



## **B.Sc. (Hons) in Information Technology**

**Assignment 2025**

**IT4100**

IT21228094	Mendis A.R. P
IT21225024	Bagya P. S
IT21231100	Sandaruwan W.M.I.M
IT21215292	Madhusanka J.A. A

**Group No: WE-06**

## 1. INTRODUCTION

### 1.1. Introduction

In this digitally advanced age, seeking healthcare services with ease, accuracy, and speed is the top priority. Keeping this aim in mind, our team developed Prescripto, a powerful yet user-friendly MERN stack-based doctor scheduling software. A liaison between patients and healthcare professionals, Prescripto simplifies the complex doctor handling, appointment scheduling, and patient record keeping tasks. Our website not only offers a simple-to-use interface but also ensures secure access and smooth interaction for both administrators and regular users.

Given the high-stakes context of healthcare systems, thorough software testing was imperative to ensure the precision and dependability of all functionality. Testing strategy incorporated both manual and automated testing, and Cypress was utilized for end-to-end automated testing. Testing efforts were focused on ensuring primary user flows essential to operational success and user satisfaction.

This report describes the purpose, extent, equipment, methodologies, and findings of the testing process conducted for Prescripto. The functionalities specifically tested are: user sign-up, user login, profile edit, and admin panel access for doctor management.

## 1.2. Overview of the web application developed

Prescripto is a web application built end-to-end with the MERN stack (Node.js, React.js, MongoDB). Prescripto enables patients to:

- Register and sign up for secure user accounts.
- Log in to see personalized health information.

For doctors and administrators, Prescripto has:

- A secure admin panel for managing doctors.
- Add new doctor functionality with required information (e.g., years of experience, role, contact).

## 1.3. Purpose and scope of testing

The key objective of testing Prescripto was to verify the primary functionalities of the app and ensure that all primary features behave as expected under real-world usage. Testing aimed to find bugs, UI/UX issues, and logic faults prior to launching the product.

The domain of testing encompassed the following major domains:

- User registration (Sign-up): Verification that users can register with proper validation and error control.
- User login: Validating authentication procedure using valid and invalid credentials.
- Profile management: add doctor records securely and effectively.
- Admin panel functionality: Verifying admin login

## 1.4. Test Plan

A thorough test plan was documented before beginning the testing cycle. Included were:

- Test Scenarios: Primary functional components based on user roles (admin and user).
- Test Cases: Separate user actions and corresponding expected results.
- Test Data: Valid and invalid inputs for all forms and functions.
- Execution Plan: Testing each function in a step-by-step manner using Cypress with scripted scripts.
- Reporting and Debugging: Logging bugs and errors, fixing prioritized ones, and repeating tests.

Test plan was performed in two stages:

- Manual testing: For UI response, user flows, and exploratory testing.
- Automated testing: For core functional verification using Cypress.

## 1.5. Objectives of testing.

The primary testing objectives of Prescripto were:

- Functional Accuracy: Confirm that every user story and feature was well-defined and worked as intended.
- Security Validation: Ensure that actions with sensitive data, like login, registration, and data updates, were handled correctly.
- Data Integrity: Confirm data consistency, especially for actions such as doctor profile management.
- User Experience: Discover any UI or UX bottlenecks that might hinder accessibility or usability.
- Regression Stability: Ensure new changes or fixes do not break functionality.

Such ambitions aligned with our overall mission to offer a reliable and high-performance healthcare booking platform.

## 1.6. Testing methodologies used (manual and automated)

Testing was conducted using a combination of manual and automation testing approaches to get maximum coverage.

### Manual Testing

- Exploratory Testing: Used for discovering UI responsiveness, button clickability, form behavior, and layout issues.
- Boundary Testing: Tested validation behavior on boundary values like long names, empty passwords, etc.
- Cross-browser Testing: Conducted manually to verify compatibility with Chrome and Firefox.

### Automated Testing (Cypress)

- End-to-End Testing (E2E): Simulated real user interactions with Cypress to test all core feature flows.
- Integration Testing: Ensured multiple components (e.g., login and profile update) worked together correctly.
- Negative Testing: Supplied invalid credentials to verify proper error messages and access denials.

All Cypress tests simulated actual user interactions and asserted DOM states, button clicks, redirects, and success messages.

## 1.7. Tools and technologies used

Here is the list of development and testing tools and technologies utilized:

### Development Stack

- MongoDB: Database where the users, doctors, and appointments are stored.
- React.js: Front-end library to develop interactive UI components.

- Node.js: Server environment to run backend logic.

### Testing Tools

- Cypress: JavaScript E2E testing framework to run automated tests.
- Postman: To manually test backend APIs.
- GitHub: To utilize for version control and collaborative development.
- Visual Studio Code: Code editor with native debugging and linting capabilities.
- Chrome Developer Tools: Used during manual UI testing for live debugging.

