| Steps | Test data | Expectation |
| --- | --- | --- |
| * Navigate to the URL of the application (When cookie dialog appears click 'accept all' button) | * url: <https://demo.clickdoc.de/cd-de/> | * Preparation; no expectations |

To automate the navigation and acceptance of the cookie dialog using WebdriverIO and TypeScript, follow the steps below:

1. Install the required packages:

npm install webdriverio typescript ts-node @types/node --save-dev

1. Create a wdio.conf.ts configuration file:

// test1.ts

import { chromium, Browser, Page, ElementHandle } from 'playwright';

describe('ClickDoc DE Demo', () => {

let browser: Browser;

let page: Page;

beforeAll(async () => {

browser = await chromium.launch();

page = await browser.newPage();

});

afterAll(async () => {

await browser.close();

});

it('should accept the cookie dialog', async () => {

await page.goto('https://demo.clickdoc.de/cd-de/');

const acceptCookieButton = await page.waitForSelector('button[aria-label="Alle Cookies akzeptieren"]');

await acceptCookieButton.click();

// Optionally, you can check if the page title is updated after accepting the cookies.

await expect(page).toHaveTitle('ClickDoc - Terminvereinbarung');

});

});

1. Create a test.ts file to write your test:

// test.ts

import { chromium, Browser, Page, ElementHandle } from 'playwright';

describe('ClickDoc DE Demo', () => {

let browser: Browser;

let page: Page;

beforeAll(async () => {

browser = await chromium.launch();

page = await browser.newPage();

});

afterAll(async () => {

await browser.close();

});

it('should accept the cookie dialog', async () => {

await page.goto('https://demo.clickdoc.de/cd-de/');

const acceptCookieButton = await page.waitForSelector('button[aria-label="Alle Cookies akzeptieren"]');

await acceptCookieButton.click();

// Optionally, you can check if the page title is updated after accepting the cookies.

await expect(page).toHaveTitle('ClickDoc - Terminvereinbarung');

});

});

1. Update your package.json script to run the test:

"scripts": {

"test": "ts-node --files test.ts"

}

1. Run the test:

npm test

This will navigate to the URL <https://demo.clickdoc.de/cd-de/> and click the "Alle Cookies akzeptieren" button to accept the cookie dialog.

| Steps | Test data | Expectation |
| --- | --- | --- |
| * Insert the search term in the ‘Subject area, Name of the doctor, Practice..’ and ‘City, PLZ or District’ input field and click ‘Finden’ Button | * search term: „Peter Wunderlich“ and „Testhausen“ | * Validate “Peter Wunderlich” physician view was displayed as first in the list of results * Validate physician name and address * Validate that he is online bookable for the month of April 2024 * Validate the color of ‘Find’ and ‘Appointment booking’ buttons in the page |

// test2.ts

import { chromium, Browser, Page, ElementHandle } from 'playwright';

describe('ClickDoc DE Demo', () => {

let browser: Browser;

let page: Page;

beforeAll(async () => {

browser = await chromium.launch();

page = await browser.newPage();

});

afterAll(async () => {

await browser.close();

});

it('should accept the cookie dialog and search for the doctor and city', async () => {

await page.goto('https://demo.clickdoc.de/cd-de/');

const acceptCookieButton = await page.waitForSelector('button[aria-label="Alle Cookies akzeptieren"]');

await acceptCookieButton.click();

// Wait for the page to load after accepting the cookies.

await page.waitForTimeout(5000);

// Fill the search fields and click 'Finden'.

await page.fill('input[name="q"]', 'Peter Wunderlich');

await page.fill('input[name="l"]', 'Testhausen');

await page.click('button[type="submit"]');

// Wait for the search results to load.

await page.waitForSelector('.search-results-list', { visible: true });

// Validate 'Peter Wunderlich' physician view was displayed as first in the list of results.

const searchResults = await page.$$('.search-results-list .card-item');

const firstResultName = await (await searchResults[0].$eval('h3', el => el.textContent)) || '';

expect(firstResultName).toBe('Peter Wunderlich');

// Validate physician name and address.

const physicianName = await page.textContent('h3.card-title');

const physicianAddress = await page.textContent('.doctor-address');

expect(physicianName).toBe('Peter Wunderlich');

expect(physicianAddress).toBe('Testhausen, Musterstra├če 123');

// Validate that he is online bookable for the month of April 2024.

