TEACH TOK

Real Work Assignment

About this Assignment

In this assessment, you will design and implement a key feature of our "TeachTok" mobile application. We aim to gauge your expertise in creating user-friendly, robust, and efficient mobile applications, keeping in mind the expectations of today's dynamic user base.

Our primary focus is to evaluate your ability to use a cross-platform development framework to create applications according to a high-level specification, aligned to a provided high-fidelity design (colors, layout, margins, paddings, fonts), and following a Quality Bar.

Business Background

In the age of technology-driven education, platforms that merge entertainment with learning gain tremendous traction. TeachTok aims to be a pioneer in this space, offering users bite-sized educational content in an engaging format, reminiscent of TikTok.

- *Requirement*: Users should have a seamless experience when navigating and consuming content.
- *Requirement*: The app must match our design, closely following the TikTok “formula”.

At the heart of TeachTok's user experience is the "For You" section on the “Home” screen. This screen offers a seamless stream of fresh content at the fingertips of our users, encouraging continued exploration and engagement.

Problem Statement

Requirement: As a user, I want to get an endless stream of relevant multiple-choice questions, such that I can validate my knowledge in select areas.

UX Design: Figma prototype.

In scope:

- Showing content for the “For you” section of the “Home” screen,
- Displaying the Multiple Choice Questions (MCQs) in this section,
- Revealing the correct answer when the user taps one choice of an MCQ,
- Showing the content’s author name (e.g., “AP US History”), playlist, avatar, and content description,
- Browsing through content in an infinite scroll fashion (like TikTok),
- Measuring the time the user spent in the up using a countdown timer at the top left.

Not in scope:

- Persisting state changes (e.g., the selected choice for MCQs) to an API,
- Clicking Like, Comment, Share, and Bookmark buttons (just show them as static icons),
- Clicking into User Profile, Search, or into the Playlist (just show as static icons/text),
- Any of the other sections of the app except Home (Discover, Activity, Bookmarks, Profile; just show some placeholder).

API endpoints:

- Retrieve MCQs: GET https://cross-platform.rp.devfactory.com/for_you
  Returns the next content item for the user’s For You section.
- Reveal answer - GET https://cross-platform.rp.devfactory.com/reveal?id=X
  Reveals the correct answer for an MCQ question with id = X.

Your Work

1. Create a new React Native application using TypeScript.
    - You can use React Native vanilla or an encompassing framework like Expo.
2. Implement the user requirements described above.
    - Follow the quality bar listed below.
3. Record a 3-5 minute video:
    - First, show the app features you have implemented based on the requirements above,
    - Then, explain the code and architecture.

Grading

Your submission will be evaluated based on the following quality bar.

Functional:

- **Completeness**: Evaluate the functionality of the final code and its alignment with the desired output. High-quality submissions will include code that fully meets the functional requirements and evidence that the solution works as designed, provided through a demo video.
- **Consistency**: Asses adherence to the given UI/UX design and the overall user experience consistency. High-quality submissions will include a UI that closely follows the provided design and ensures a smooth and consistent user experience (e.g., scrolling is smooth and uninterrupted, and users can scroll back to content loaded previously in the same session and see their answers).

Technical:

- **Technology**: Evaluate the adherence to the technical constraints (i.e., usage of React Native & TypeScript) and to the industry standards related to these technologies. High-quality submissions will use React Native and standard patterns for managing state, calling APIs, handling navigation, recovering from errors, etc.
- **Clean Code**: Gauge the code's cleanliness, efficiency, consistency, and adherence to best practices. High-quality submissions will respect clean coding standards, and SOLID/DRY principles, and will not include unused code and/or compilation errors/warnings.