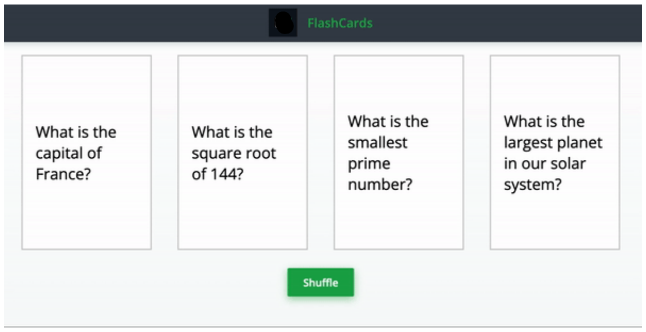
[redacted]

Question - 1
React (Typescript): Flashcards

Hide animation



What is the capital of France?

What is the square root of 144?

What is the smallest prime number?

What is the largest planet in our solar system?

Using Typescript and React, design a flashcard app that displays a series of flashcards with questions on the front and answers on the back. Certain core React functionalities are already implemented.

The application has two components: *FlashCardDeck.tsx* and *FlashCard.tsx* where the functionalities should be implemented.

The component must have the following functionalities:

- Display a series of flashcards with questions on the front and answers on the back.
- Clicking a flashcard should flip it to reveal the other side.
- Update the *isFlipped* constant to a state variable in *FlashCard/index.tsx*:
  - When *isFlipped* is true
    - *flipped* is appended to divs with the class name *flashcard-content.*
    - the answer is shown.
  - When *isFlipped* is false
    - '' is appended to divs with the class name *flashcard-content*.
    - the question is shown.
- Clicking the *Shuffle* button should reorder the flashcards.
  - Ensure the shuffled order of the flashcards is different from the original order present in *src/data/cards-data.ts*.
- All the types are defined under file *src/types/FlashCard.ts*

Create a type for *FlashCard* in *src/types/FlashCard.ts* with the following properties:

- *id,* a *number*
- *question*, a *string*
- *answer*, a *string*

The following *data-testid* attributes are required in the components for the tests to pass:

Attribute	Component
flashcard-deck	the main FlashCardApp component
flashcard-container-{card.id}	an individual FlashCard
flashcard-question-{card.id}	the question text in a FlashCard
flashcard-answer-{card.id}	the answer text in FlashCard

shuffle-button	a button to shuffle the flashcards

Note:
- Components have *data-testid* attributes for test cases and certain classes and ids for rendering purposes. They should not be changed.
- The files that should be modified by the candidate are *src/components/FlashCardDeck.tsx*, *src/components/FlashCard.tsx*, and *src/types/FlashCard.ts*.

Question - 2
React: Image Preview

Complete a React image preview application as shown below to pass all the unit tests. Certain core React functionalities are already implemented.

Show animation

The application has two components:
- The ImagePreview component that allows users to view and hide images by clicking on them.
- The HiddenImageDiv component that should be rendered when an image is to be hidden,

The image data is passed to the component as "images" to render in the component.

The application has the following functionalities:
- The ImagePreview component renders the following conditionally :
  - If the *visible* attribute is true, then the image with the *src* and *alt* attributes passed in the image data.
  - If the *visible* attribute is false, then the *HiddenImageDiv* component
- Clicking an image should hide it from the DOM, and the *HiddenImageDiv* should be visible in its place.
- Clicking the *HiddenImageDiv* component should render the original image again.
- On clicking the *ShowAll* button, all the images should be visible, and any *HiddenImageDiv* component should be hidden.
- On clicking the *HideAll* button, all the images should be hidden, and only the *HiddenImageDiv* components should be visible.

The following *data-testid* attributes are required in the components for the tests to pass:

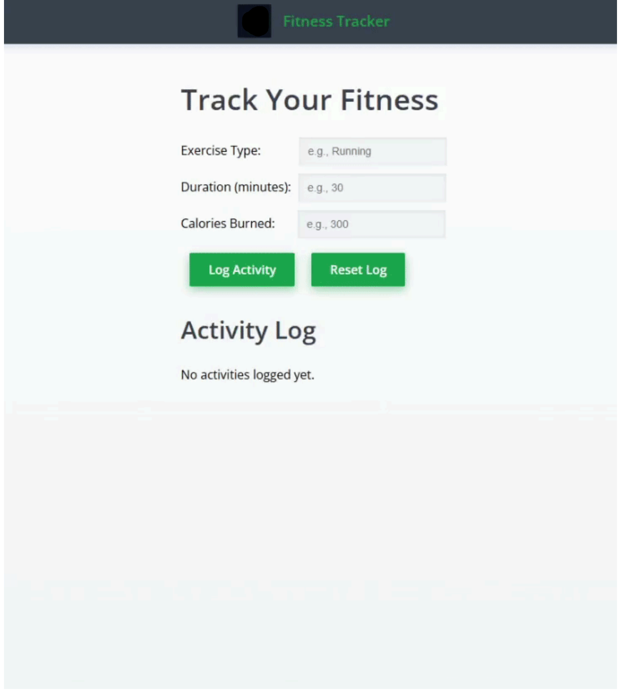
Attribute	Component
images-div	Images display div
show-all-btn	Show all images button
hide-all-btn	Hide all images button