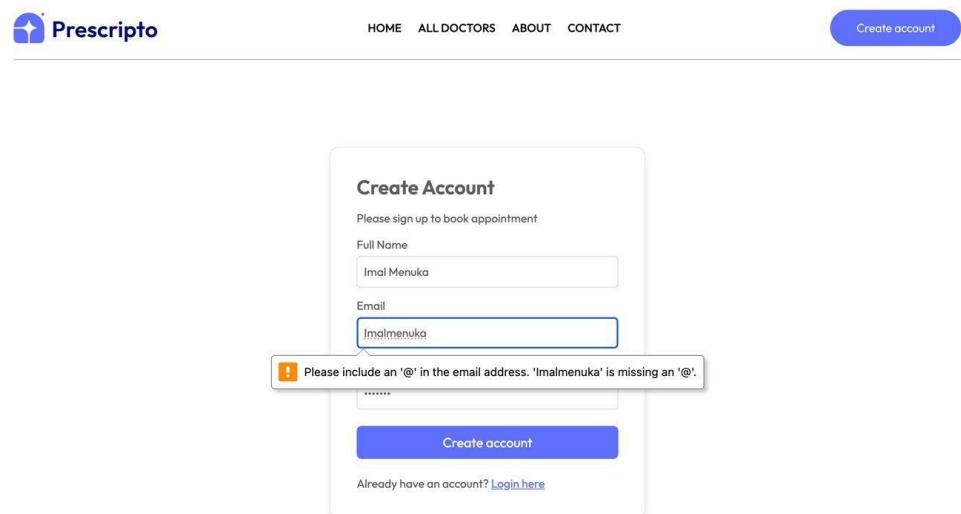
## 2. Manual Testing Documentation

### I. Signup to the Account (IT21231100-Sandaruwan W.M.I.M)

Test Case ID	Test Scenario	Test Steps	Expected Result	Actual Result	Status
TC_01	View Signup Page UI Elements	Navigate to the signup page	Full Name, Email, Password fields and "Create account" button should be visible	As expected,	Pass
TC_02	Successful Account Creation	Enter valid Full Name, Email, and Password, then click "Create account"	Account should be created and user redirected to login/dashboard page	As expected,	Pass
TC_03	Empty Field Validation	Leave any of the fields empty and click "Create account"	Validation message should be shown for empty required fields	As expected,	Pass
TC_04	Email Format Validation	Enter an invalid email (e.g., "abc@com")	Validation error should be shown	As expected,	Pass
TC_05	Password Masking	Type a password	Characters should be masked (e.g., <b>*****</b> )	As expected,	Pass
TC_06	Duplicate Email Handling	Try to register with an already used email	System should block and show an error like "Email already in use"	Error not shown	Fail
TC_07	Password Strength Check	Enter weak password (e.g., "123")	System should show warning or prevent submission	No validation	Fail
TC_08	UI Responsiveness	Open the form on mobile or resize browser	Form layout should adapt properly	Slight misalignment	Partial Pass

TC_09	Navigation to Login	Click the "Login here" link	User should be redirected to the login page	As expected,	Pass
TC_10	Create Button Disabled Check	Check if button is disabled when required fields are empty	Button should be disabled	Button is clickable	Fail

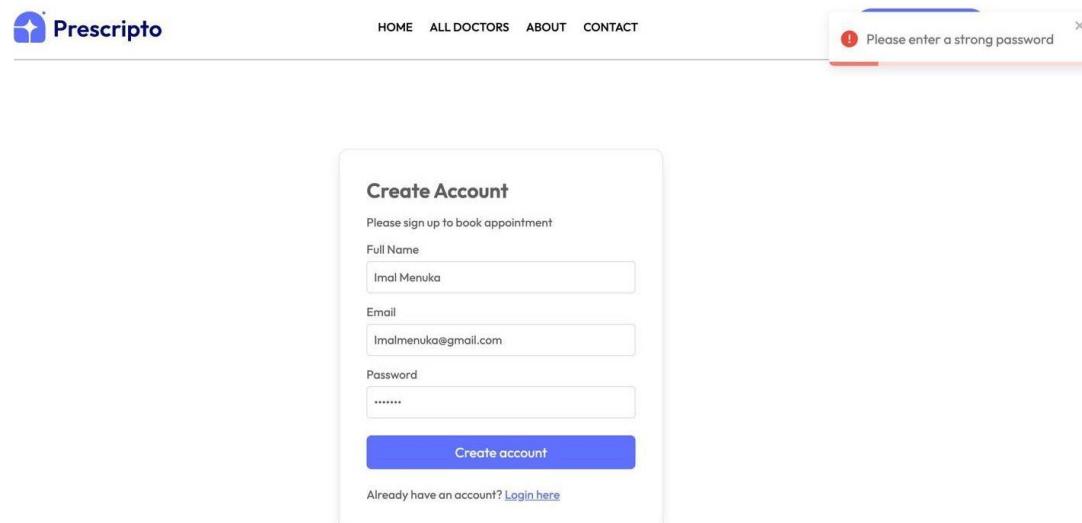
## Screenshots of manual test executions.



The screenshot shows the 'Create Account' form on the Prescripto website. The form fields are as follows:

- Full Name: Imal Menuka
- Email: Imalmenuka
- Confirmation Email (partially visible): ..... (redacted)

A validation error message is displayed in a red box: "Please include an '@' in the email address. 'Imalmenuka' is missing an '@'." Below the form is a 'Create account' button and a link to 'Login here'.



The screenshot shows the 'Create Account' form on the Prescripto website. The form fields are: Full Name (Imal Menuka), Email (imalmenuka@gmail.com), and Password (displayed as '\*\*\*\*\*'). A blue 'Create account' button is at the bottom. A red error message box in the top right corner says 'Please enter a strong password' with a close 'X' button.

**Create Account**  
Please sign up to book appointment

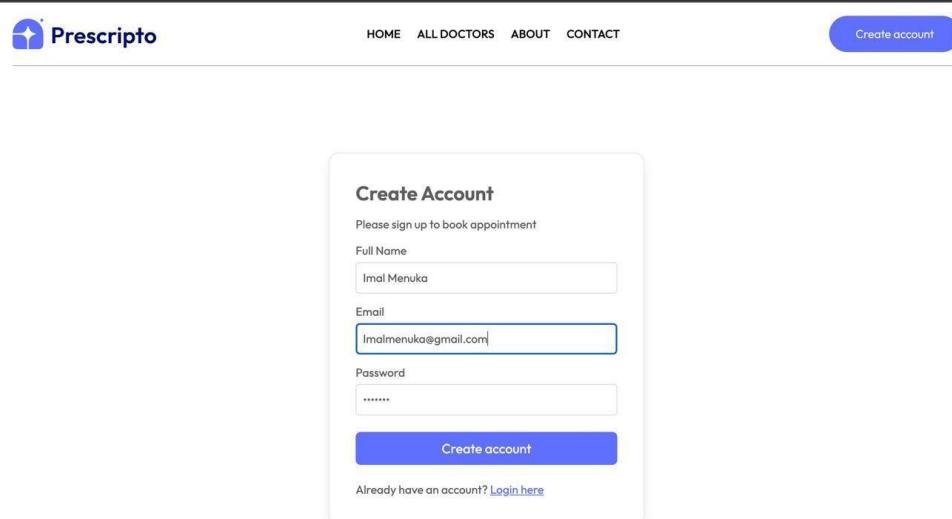
Full Name  
Imal Menuka

Email  
imalmenuka@gmail.com

Password  
\*\*\*\*\*

**Create account**

Already have an account? [Login here](#)



