Testing is a crucial part of the software development process, ensuring that the application behaves as expected and meets the requirements. Here’s a brief overview of how you can approach testing for the

**To-Do List Application**:

**1. Types of Testing**

**1.1 Unit Testing**

* **Purpose**: Test individual functions or components in isolation.
* **Tools**:
  + **Jest**: A JavaScript testing framework that can be used to write unit tests.
  + **Mocha**: Another popular testing framework.

**1.2 Integration Testing**

* **Purpose**: Test the interaction between different modules or components.
* **Focus**: Ensure that the UI correctly reflects the state of the application, especially after adding, editing, or deleting tasks.

**1.3 Functional Testing**

* **Purpose**: Verify that the application behaves as expected based on the requirements.
* **Approach**: Test the complete flow of the application, from adding a task to marking it as completed or deleting it.

**1.4 User Interface Testing**

* **Purpose**: Ensure that the UI is user-friendly and works correctly across different devices and browsers.
* **Tools**:
  + **Selenium**: A framework for automated testing of web applications.
  + **Cypress**: A modern testing framework for end-to-end testing.

**2. Test Cases**

Here are some example test cases for the To-Do List Application:

**2.1 Add Task**

* **Test Case**: Verify that a task can be added successfully.
* **Steps**:
  1. Enter a valid task name in the input field.
  2. Click the "Add" button.
  3. Check if the task appears in the task list.

**2.2 Edit Task**

* **Test Case**: Verify that an existing task can be edited.
* **Steps**:
  1. Add a task to the list.
  2. Click the edit button next to the task.
  3. Change the task name and confirm.
  4. Check if the task name is updated in the task list.

**2.3 Delete Task**

* **Test Case**: Verify that a task can be deleted successfully.
* **Steps**:
  1. Add a task to the list.
  2. Click the delete button next to the task.
  3. Check if the task is removed from the task list.

**2.4 Toggle Complete**

* **Test Case**: Verify that the completion status of a task can be toggled.
* **Steps**:
  1. Add a task to the list.
  2. Click the complete button next to the task.
  3. Check if the task appears crossed out.
  4. Click the complete button again.
  5. Check if the task is no longer crossed out.

**2.5 Local Storage**

* **Test Case**: Verify that tasks persist in local storage after a page refresh.
* **Steps**:
  1. Add a task to the list.
  2. Refresh the page.
  3. Check if the task is still present in the task list.

**3. Conclusion**

By conducting thorough testing, you can ensure the reliability and quality of the To-Do List Application. Make sure to document your tests and their results to keep track of any issues that arise and verify that they have been resolved.

**Unit Test:**

To perform unit testing on the **To-Do List Application**, we'll write tests for the JavaScript functions using a testing framework like **Jest**. Below are the steps to set up Jest and write unit tests for the core functionalities of the application.

**Step 1: Set Up Jest**

1. **Initialize Your Project**: If you haven’t already, create a new directory for your project and navigate into it.
2. **Initialize npm**: Create a package.json file. (npm init –y)
3. **Install Jest**: Install Jest as a development dependency. (npm install --save-dev jest)
4. **Update package.json**: Add a test script in your package.json file to run Jest.( "scripts": {

"test": "jest"})

1. **Create a Directory for Tests**: Create a new directory for your test files.

**Step 2: Write Unit Tests**

Now, let's write unit tests for the key functions of your To-Do List application. Create a new file named app.test.js inside the \_\_tests\_\_ directory.

**Step 3: Run the Tests**

To run the tests, use the following command in your terminal. (npm test)