Note:
- Components have *data-testid* attributes for test cases and certain classes and ids for rendering purposes. They should not be changed.
- The only file that should be modified by the candidate is the src/components/ImagePreview.js.

Question - 3
React: FitTrack Pro

In this application, the task is to create a fitness tracker that allows users to log their daily activities, including exercise type, duration, and calories burned. Edit the *FitnessTracker.js*, *LogForm.js,* and *LogList.js* within the project structure to ensure the application functions as described below:

Hide animation



Functionality Requirements:

1. Initial State of the project:
    - The exercise-type, duration, and calories-burned input fields should be empty initially.
    - The exercise-type input field should be of type text.
    - The duration and calories-burned input fields should be of type number.
    - The *Log Activity* button should not add any log until all fields have valid inputs.
2. Logging functionality:
    - The *Log Activity* button should add a log only when all fields have valid values.
    - The duration and calories burned should be non-zero and positive numbers.
    - On clicking the *Log Activity* button:
        - Add a new entry in the log table, displaying the *exercise-type*, *duration* (in minutes), and *calories-burned*.
        - Clear the input fields after successfully adding the entry.
    - On clicking the *Reset Log* button:
        - All logs are cleared and reset to an empty list.
        - The error message should also be cleared.
3. Log Display:
    - All log entries should appear within the application, under the *Activity Log* heading.
    - Each log entry, displays:
        - Exercise Type: the type of exercise logged
        - Duration: the time spent in minutes
        - Calories Burned: the calories burned during the activity
4. Error Handling:
    - Display the error message, if the user tries to submit invalid data, such as empty fields or non-positive numbers.
        - "Exercise type must not be empty." : If the exercise type input is empty.
        - "Duration must be a positive number.": If the duration input is empty or non-positive.
        - "Calories must be a positive number.": If the calories burned input is empty or non-positive.

Note:
The following data-testid attributes are required in the components for the test cases to pass, do not change/delete them:

Table of data-testid

data-testid	Description
input-exerciseType	Input box to enter the type of exercise
input-duration	Input box to enter the duration of the activity (in minutes)
input-caloriesBurned	Input box to enter the calories burned

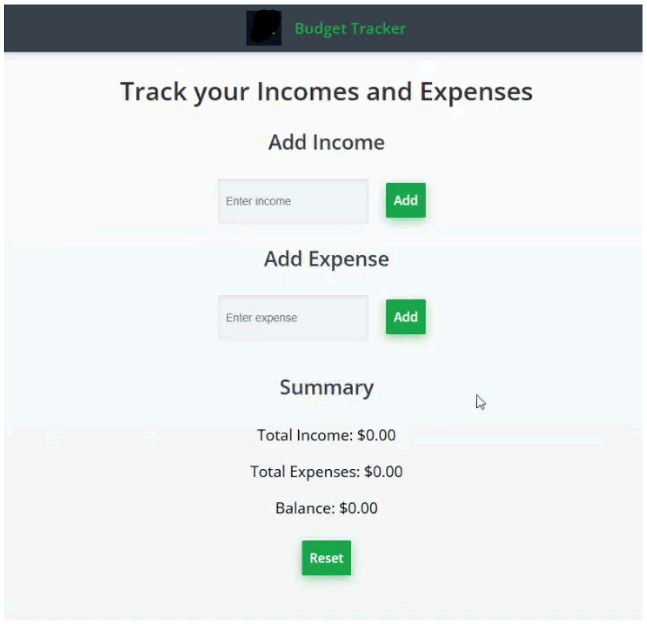
btn-logActivity	Button to log the activity
btn-resetLog	Button to reset all the activities
log-list	List of all the logs
log-entry	To display the user's logged activities
error-message	Error message for displaying the invalid inputs

Question - 4

React: Budget Balance Assistant

In this application, the task is to build a budget tracker that helps users manage their income and expenses effectively. Edit the *BudgetTracker.js*, *IncomeForm.js*, *ExpenseForm.js* and *Summary,js* components within the project structure to ensure the application functions as described below:

