
    - Exceptions are not likely for this use case.
- *Trigger:*
    - User logs in.
- *Main Success Scenario:*
    - Brings the user to the current position on the graph represented by where they are in the course.
- *Error Scenarios:*
    - User is brought to the wrong position on the graph.

Consume Non - Interactive DSA Educational Content (Module 1)

- *Description:*
    - Module 1 will be the first required part of every lesson. Users will consume external content related to the lesson.
- *Goal:*
    - Educate users on the current lesson’s content using external resources.
- *Actors:*
    - Any user.
- *Preconditions:*
    - Unlock the current lesson by completing the previous.
- *Postconditions:*
    - User is then prompted to answer the set of questions following the content.

- *Exceptions:*
    - User is locked out of the lesson.
- *Trigger:*
    - User starts a new lesson.
- *Main Success Scenario:*
    - Content is completed and the user is prompted to the question section.
- *Error Scenarios:*
    - User completes the lesson beforehand but the next lesson doesn't become unlocked.

Non-Interactive Multiple Choice Questions pertaining to lesson content (Module 1)

- *Description:*
    - Module 1 will be the first required part of every lesson. Users will complete questions pertaining to content learned in the first part of module one.
- *Goal:*
    - Check the user's knowledge of content learned in the non-interactive DSA educational content section.
- *Actors:*
    - Any user.
- *Preconditions:*
    - Unlock the current lesson by completing the previous.
    - Users do not have to complete the video to start on the questions.
- *Postconditions:*
    - User receives feedback on the questions, must complete all questions correctly to move on
    - If a question is wrong, user is prompted back to redo all the questions
    - After 5 tries, the user is able to view the solution, but they will not be able to submit it
- *Trigger:*
    - User starts a new lesson.
- *Exceptions:*
    - User is locked out of the lesson.
- *Main Success Scenario:*
    - Content is completed correctly and the user is prompted to Module 2
- *Error Scenarios:*
    - User answers correctly but the application doesn't let the user move on to the next module.

Daily practice reminder through email

- *Description:*
    - If a user decides to allow notification from CodeQuest, they will receive emails notifying them to practice.
- *Goal:*
    - The user gets notified daily that they need to practice on CodeQuest.
- *Actors:*

- Any user that allows notification through email.

- *Preconditions:*
  - The user must have an account with CodeQuest.
  - The user must have signed up for emails by CodeQuest notification.
- *Postconditions:*
  - The user gets emails by CodeQuest that they need to practice.
- *Exceptions:*
  - The user has chosen to explicitly opt-out of notifications.
- *Trigger:*
  - Users when registering or through profile edits select the “allow to be notified by CodeQuest via email.”
- *Success Scenario:*
  - User receives emails from CodeQuest.
  - User uses email to open CodeQuest to practice.
- *Error Scenarios:*
  - User did not list email in account by using discord account.
    - App communicates, “please provide an email in your profile settings to receive email notifications.”

View Profile of Users

- *Description:*
  - Once the user opens the app, they will be able to access their own profile page, as well as other users on the app.
- *Goal:*
  - Display a visual summary of user information.
  - View questions and solutions submitted by other users, on their respective profile page.
  - View only exercise completed by the user themselves.
- *Actors:*
  - Any logged-in user
- *Preconditions:*
  - Must log-in.
  - Must click the “Profile” button to see their own profile or others
- *Postconditions:*
  - User can view alternative solutions to problems that they have completed in other users’ profiles
  - Users can view their own problems that they have solved
- *Exceptions:*
  - None for now
- *Trigger:*
  - User clicks on a profile (own or another user’s)
- *Main Success Scenario:*
  - User is able to view questions, solutions, and completed exercises from their own or

other peoples' profiles
- *Error Scenarios:*
  - User sees incorrect or incomplete data

View Module 3 Exercises
- *Description:*
  - Following a user's completion of the Module 2 exercise for a topic, they are able to optionally complete or contribute to crowdsourced exercises. This is how a user would select an exercise to perform.
- *Goal:*
  - The user can choose a crowdsourced exercise to complete.
- *Actors:*
  - Any user.
- *Preconditions:*
  - The user must have logged in.
  - The user must have completed the module 2 exercise for the parent topic.
- *Postconditions:*
  - The user has viewed the module 3 exercises.
- Exceptions:
  - None.
- *Trigger:*
  - The user decides to view module 3 exercises having completed the module 2 content for that topic.
- *Main Success Scenario*
  - The user sees the exercises they can perform.
- *Error Scenarios*
  - There is no user input in this operation. Unlikely to occur.