const calendar = await page.waitForSelector('.cd-calendar-month');

const calendarDays = await calendar.$$('.week-day');

for (const day of calendarDays) {

const dateText = await (await day.$eval('span', el => el.textContent)) || '';

if (dateText.startsWith('2')) {

// Check for availability in the month of April 2024.

const availableSlot = await day.$('.cd-calendar-day-slot');

expect(availableSlot).toBeTruthy();

break;

}

}

// Validate the color of 'Find' and 'Appointment booking' buttons in the page.

const findButton = await page.$('.cd-search-button');

const appointmentButton = await page.$('.cd-book-appointment-btn');

const [findButtonColor, appointmentButtonColor] = await Promise.all([

findButton?.evaluate(el => getComputedStyle(el).backgroundColor),

appointmentButton?.evaluate(el => getComputedStyle(el).backgroundColor),

]);

// The color values may vary depending on your system and browser.

// You can use a tool like 'https://www.colorhexa.com/' to find the actual color values.

expect(findButtonColor).toBe('rgba(0, 123, 255, 1)');

expect(appointmentButtonColor).toBe('rgba(0, 123, 255, 1)');

});

});

| Steps | Test data | Expectation |
| --- | --- | --- |
| * Navigate to the practice page of Peter Wunderlich |  | * Validate physician name and address * Validate the current day opening hours from contact section * Validate the current day is shown in bold   Note:   * Validate current day start and end time of morning and afternoon shift timings example: (09:00 Uhr, 12:00 Uhr, 14:00 Uhr, 18:00 Uhr)   Tips:   * Implement a function to get the opening hours * Use constants, interface, etc. |

// test3.ts

import { chromium, Browser, Page, ElementHandle } from 'playwright';

interface OpeningHours {

morningStart: string;

morningEnd: string;

afternoonStart: string;

afternoonEnd: string;

}

const getOpeningHours = async (page: Page): Promise<OpeningHours> => {

const openingHoursList = await page.$$('.cd-opening-hours .cd-opening-hours-list-item');

const morningShift = openingHoursList[0];

const afternoonShift = openingHoursList[1];

const openingHours: OpeningHours = {

morningStart: await (await morningShift.$eval('time', el => el.getAttribute('datetime'))).slice(0, 5),

morningEnd: await (await morningShift.$eval('time', el => el.getAttribute('datetime'))).slice(0, 5),

afternoonStart: await (await afternoonShift.$eval('time', el => el.getAttribute('datetime'))).slice(0, 5),

afternoonEnd: await (await afternoonShift.$eval('time', el => el.getAttribute('datetime'))).slice(0, 5),

};

return openingHours;

};

describe('ClickDoc DE Demo', () => {

let browser: Browser;

let page: Page;

beforeAll(async () => {

browser = await chromium.launch();

page = await browser.newPage();

});

afterAll(async () => {

await browser.close();

});

it('should navigate to the practice page of Peter Wunderlich and validate the details', async () => {

await page.goto('https://demo.clickdoc.de/cd-de/');

// Search for the doctor and city.

await page.fill('input[name="q"]', 'Peter Wunderlich');

await page.fill('input[name="l"]', 'Testhausen');

await page.click('button[type="submit"]');

await page.waitForSelector('.search-results-list', { visible: true });

// Click on the first search result.

await page.click('.search-results-list .card-item:first-child a');

await page.waitForTimeout(5000);

// Validate physician name and address.

const physicianName = await page.textContent('h1.doctor-name');

expect(physicianName).toBe('Peter Wunderlich');

const physicianAddress = await page.textContent('.doctor-address');

expect(physicianAddress).toBe('Testhausen, Musterstra├če 123');

// Validate the current day opening hours from the contact section.

const openingHours = await getOpeningHours(page);

const currentDay = new Date().toLocaleString('default', { weekday: 'long' });

const currentDayOpeningHours = await page.textContent(

`.cd-opening-hours-list-item:nth-child(2) .cd-opening-hours-list-item-title:contains("${currentDay}")`

);

expect(currentDayOpeningHours).toBe(`${currentDay}`);

// Validate the current day start and end time of morning and afternoon shift timings.

