One-time setup

1. **Verify app scripts (Vite)**
   - In `package.json` make sure you have:

```
{
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "preview": "vite preview"
  }
}
```

Install Playwright

```
npm i -D @playwright/test

npx playwright install --with-deps chromium firefox
```

Create the test folder

```
mkdir -p tests
```

Add the config

Create `playwright.config.ts` at the project root with **your** content:

```
import { defineConfig, devices } from '@playwright/test';

export default defineConfig({
  testDir: './tests',
  use: {
    baseURL: 'http://localhost:5173',   // so
page.goto('/') works
    trace: 'on-first-retry',
  },
  projects: [
    { name: 'chromium', use: { ...devices['Desktop Chrome']
} },
    { name: 'firefox',  use: { ...devices['Desktop
Firefox'] } },
    // { name: 'webkit',   use: { ...devices['Desktop
Safari'] } },
  ],
  reporter: [['list'], ['html', { open: 'never' }]],
  webServer: {
    command: 'npm run dev -- --port 5173', // force the
port for reliability
    port: 5173,
    reuseExistingServer: !process.env.CI,
    timeout: 120_000,
  },
});
```

2. **Add the spec**

    - Create `tests/app.spec.ts` with **your** content:

```
import { test, expect } from '@playwright/test';

test.describe('Mantine + Router app', () => {
  test('navigation: Home ⇄ About', async ({ page }) => {
    await page.goto('/');
    await expect(page.getByRole('heading', { level: 1,
name: /home page/i })).toBeVisible();
    await page.getByRole('link', { name:
'About' }).click();
    await expect(page.getByRole('heading', { level: 1,
name: /about us/i })).toBeVisible();
  });

  test('login as Guest → see Dashboard → logout', async
({ page }) => {
    await page.goto('/');
    await expect(page.getByRole('link', { name: 'Dashboard'
})).toHaveCount(0);
    await page.getByRole('button', { name:
'Login' }).click();
    await expect(page.getByRole('link', { name: 'Dashboard'
})).toBeVisible();
    await page.getByRole('link', { name:
'Dashboard' }).click();
    await expect(page.getByRole('heading', { level: 1,
name: /dashboard/i })).toBeVisible();
    await page.getByRole('button', { name:
'Logout' }).click();
    await expect(page.getByRole('link', { name: 'Dashboard'
})).toHaveCount(0);
  });

  test('toggle theme (light ↔ dark)', async ({ page }) => {
    await page.goto('/');
    const html = page.locator('html');
    const before = await html.getAttribute('data-mantine-
color-scheme');
    await page.getByRole('button', { name: 'Toggle color
scheme' }).click();
    const after = await html.getAttribute('data-mantine-
color-scheme');
    expect(after).not.toBe(before);
  });
});
```

Run each time

Run the tests

```
npx playwright test
```

- Playwright will **start Vite** on port **5173**, open your app, and run the 3 tests on Chromium + Firefox.

See the results

```
npx playwright show-report
```

- Opens the HTML report with pass/fail, timings, and traces (on first retry).

Optional (nice to have)

8. **Watch the browser**

   ```
   npx playwright test --project=chromium --headed
   ```

Or use the interactive UI:

```
npx playwright test --ui
```

Add handy scripts

```
{

  "scripts": {
    "test:e2e": "playwright test",
    "test:e2e:ui": "playwright test --ui",
    "test:e2e:headed": "playwright test --project=chromium
--headed"
  }
}
```

Quick troubleshooting

- **“Cannot navigate to invalid URL /”** → Your config must have both:
  - `use.baseURL: 'http://localhost:5173'`
  - `webServer.port: 5173` and `webServer.command: 'npm run dev -- --port 5173'`
- **Port already in use** → Change **both** places to a free port (e.g., 5199):
  - `baseURL: 'http://localhost:5199'`

    - `webServer: { command: 'npm run dev -- --port 5199', port: 5199 }`

- **Want WebKit later** → `npx playwright install --with-deps webkit` and add it back to `projects`.

Using Playwright uI

Using the Playwright UI

1) Start the UI

```
npx playwright test —ui
```

This opens the Playwright runner window. With your config, it will also **start Vite** on port **5173** automatically.

2) Pick what to run

- **Projects (browsers):** In the left sidebar, you'll see **Chromium** and **Firefox**. Check/uncheck to include/exclude a browser.
- **Filter tests:** Use the search box at the top to filter by test name or file.
- **Run a single test:** Click the ▶ (play) icon next to a test.
- **Run a whole file/suite:** Click the ▶ next to the file or the suite ("Mantine + Router app").

3) Watch it execute

- A **browser window** opens while the test runs (headed mode in UI).
- The **center pane** shows live logs (steps, locators, assertions).
- The **right pane** shows errors/stack traces if something fails.

4) Rerun fast

- Click the ▶ **Run** button at the top to rerun all visible tests.
- Change code → the UI **auto-watches** your files; hit ▶ again to rerun.

5) Inspect failures (traces, screenshots, video)

Your config has `trace: 'on-first-retry'`. To always capture a trace (handy while debugging), change to:

```
// in playwright.config.ts
use: {
  baseURL: 'http://localhost:5173',
  trace: 'on',                // always record
  screenshot: 'only-on-failure',
  video: 'retain-on-failure',
}
```

Then in the UI, click **"View trace"** on a failed run to step through each action, see DOM snapshots, console logs, and network calls.

6) Debug step-by-step (optional)

- Start the UI **paused at the first step**:

  ```
  npx playwright test --ui --debug
  ```

- Or drop `await page.pause()` inside a test; it opens the inspector and waits for you.

7) Common "UI didn't open / server didn't start" fixes

Make sure your config has both:

```
use: { baseURL: 'http://localhost:5173' },

webServer: { command: 'npm run dev -- --port 5173', port:
5173 }
```

If port 5173 is busy, change **both** `baseURL` and `webServer.port` to a free port (e.g., 5199) and rerun:

```
use: { baseURL: 'http://localhost:5199' },

webServer: { command: 'npm run dev -- --port 5199', port:
5199 }
```

If the UI says "Waiting for web server…": make sure `npm run dev` works by itself.