Submit a Solution to an Exercise
- *Description:*
  - This is the primary use case for those using Interactive Programming Exercises to prepare for interviews.
  - Given instructions for a programming exercise, provide code constituting a valid solution to that exercise.
- *Goal:*
  - User demonstrates knowledge of content.
- *Actors:*
  - Any user.
- *Preconditions:*
  - The user must have logged in.
- *Postconditions:*
  - The exercise is marked as completed in any page it is shown. Metadata for the solution

is available where applicable as needed.
    - If the exercise was a module 2 exercise, then if this topic was a prerequisite for any other topics, those topics are now accessible.
- *Exceptions:*
    - The provided code does not pass the test cases.
    - The provided code exceeds resource limitations.
- *Trigger:*
    - The user submits a solution to an interactive programming exercise.
- *Success Scenario:*
    - The user is brought back to the main module 3 page for that topic.
- *Error Scenarios:*
    - The code editor is re rendered in the browser, with information on the first failing test case along with a diff of the expected and actual outputs, or a stack trace.

Viewing solutions to an interactive programming exercise (Module 3)

- *Description:*
    - View master solution associated with an exercise.
- *Goal:*
    - User understands where they went wrong in their approach to the exercise.
- *Actors:*
    - Any user
- *Preconditions:*
    - The user is currently attempting an interactive programming exercise.
    - The user has decided to give up.
- *Postconditions:*
    - The user cannot attempt the interactive programming exercise until 24 hours have passed since the solution was viewed.
- *Exceptions:*
    - Very few ways for this one to go wrong, probably.
- *Trigger:*
    - The user has decided to give up.
- *Main Success Scenario:*
    - The master solution has been viewed.
- *Error Scenarios:*
    - Very few ways for this one to go wrong, probably.

Creating a new interactive programming exercise (Module 3)

- Description:
    - Community-provided interactive programming exercises ‘live’ under their topic in the graphical curriculum. This means that a problem is associated with a parent topic (Example: The classic “Is Palindrome?” problem would likely be associated with a Two Pointer Technique parent topic. We may also have another, identical, exercise under a String/Array parent topic).

- A Programming Exercise consists of its name, instructions, a collection of pairs of inputs and expected outputs, and a master solution.
- The solution is validated to produce the expected output for each given input through the same code execution workflow as submitting a solution.

- *Goal:*
  - Enable users who may or may not be the creator to complete a community provided programming exercise.
- *Actors:*
  - Any user.
- *Preconditions:*
  - The user must have logged in.
  - The user has completed all Module 1 and Module 2 tasks associated with the parent topic.
- *Postconditions:*
  - The exercise is available to users who have completed the prerequisite content.
  - The master solution is validated to pass the provided test cases.
- *Exceptions:*
  - The code provided does not pass all argument test cases.
  - Running the provided code exceeded the resource limits associated with the submission.
- *Trigger:*
  - User's innate free will.
- *Main Success Scenario:*
  - A programming exercise is created, ready for others to access in the Module 3 section of the parent topic.
- *Error Scenarios:*
  - The user is provided with an alert or flash error message containing any necessary information to fix the error (Example: Failing test cases should be highlighted red).

Adding a solution in a new language to an existing interactive programming exercise.

