Unit Testing

Unit Testing Framework for Project

This section outlines a simple unit testing strategy and framework suggestion for the project, which primarily uses HTML, CSS, and JavaScript.

Why Unit Testing?

- Ensures individual components work as expected.
- Helps catch bugs early.
- Facilitates future code refactoring with confidence.
- Promotes modular and maintainable code.

Recommended Frameworks

Since the project is front-end focused, the following tools are recommended:

1. Jasmine A behavior-driven JavaScript testing framework.
2. Mocha A flexible JavaScript test framework running on Node.js.
3. Jest A zero-config testing framework developed by Facebook.
4. QUnit Used for testing jQuery-based code.
5. Karma A test runner that can be used with any framework.

Testable Components in Webside

You can create unit tests for the following parts of yproject:

- JavaScript functions (e.g., DOM manipulation, event listeners)
- Form validation logic
- Navigation toggles

- Sliders or dynamic UI components

Example (Using Jasmine)

Heres a simple example of a Jasmine test case:

```
function add(a, b) {
  return a + b;
}

describe('Addition function', function() {
  it('should return 5 for add(2, 3)', function() {
  expect(add(2, 3)).toBe(5);
  });
});
```

How to Set Up

- Download Jasmine standalone or use npm to install Jest/Mocha.

- Write test cases in a `tests/` directory.

- Use `npm test` or open `SpecRunner.html` depending on the framework.

- Integrate with browser or CI for auto testing.

References

Jasmine Official Site

Jest Documentation

Unit Testing

Mocha Framework

QUnit Guide