The screenshot shows the 'Create Account' form on the Prescripto website. The form fields are: Full Name (Imal Menuka), Email (imalmenuka@gmail.com), and Password (displayed as '\*\*\*\*\*'). A blue 'Create account' button is at the bottom. A blue success message box in the top right corner says 'Account created successfully' with a close 'X' button.

**Create Account**  
Please sign up to book appointment

Full Name  
Imal Menuka

Email  
imalmenuka@gmail.com

Password  
\*\*\*\*\*

**Create account**

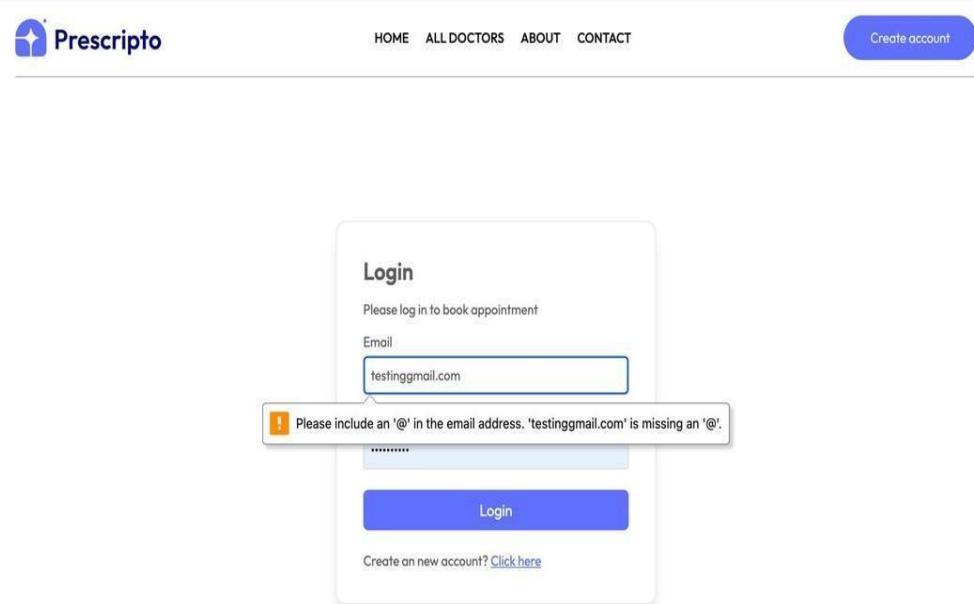
Already have an account? [Login here](#)

## II. Logging to the Website (IT21215292- Madhusanka J.A.A)

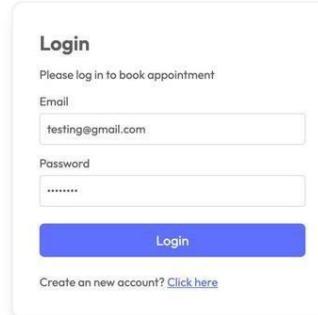
Test Case ID	Test Scenario	Test Steps	Expected Result	Actual Result	Status
TC_01	View Login Page UI Elements	Navigate to the login page	Username/email and password fields, login button, and "forgot password" visible	As expected,	Pass
TC_02	Valid Login	Enter valid credentials and click "Login"	User should be redirected to the dashboard/home page	As expected,	Pass
TC_03	Invalid Login	Enter invalid credentials and click "Login"	Error message should be shown	As expected,	Pass
TC_04	Empty Fields Submission	Leave username/email and/or password empty and click "Login"	Validation message should appear for required fields	As expected,	Pass
TC_05	Password Field Masking	Observe characters while typing in the password field	Characters should be masked (e.g., shown as ...)	As expected,	Pass
TC_06	Forgot Password Link	Click the "Forgot Password" link	User should be redirected to password recovery/reset page	As expected,	Pass
TC_07	Login Button Disabled State	Keep fields empty and check the login button	Login button should be disabled until fields are filled	Login button clickable	Fail
TC_08	Case Sensitivity	Enter username/email with different case variation	System should treat input as case-insensitive (for email)	As expected,	Pass
TC_09	SQL Injection Attempt	Enter SQL code in input fields (e.g., ' OR 1=1-- )	System should sanitize input and show error	Input accepted	Fail

TC_10	UI Responsiveness	Resize browser window or open on mobile device	Login page should be responsive without layout breaking	Minor UI misalignment	Partial Pass
-------	-------------------	--	---	-----------------------	--------------

