Create a test plan outline

To complete this exercise create a bulleted list of test descriptions for tests you would run if you were testing this application.

Note: I'm listing all these tests as manual checks. If I were to automate them, they would not all be E2E. These would be a mix of E2E, component, and unit tests.

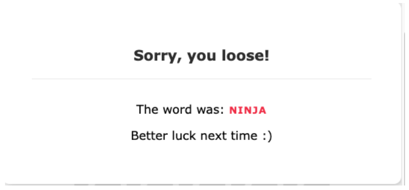
- **Navigation to App:** Ensure routing to Wordle app (https://localhost:3000) produces the initial new game state (Fresh Wordle board and alphabet) and all expected elements are visible.
- **Browser/Platform Compatibility:** For each supported browser or platform, verify the app renders and functions. (Note: We should make separate test run executions for each supported browser and platform, with each running all relevant tests)
- **Viewport Responsiveness:** Verify changing the window size does not shift or cut out UI elements
- **Header:** Verify the "Wordle" Header renders
- **Query String:** Verify you can set the game up with a specific 5-letter word in the query string in the URL. Example: http://localhost:3000/?test=ninja
- **Query String boundaries:** Verify the game does not accept the Query String to be anything but 5 alphabetic characters
- **Wordle Board Tests**
    - **Board display:** Verify the board renders in a 5x6 grid of squares, initially empty
    - **Input Boundaries:** Make sure Wordle only accepts 5 characters each attempt, and once you reach 5, it doesn't overwrite any characters.
    - **Enter Boundary:** Very hitting enter only accepts a guess after there are 5 characters typed
    - **Incorrect Characters:** Verify only the 26 alphabet characters are accepted as input
    - **Don't allow reused guesses:** Verify that if the user types a guess that is already on the board, the app does not accept it when hitting enter.
    - **Tile Color:** Verify the color of each tile state is as expected:
        - Before a guess is submitted, tiles are white with black text when letters are typed by user
        - After a guess is submitted:
            - Tiles are gray with white text if the letters is not in wordle
            - Tiles are blue with white text if the letters is in the word, but not in the right position
            - Tiles are green with white text if the letter is correctly positioned.
    - **Delete characters:** Verify you can delete types characters with the delete key
- **Alphabet Tests**
    - **Letters display:** Verify the alphabet is displayed below the board and renders boxes for all 26 letters. They should initially be all light gray.

    - **Used Letters:** Verify when a letter is used on a guess, it is either grayed out if it is not in the word, or colored blue when it is in the word.
- **Game State Tests**
    - **Guess Animation:** Every guess should show an animation once accepted.
    - **Tile changes:** Every guess should appropriately switch the tiles and alphabets to the correct colors, depending on if the letters are incorrect, correct, or in the wrong position.
    - **Word Boundary:** Verify if you use a query string, you can't give it a word that is shorter or longer than 5 characters
    - **Guesses:** Verify the game only ends with a win/lose modal after 6 guesses
    - **Refresh Page:** Verify that if you refresh the page while any progress is made in the game, it resets the state of the game to its initial state (new game).
- **Happy Path Win:** Verify you can win the game. Set the board to a word like "drink" using a query string and guess. NOTE: We could parametrize this test so we could win with 1-6 guesses to ensure you can win at any particular point in the game, if we wanted to test that logic, but you could probably write a lower level test for that logic.
- **Happy Path Lose:** Verify that you can lose the game by guessing incorrectly 6 times, and a modal appears with a header that says: "Sorry, you Lose!" and underneath it says: "The word was: <word>, Better luck next time :)"

2. Write a manual test

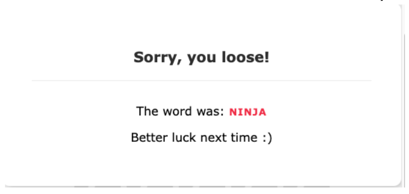
Further below, I've written a more full test case, but in the test I automate, I'm going to scope the test to assert specifically that a user can lose the game. I won't check the states of tiles or the alphabet grid. Here is the test case:

