CSCI 448 – Lab 05B
Friday, February 17, 2023
LAB IS DUE BY **Tuesday, February 28, 2023, 11:59 PM**!!

The Dev Team needs to finish the ability to cheat, but nobody truly agrees with the intent of the request – what does is a quiz game if you can get all the answers! You decide to tweak the ticket. If a user does decide to cheat, then they won't be able to score the point for the question.

## Step 0 – Track the Cheated State

In the View Model, first add to the `QuestionStatus` enum values for `CHEATED` and `ANSWER_CHEATED`. The first value corresponds to when they hit the show answer button and thus have cheated. The second value corresponds to when they go to answer after they previously cheated. We need to know when both things have occurred to trigger different events.

On the `QuizlerViewModel` class, add a public function called `cheated()` that sets the status of the current question to `CHEATED` (be sure to update this in the list of statuses and in the current question status state).

Add a second public function called `answeredCheated()` that sets the status of the current question to `ANSWERED_CHEATED`.

## Step 1 – Display the Answer

It's time to finish out `CheatScreen`.

First, the existing button: when it is clicked, call the cheated method on our view model. This trigger's a state change for our current question.

Add to the composable tree being displayed, but add UI Logic based on the current question status:

- `Column`
  - `Text – "Are you sure you want to cheat?"`
  - `Button – **onClick = cheated()**`
    - `Text – "Cheat!"`
  - **`if currentStatus == CHEATED`**
    - **`Text – current question answer`**

If the current question status has been set to cheated, then display another Text composable that displays the answer for the current question. Be sure to handle the possibility for multiple choice questions with four options.

## Step 2 – To Catch A Cheater

The `QuestionDisplay` composable emits the buttons corresponding to our answer choices. If the user cheated, the question status changes but we want to make sure the buttons are still enabled. For each of our buttons, set them to be enabled if the current question status is `UNANSWERED` or `CHEATED`.

For each button, the onClick event will become a little more complex:

if the current question status is `CHEATED`
    then call a soon to be defined method called `onCheatAnswer()`
otherwise if this button corresponds to the correct answer
    then call the existing `onCorrectAnswer()` method
otherwise
    call the existing `onWrongAnswer()` method

Flow the `onCheatAnswer` function up by adding it as a function object parameter to `QuestionDisplay`.

In `QuestionScreen`, when we call `QuestionDisplay` define the `onCheatAnswer` function. We will do two things within the corresponding lambda expression:

1. Call the View Model `answeredCheated()` method
2. Display a `Toast` stating `"Cheaters never prosper."`

Build, deploy, run, cheat, try to answer. No point should be awarded, the buttons should be disabled, and the Dev Team wins!

The Quizler app is now complete. Feel free to add more questions to your repository and see how well your friends and family can answer all the questions.

# Step XC – Polish the App

There are two different opportunities for extra credit as it pertains to Quizler:

## Part XC.I – Close A Loophole

Currently, a user can perform the following actions:

1. Press the cheat button
2. Cheat to show the answer
3. Go back to question screen
4. Answer the question
5. Go back to step 1 and cheat again

While this doesn't have any negative side effect on the game, it's a poor experience that the user can get stuck in. Close this loophole so if after Step 4 the user returns to the cheat screen for this same question, they cannot press the show answer button and then cannot attempt to answer a second time.

## Part XC.II – End The Game

Once all the questions in the game have been answered (whether correct, wrong, or cheated), then navigate to a third screen that displays "Game Over" and shows the results for all the questions. See the sample video on the resources page for an example flow.

## Step 3 – Deploy Your App & Submit

When Lab05 is fully complete, you will submit a video of your working app to Canvas. Demonstrate the following actions inside the app:

- Press the cheat button
- Press the back button to go back to the question screen
- Answer the question correctly
- Go to the next question
- Press the cheat button
- Show the answer
- Press the back button to go back to the question screen
- Answer the question correctly

If doing Extra Credit 1

- Press the cheat button
- Press the back button to go back to the question screen

If doing Extra Credit 2

- Answer all questions

Then stop the recording. Save it as webm format, name the video `<username>_L05.webm`, and upload this file to Canvas Lab05.

You and the team push the new features out and await to hear what the QA Team thinks of your little addition to the build.

LAB IS DUE BY **Tuesday, February 28, 2023, 11:59 PM**!!