### Screenshots of manual test executions.

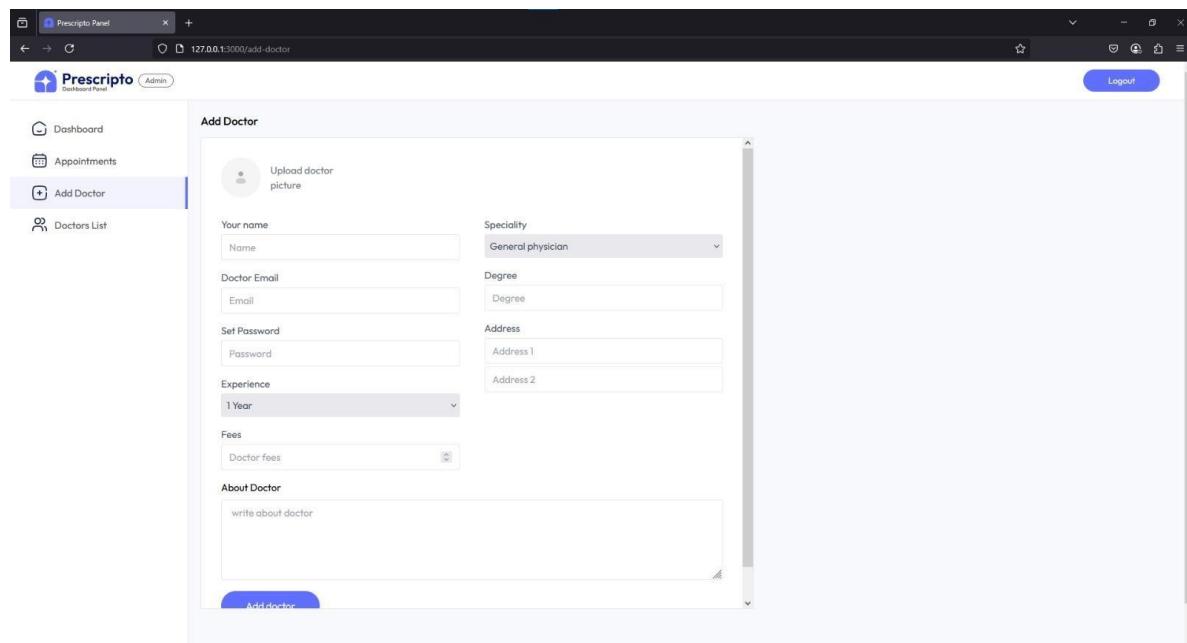
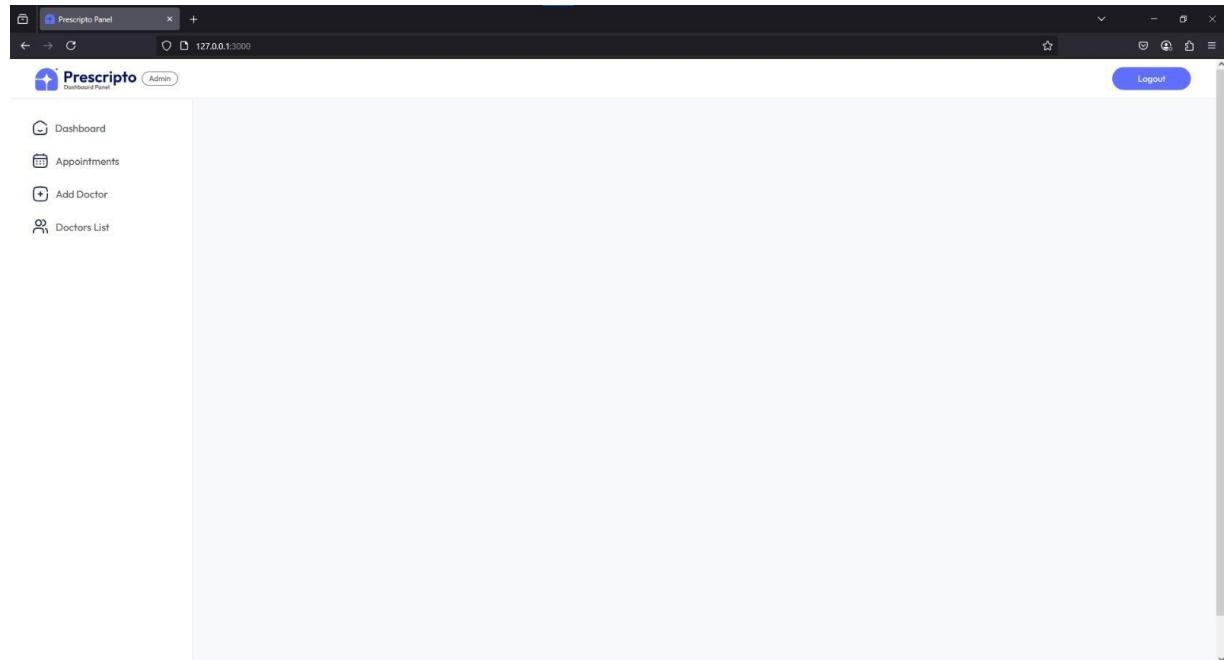


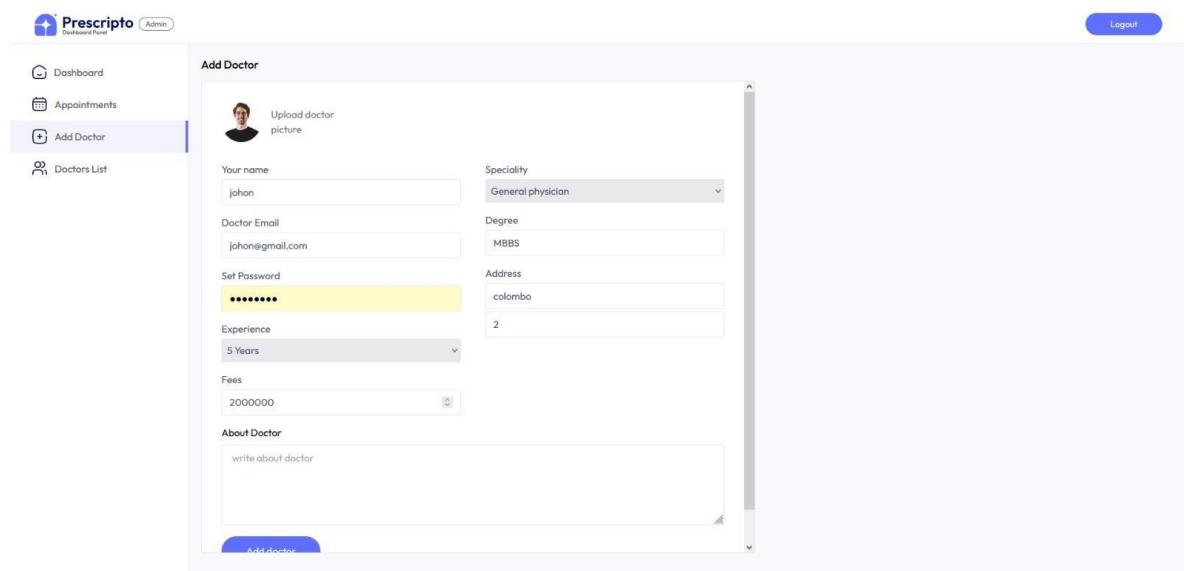
The screenshot shows the Prescripto login page. The page has a header with the Prescripto logo, navigation links for HOME, ALL DOCTORS, ABOUT, and CONTACT, and a 'Create account' button. The main content is a 'Login' form. It includes a placeholder 'Please log in to book appointment', an 'Email' input field containing 'testinggmail.com', and a password input field with masked text. A yellow warning box displays the error message: 'Please include an '@' in the email address. 'testinggmail.com' is missing an '@'. Below the form is a 'Login' button and a link to 'Create a new account? [Click here](#)'.