	Action	**Data**	**Expected Result**
1	Open browser under test and navigate to http://localhost:3000 with a query string "/?test=ninja		Wordle App renders with a new game state
2	Type a guess containing five letters not in the word. Hit the enter key.	Type: "lbcde"	The guess is accepted in the 1st row on the wordle board and the tiles are filled in
3	Type in a second guess with at least one correct character in the grid, but in the wrong position, and hit the enter key.	Type: "zxuwb"	The guess is accepted and the 2nd row tiles are filled in
4	Type in a third guess with at least one correct character in the grid in the right position, and hit the enter key.	Type: "cccyd"	The guess is accepted and the 3rd row tiles are filled in
5	Type in a fourth guess and put the	Type: "hkcjb"	The guess is accepted and the

	incorrectly positioned character in the correct position. Hit the enter key.		4th row tiles are filled in
6	Type in a 5th guess and ensure it is incorrect. Hit the enter key.	Type: “ccccc”	• The guess is accepted and the 5th row tiles are filled in
7	Type in a 6th guess and ensure it is incorrect. Hit the enter key.	Type: “bbbbb”	• The guess is accepted and the 6th row tiles are filled in • The game is lost and a modal displays saying: “Sorry, you lose!” in the header • Under the header, the modal says: “The word was: Ninja Better luck next time :)”  Sorry, you loose! The word was: NINJA Better luck next time :)

For a User Flow for manual testing or a single big user flow E2E test I would probably write a test case like this:

	HAPPY PATH LOSE USER FLOW		
	Action	**Data**	**Expected Result**
1	Open browser under test and navigate to http://localhost:3000 with a query string “/?test=ninja		Wordle App renders with the following elements: • Wordle Header at the top • 5x6 square board • Alphabet grid below the board, with all 26 characters
2	Type a guess containing five letters not in the word.	Type: “lbcde”	The letters show up in the 1st row on the wordle board.
3	Hit the enter key.		• The guess is accepted and turn gray on the board and alphabet grid

4	Type in a second guess with at least one correct character in the grid, but in the wrong position, and hit the enter key.	Type: “zxjwb”	● The guess is accepted and new incorrect letters turn gray on the board and alphabet grid ● The correct letter turns blue on the board and alphabet grid
5	Type in a third guess with at least one correct character in the grid in the right position, and hit the enter key.	Type: “cccya”	● The guess is accepted and new incorrect letters turn gray for being incorrect ● The correct letter turns green on the board and alphabet grid
6	Type in a fourth guess and put the incorrectly positioned character in the correct position. Hit the enter key.	Type: “hkcjb”	● The guess is accepted and incorrect letters turn gray for being incorrect ● The letter placed in a correct position turns green on the board and alphabet grid
7	Type in a 5th guess and ensure it is incorrect. Hit the enter key.	Type: “ccccc”	● The guess is accepted and new incorrect letters turn gray for being incorrect
8	Type in a 6th guess and ensure it is incorrect. Hit the enter key.	Type: “bbbbb”	● The guess is accepted ● The game is lost and a modal displays saying: “Sorry, you lose!” in the header ● Under the header, the modal says: “The word was: Ninja Better luck next time :)”  **Sorry, you loose!** The word was: NINJA Better luck next time :)

4. Create an automation test for winning the game

Since my goal is to just test that a user can win the game, I'll make it fast and target to a single guess. If I wanted to go further and test the tile colors more thoroughly, I could in another test case.

Bugs and observations found

- Bug: Typing random characters rapidly can make the game screen turn white (hard to repro)
- Bug: The losing screen has a typo: "Sorry, you Loose!" It should say "Sorry, you lose!"
- Bug: You can make an unsolveable wordle by passing in a word that isn't exactly 5 characters: /**?test=a** or /**?test=coolness**
- I found a few spelling errors in the instructions (bullted list) (manaual test)
- CSS-only elements and lack of unique identifiers made automation locator strategy less effective.