Junior Developer Coding Test

Section 1: Algorithmic Challenges (60 Minutes)

1. FizzBuzz (Basic Logic and Loops)

Write a function that prints the numbers from 1 to 100. However, for multiples of three print "Fizz" instead of the number and for the multiples of five print "Buzz". For numbers which are multiples of both three and five print "FizzBuzz".

Example Output:

1

2

Fizz

4

Buzz

Fizz

7

8

Fizz

Buzz

11

Fizz

13

14

FizzBuzz

2. Palindrome Checker (String Manipulation)

Write a function that checks if a given word is a palindrome. A palindrome is a word that reads the same backward as forward (e.g., "radar", "level"). The function should ignore case and spaces.

Function Signature:

```
def is_palindrome(word: str) -> bool:
    pass
```

Example:

```
print(is_palindrome("Radar"))  # True
print(is_palindrome("Hello"))  # False
print(is_palindrome("A man a plan a canal Panama"))  # True
```

3. Sum of Unique Elements (Lists and Sets)

Given a list of integers, write a function that returns the sum of all unique elements (i.e., elements that appear only once in the list).

Function Signature:

```
def sum_of_unique_elements(nums: list[int]) -> int:
    pass
```

Example:

```
print(sum_of_unique_elements([1, 2, 3, 2, 4]))  # 8
print(sum_of_unique_elements([5, 5, 5, 5, 5]))  # 0
```

Section 2: Mini Project (90 Minutes)

Build a Simple To-Do List Application (Console Based)

Requirements:

- The application should allow a user to add, view, delete, and mark tasks as complete.
- Each task should have a title and a status (pending or completed).
- The tasks should be stored in a list, and the application should keep running until the user chooses to exit.
- Include proper error handling for invalid inputs.

Example Console Interaction:

Welcome to the To-Do List App!

1. Add a Task
2. View All Tasks
3. Mark Task as Complete
4. Delete a Task
5. Exit

Choose an option: 1

Enter the task title: Buy groceries

Choose an option: 2

Tasks:

1. [ ] Buy groceries

Choose an option: 3

Enter task number to mark as complete: 1

Task marked as complete!

Choose an option: 2

Tasks:

1. [X] Buy groceries

Choose an option: 5

Goodbye!

Section 3: Code Quality and Best Practices (30 Minutes)

- Use meaningful variable names.
- Include comments where necessary.
- Ensure your code is readable and follows standard formatting conventions.
- Optimize for efficiency where possible.

Submission Guidelines:

- Submit your code as a single .zip file or a public GitHub repository.
- Include a README.md file that briefly describes your approach and any assumptions made.
- Ensure the code is tested and free from syntax errors.