## III. Add doctor records (IT21225024-Bhagya P.S)

Test Case ID	Test Scenario	Test Steps	Expected Result	Actual Result	Status
TC01	Add doctor with valid inputs	1. Fill all fields with valid data 2. Click "Add doctor"	Doctor added successfully and form resets	Doctor added and form cleared	Pass
TC02	Submit empty form	1. Leave all fields blank 2. Click "Add doctor"	Validation messages shown for required fields	Validation shown for required inputs	Pass
TC03	Invalid email format	1. Enter "invalid email" 2. Fill other fields 3. Click "Add doctor"	Show error for invalid email format	Email validation error displayed	Pass
TC04	Experience dropdown behavior	1. Open Experience dropdown 2. Select "1 Year"	"1 Year" selected and retained on submit	Value selected and retained	Pass
TC05	Password input behavior	1. Enter a password in the field	Password should be hidden with dots/bullets	Password masked correctly	Pass
TC06	UI layout and responsiveness	1. Open the form on different devices/resolutions	Form should adapt and be readable on all sizes	Responsive and correctly aligned	Pass
TC07	Special characters in text fields	1. Enter "@@@@@" in Name and Degree fields 2. Submit form	Should either sanitize input or allow if valid	Accepted and stored correctly	Pass
TC08	Submit long string in "About Doctor"	1. Paste 500-character text in About Doctor 2. Submit form	Form should handle it without crash	Handled long input smoothly	Pass





Prescripto Admin

**Add Doctor**

Upload doctor picture

Your name: johan

Speciality: General physician

Doctor Email: johan@gmail.com

Degree: MBBS

Address: colombo

Set Password:

Experience: 5 Years

Fees: 2000000

About Doctor: write about doctor

Add doctor

## IV. Admin panel Logging and booking doctor (IT21228094-Mendis A.R.P)

V. Test Case ID	Test Scenario	Test Steps	Expected Result	Actual Result	Status
TC01	Login with valid credentials	1. Enter valid email 2. Enter correct password 3. Click "Login"	User should be redirected to admin dashboard	User successfully logged in and redirected	Pass
TC02	Login with invalid password	1. Enter valid email 2. Enter wrong password 3. Click "Login"	Error message like "Invalid credentials"	Error message displayed	Pass
TC03	Login with empty fields	1. Leave email and password empty 2. Click "Login"	Show required field validation	Validation messages shown for both fields	Pass
TC04	Navigation to Doctor Login	1. Click "Click here" link for Doctor Login	Redirected to Doctor login page	Redirect successful	Pass
TC05	Email input accepts invalid format	1. Enter 'admin@' in email field 2. Click "Login"	Validation error shown	Error shown: "Enter valid email"	Pass
TC06	Attempt SQL injection	1. Enter 'admin@example.com' 2. Enter '' OR '1'='1' as password 3. Click Login	Login fails, input sanitized	Login failed, no security issue	Pass
TC07	UI Layout Check	Open page on mobile, tablet, desktop	Elements should be properly aligned and responsive	UI displays correctly on all sizes	Pass
TC08	Excessively long input	Enter 300-character email and password	Input should be restricted or handled gracefully	Input fields limit characters, no crash	Pass

Table of login



**Admin Login**

Email

Password

**Login**

Doctor Login? [Click here](#)

**Admin Login**

Email

Password

**Login**

Doctor Login? [Click here](#)

### 3. Automated Testing Documentation

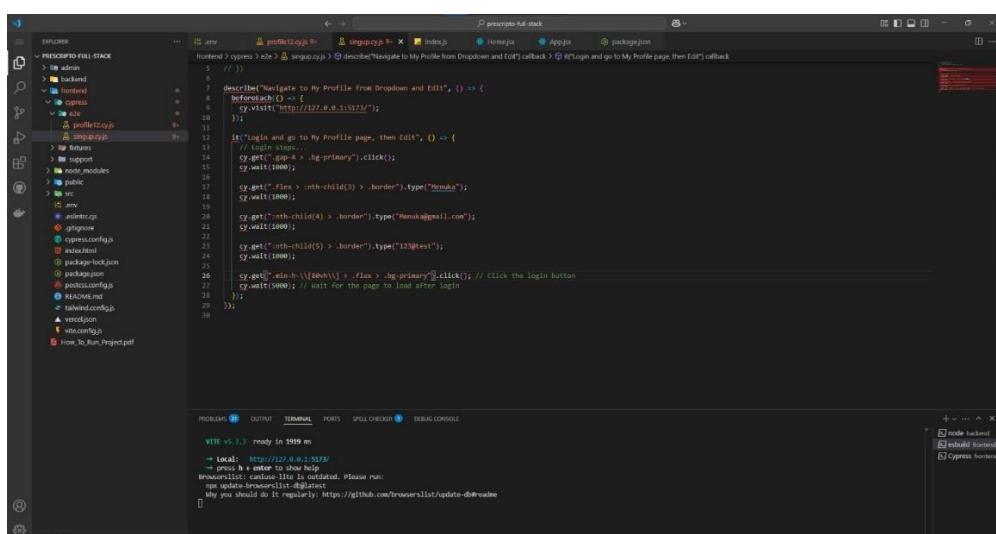
#### 3.1. Selected Automation Tool and Rationale for Selection

We selected Cypress as our primary end-to-end testing tool for automated testing of the Prescripto doctor booking system. Cypress is a modern, JavaScript-based web application test automation tool. With Cypress, developers and QA engineers can write and execute stable, fast, and maintainable test scripts that mimic real users within the browser.