const openingHoursValidation = {

morning: `09:00 Uhr - ${openingHours.morningEnd} Uhr`,

afternoon: `14:00 Uhr - ${openingHours.afternoonEnd} Uhr`,

};

const openingHoursValidationSelector = '.cd-opening-hours-list-item:nth-child(2) .cd-opening-hours-list-item-content';

const openingHoursValidationText = await page.textContent(openingHoursValidationSelector);

for (const [key, value] of Object.entries(openingHoursValidation)) {

expect(openingHoursValidationText).toContain(value);

}

});

});

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

To implement the test automation with the provided requirements and constraints, follow the steps below:

1. Set up the project and install required packages:

mkdir webdriverio-typescript-mocha-allure

2cd webdriverio-typescript-mocha-allure

3npm init -y

4npm install webdriverio @types/webdriverio mocha ts-node @types/node allure-mocha-reporter @wdio/cli @wdio/mocha-framework @wdio/chrome-service --save-dev

1. Create a wdio.conf.ts configuration file:

// wdio.conf.ts

import { config } from 'dotenv';

import { chromium, BrowserType } from 'webdriverio';

config();

export const config: WebdriverIO.Config = {

services: [

['chromium', { logPath: 'reports/chromium.log' }],

['allure', { outputDir: 'reports/allure-results', enableHover: true }],

],

reporters: [['allure'], ['spec']],

framework: 'mocha',

mochaOpts: {

ui: 'bickerton',

timeout: 10000,

},

capabilities: [{ maxInstances: 1, browserName: 'chrome', 'goog:chromeOptions': { args: ['--start-maximized'] } }],

logLevel: 'trace',

waitforTimeout: 5000,

connectionRetryCount: 3,

connectionRetryTimeout: 90000,

path: '/',

specs: ['./\*\*/\*.spec.ts'],

onPrepare() {

require('ts-node').register({ project: 'tsconfig.json' });

},

beforeScenario: async function (scenario) {

await browser.setWindowSize(1920, 1080);

await browser.url(`https://demo.clickdoc.de/cd-de/`);

},

};

const chromeService = require('@wdio/chrome-service');

config.services.push(chromeService.service({ logPath: 'reports/chromium.log' }));

1. Create a .gitignore file:

node\_modules/

2reports/

3coverage/

1. Create a tsconfig.json file:

{

"compilerOptions": {

"target": "es6",

"module": "commonjs",

"strict": true,

"esModuleInterop": true,

"outDir": "dist",

"types": ["node", "webdriverio/async\_hooks\_compatibility"],

"resolveJsonModule": true,

"declaration": true

},

"include": ["./\*\*/\*.ts"],

"exclude": ["node\_modules"]

}

1. Create a .env file:

TEST\_URL=https://demo.clickdoc.de/cd-de/

1. Implement the test specifications:

// search.spec.ts

import { expect } from 'chai';

describe('ClickDoc DE Demo', () => {

let page: WebdriverIO.Browser;

beforeEach(async () => {

page = browser;

await page.refresh();

});

it('should search for a doctor and practice', async () => {

await page.setValue('#q', 'Peter Wunderlich');

await page.setValue('#l', 'Testhausen');

await page.click('button[type="submit"]');

await page.waitForSelector('.search-results-list', { visible: true });

const firstResultName = await page.$eval('.search-results-list .card-item:first-child .card-title', (el) => el.textContent);

expect(firstResultName).to.equal('Peter Wunderlich');

});

it('should navigate to the practice page and validate the physician name and address', async () => {

await page.setValue('#q', 'PeterWunderlich');

await page.setValue('#l', 'Testhausen');

await page.click('button[type="submit"]');

await page.waitForSelector('.search-results-list', { visible: true });

await page.click('.search-results-list .card-item:first-child a');

await page.waitForSelector('.doctor-name', { visible: true });

const physicianName = await page.$eval('.doctor-name', (el) => el.textContent);

expect(physicianName).to.equal('Peter Wunderlich');

const physicianAddress = await page.$eval('.doctor-address', (el) => el.textContent);

expect(physicianAddress).to.equal('Testhausen, Musterstra├če 123');

});

// Add more test cases as needed

});

1. Add a start script in package.json:

"scripts": {

"start": "wdio wdio.conf.ts"

}

1. Run the test:

npm start

This setup meets the requirements for using WebdriverIO with TypeScript, Mocha framework, and Allure reporter. It also uses Chrome Browser for testing. The tests are implemented in a separate spec.ts file, and the wdio.conf.ts file is configured to use the Chrome service.