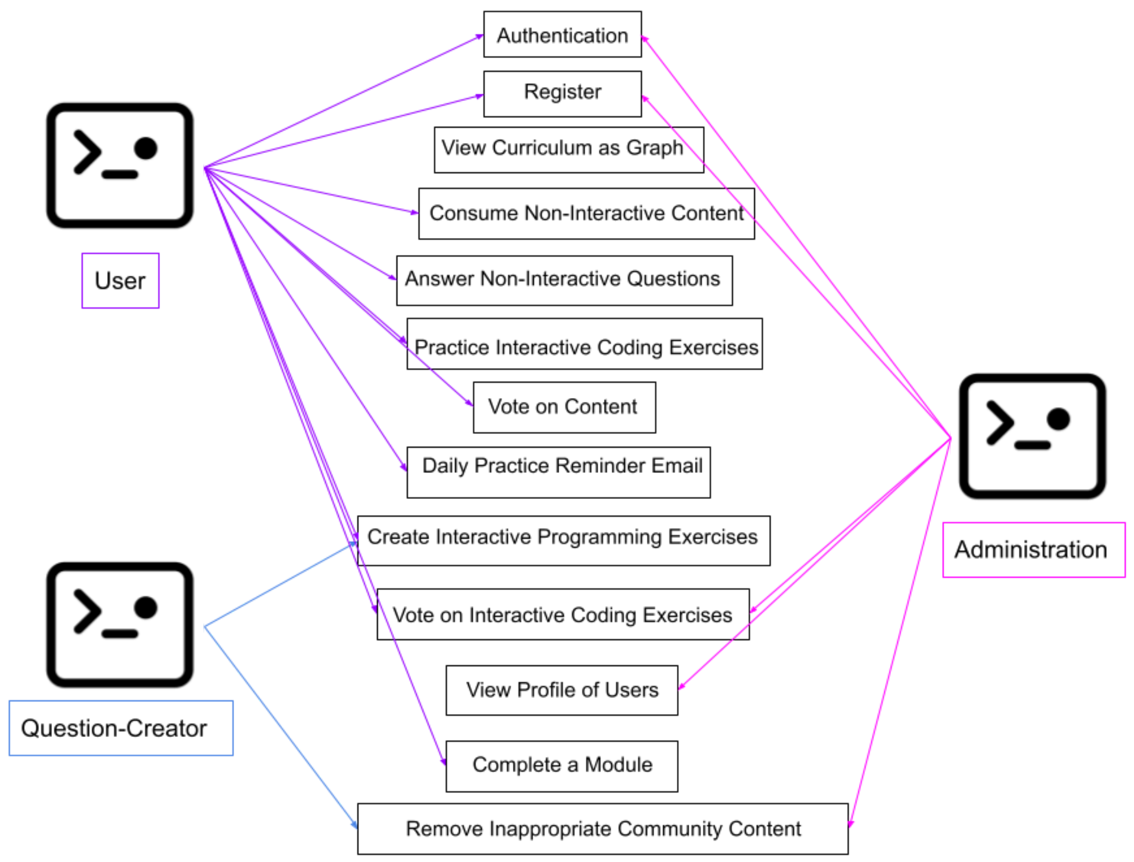
- *Description:*
  - Use the same workflow as creating a new interactive programming exercise, but adapt the instructions, test cases, and master solutions for a new language.
- *Goal:*
  - Enrich existing crowdsourced questions by adding support for a new language.
- *Actors:*
  - Any user.
- *Preconditions:*
  - The user must have logged in.
  - The user has completed the programming exercise they are adding new language support for.
  - The programming exercise does not already have support for the new language.
- *Postconditions:*
  - Users can now complete the original exercise in a new language.
- *Exceptions:*
  - The submission of the master solution exceeded resource use limits.

    - The submission of the master solution did not pass the provided test cases.
- *Trigger:*
    - User decides to add support for a new language to a problem they have completed.
- *Main Success Scenario:*
    - Other users are now able to complete the exercise in the new language.
- *Error Scenarios:*
    - Communicate errors to the user through the same mechanism as submitting a solution.

Removing a solution to an interactive programming exercise or removing an exercise.

- *Description:*
    - Given that we are accepting submissions and test cases from the same user to create exercises, some exercises are bound to be buggy/incorrect.
    - We fix this by providing a mechanism for a solution in a given language to be removed according to some TBD metric or schedule.
        - If there is no valid solution remaining to the interactive programming exercise, we fully remove the exercise.
- *Goal:*
    - Improve the quality of crowdsourced curriculum by removing buggy exercises.
- *Actors:*
    - Admin User who determined the solution to be faulty.
    - Creator of solution.
- *Preconditions:*
    - The user must have logged in.
- *Postconditions:*
    - Solutions can no longer be submitted to be compared against the faulty solution/tests
    - The solution is removed.
- *Exceptions:*
    - Very few ways for this one to go wrong, probably.
- *Trigger:*
    - A solution belonging to an interactive programming exercise has been determined to be faulty by some means.
- *Main Success Scenario:*
    - The solution is removed.
- *Error Scenarios:*
    - Very few ways for this one to go wrong, probably.

Use Case Summary Diagram


Authentication
Register
View Curriculum as Graph
Consume Non-Interactive Content
Answer Non-Interactive Questions
Practice Interactive Coding Exercises
Vote on Content
Daily Practice Reminder Email
Create Interactive Programming Exercises
Vote on Interactive Coding Exercises
View Profile of Users
Complete a Module
Remove Inappropriate Community Content
User
Question-Creator
Administration