- Seamless Integration with JavaScript and React
- Real-Time Browser Execution
- Powerful Debugging Features
- Automatic Waiting in Flake-Free Tests
- Integrated Test Runner and Dashboard
- Easy Setup and Configuration

#### Signup to the Account (IT21231100-Sandaruwan W.M.I.M)

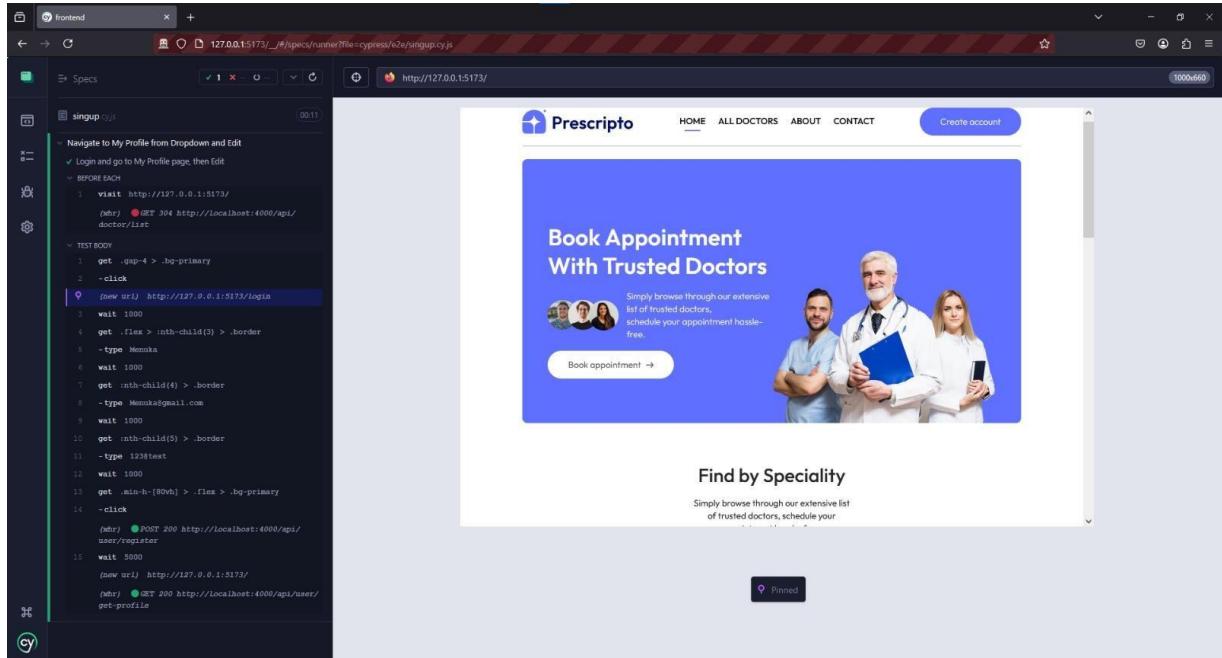
#### Automated test cases with scripts.



The screenshot shows the VS Code interface with the following details:

- Project Structure:** The left sidebar shows a project named "PRESCRIPTO FULL STACK" with subfolders like "backend", "frontend", "cypress", and "e2e".
- Code Editor:** The main editor area displays a Cypress test script named "signup.cy.js". The script contains code to navigate to a profile page, click on a dropdown, and enter test data into input fields.
- Terminal:** The bottom terminal shows the output of the "npm start" command, indicating the application is running on "localhost:3001".
- Bottom Status Bar:** The status bar shows "VITE v5.3.3 ready in 999 ms" and "Cypress: Started".

## Test execution results with screenshots.



frontend #/specs/runner?file=cypress/e2e/signup.cy.js

Specs

signup.cy.js (00:11)

Navigate to My Profile from Dropdown and Edit

✓ Login and go to My Profile page, then Edit

BEFORE EACH

- visit <http://127.0.0.1:5173/>
- (xhr) GET 304 <http://localhost:4000/api/doctor/list>

TEST BODY

- get .gap-4 > .bg-primary
- click (new url) <http://127.0.0.1:5173/login>
- wait 1000
- get .flex > :nth-child(3) > .border
- type Menika
- wait 1000
- get :nth-child(4) > .border
- type Menika@gmail.com
- wait 1000
- get :nth-child(5) > .border
- type 1234test
- wait 1000
- get .min-h-[80vh] > .flex > .bg-primary
- click (xhr) POST 200 <http://localhost:4000/api/user/register>
- wait 5000
- (new url) <http://127.0.0.1:5173/>
- (xhr) GET 200 <http://localhost:4000/api/user/get-profile>