Hide animation



Functionality Requirements:

1. Initial State of the project:
   - The income and expense input fields should be empty and must have the attribute type as number.
   - Ensure that the total income, total expenses, and balance are initialized to 0.
2. Validation:
   - Positive Numbers Only:
     - The income and expense values must be positive. Negative or zero values should not be added.
   - Empty Input Handling:
     - No update to the total income, expense and balance, if non-positive income or expense is added.
3. Income and Expense Functionality:
   - Add Income: When the user enters a positive number in the income input box and clicks the "Add" button:
     - The value is added to the total income.
     - The balance is updated as total income - total expenses.
     - The income input box is cleared.
   - Add Expense: When the user enters a positive number in the expense input box and clicks the "Add" button:
     - The value is added to the total expenses.
     - The balance is updated as total income - total expenses.
     - The expense input box is cleared.
4. Reset Functionality:
   - Add a "Reset" button which resets the Total Income, Total Expenses, and Balance to 0.
   - Clicking the "Reset" button ensures the application returns to its initial state.

Note:
The following data-testid attributes are required in the components for the test cases to pass, do not change/delete them:

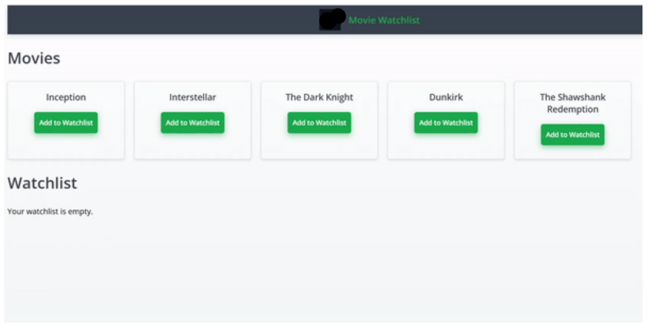
Table of data-testid

data-testid	Description
income-input	Input box to enter the income amount.
add-income-btn	Button to add the entered income.
expense-input	Input box to enter the expense amount.
add-expense-btn	Button to add the entered expense.
total-income	Text displaying the total income value.
total-expenses	Text displaying the total expenses value.
balance	Text displaying the current balance (income-expense).
reset-btn	Button to reset the income, expense, and balance.

Question - 5
React: Movie Watchlist

In this application, the task is to manage a user's movie watchlist. Edit the *Watchlist.js*, *WatchlistApp.js*, and *MovieCard.js* components within the project structure to ensure the application functions as described below:

Hide animation



Functionality Requirements:

1. Initial State of the project:
    - All movies should be displayed as cards in the "Movies" section, each with an "Add to Watchlist" button. The MovieCard component is used to generate a card for movie.
    - The watchlist section should be empty initially, displaying a message: "Your watchlist is empty."
    - The "Add to Watchlist" button should be enabled for all movies by default.
2. Watchlist Addition Functionality:
    - When the "Add to Watchlist" button is clicked on a movie, the movie should be added to the watchlist.
    - The "Add to Watchlist" button should be disabled for movies already in the watchlist, and it's text should be changed to "Added".
3. Watchlist Display Functionality:
    - The watchlist should display movies as cards, similar to the main movie section, with each card containing a "Remove" button, using the same MovieCard component.
    - Clicking the "Remove" button should remove the movie from the watchlist, and the corresponding "Add to Watchlist" button should be re-enabled.
4. Clear Watchlist Functionality:
    - A "Clear Watchlist" button should be displayed in the watchlist section if there are movies in the watchlist.
    - Clicking the "Clear Watchlist" button should remove all movies from the watchlist and display the message: "Your watchlist is empty."
5. Movie Count Display:
    - The total number of movies in the watchlist should be displayed in a badge/icon near the watchlist title. For example: "Watchlist (2)".

Note:
The following data-testid attributes are required in the components for the test cases to pass, do not change/delete them:

Table of data-testid

data-testid	Description
movie-card-{id}	Movie card displaying the movie alongwith Remove button.
movie-title-{id}	The title of the movie displayed in either the movie list or watchlist.
add-btn-{id}	Button to add the movie with the specified ID to the watchlist.
remove-btn-{id}	Button to remove the movie with the specified ID from the watchlist.
watchlist-container	Displays all the movies in the watchlist alongwith its count.
watchlist-empty	Message indicating that the watchlist is empty.
movie-container	Displays all the movies in the watchlist.
clear-watchlist-btn	Button to clear all movies from the watchlist.