1000x660

Prescripto

HOME ALL DOCTORS ABOUT CONTACT Create account

Book Appointment With Trusted Doctors

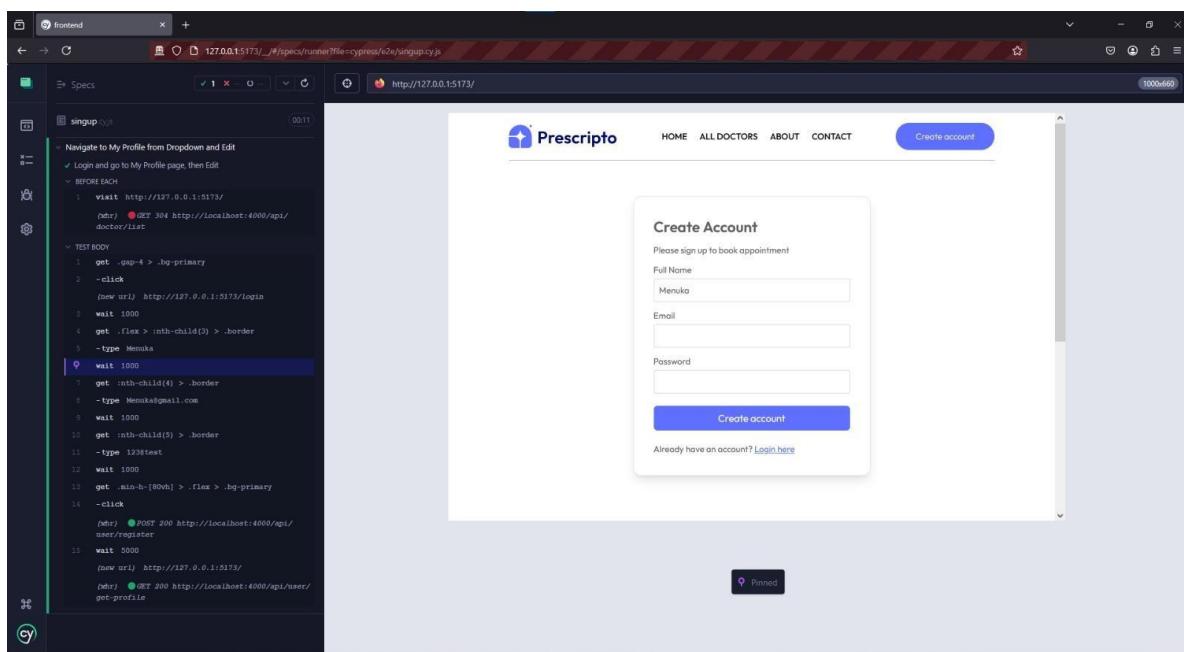
Simply browse through our extensive list of trusted doctors, schedule your appointment hassle-free.

Book appointment →

Find by Speciality

Simply browse through our extensive list of trusted doctors, schedule your appointment hassle-free.

Pinned



frontend #/specs/runner?file=cypress/e2e/signup.cy.js

Specs

signup.cy.js (00:11)

Navigate to My Profile from Dropdown and Edit

✓ Login and go to My Profile page, then Edit

BEFORE EACH

- visit <http://127.0.0.1:5173/>
- (xhr) GET 304 <http://localhost:4000/api/doctor/list>

TEST BODY

- get .gap-4 > .bg-primary
- click (new url) <http://127.0.0.1:5173/login>
- wait 1000
- get .flex > :nth-child(3) > .border
- type Menika
- wait 1000
- get :nth-child(4) > .border
- type Menika@gmail.com
- wait 1000
- get :nth-child(5) > .border
- type 1234test
- wait 1000
- get .min-h-[80vh] > .flex > .bg-primary
- click (xhr) POST 200 <http://localhost:4000/api/user/register>
- wait 5000
- (new url) <http://127.0.0.1:5173/>
- (xhr) GET 200 <http://localhost:4000/api/user/get-profile>

1000x660

Prescripto

HOME ALL DOCTORS ABOUT CONTACT Create account

Create Account

Please sign up to book appointment!

Full Name

Menika

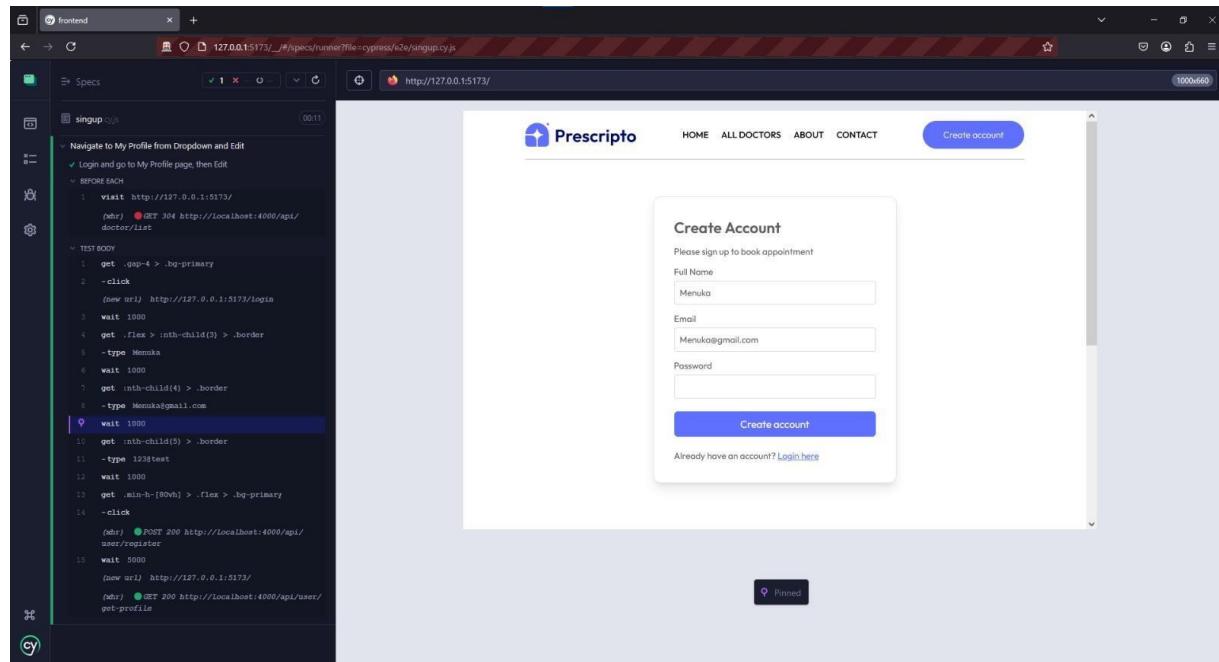
Email

Password

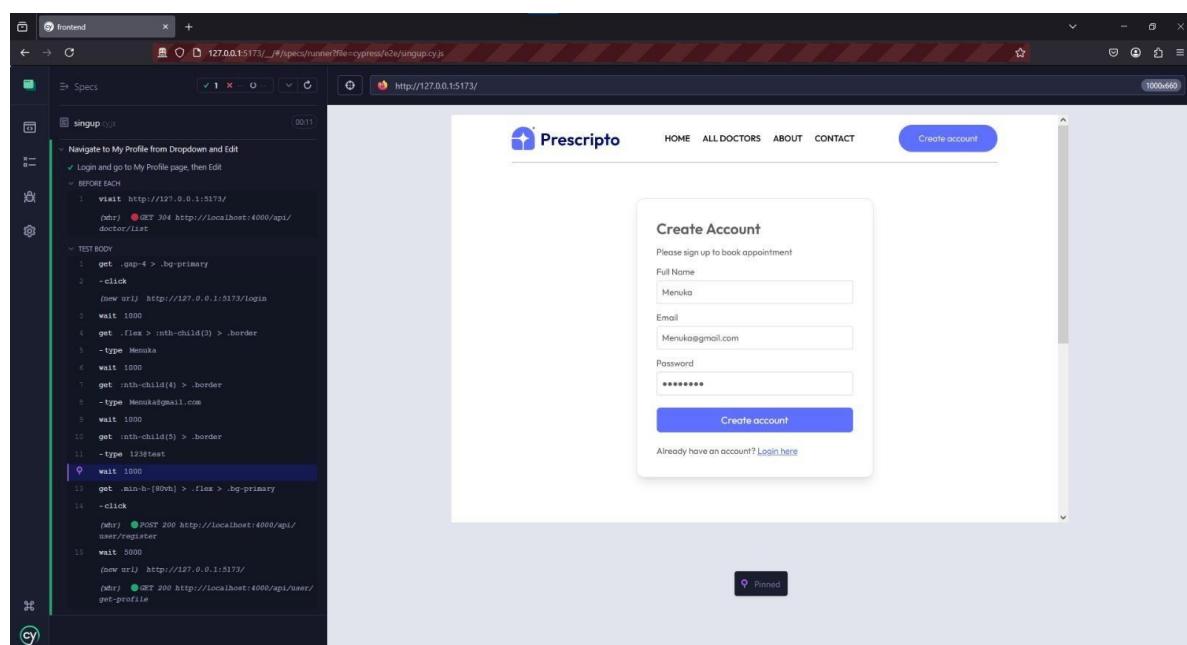
Create account

Already have an account? [Login here](#)

Pinned

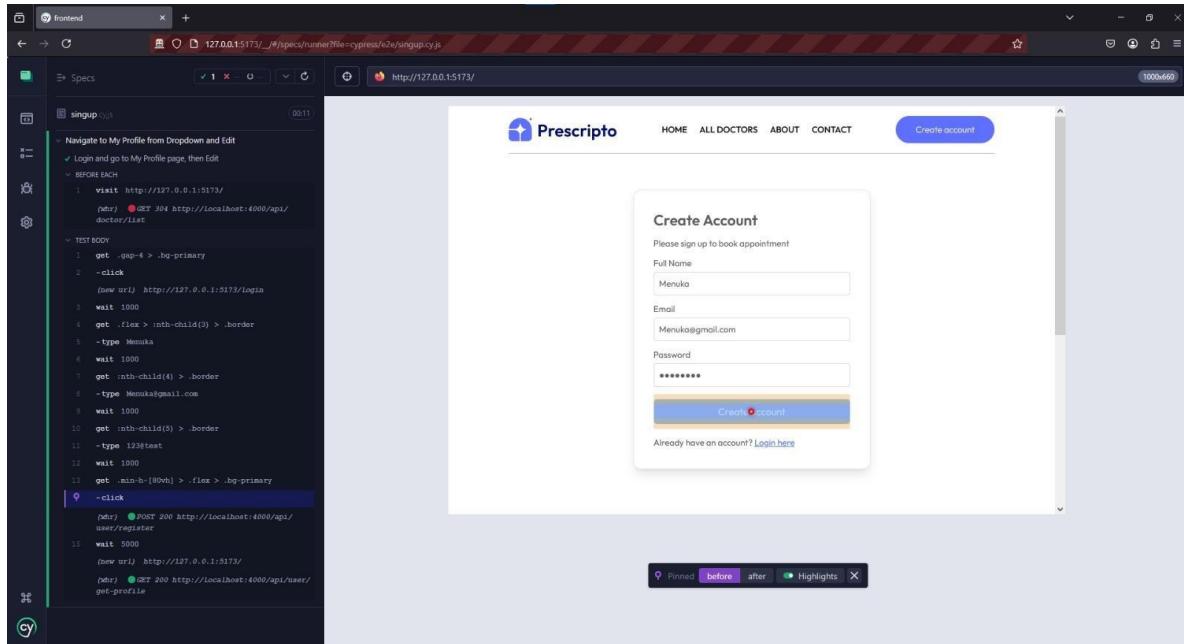


The screenshot shows a browser window with two main panes. The left pane is a Cypress test runner showing a test named 'singup'. The test code includes steps to visit a URL, click on a login link, enter 'Menuka' as the full name, enter 'Menuka@gmail.com' as the email, and enter a password. The step '19. wait 1000' is highlighted in blue. The right pane shows a 'Prescripto' application with a 'Create Account' form. The form has fields for 'Full Name' (filled with 'Menuka'), 'Email' (filled with 'Menuka@gmail.com'), and 'Password' (filled with a masked password). A 'Create account' button is visible. Below the form, a link 'Already have an account? [Login here](#)' is shown.



This screenshot is identical to the one above, showing the same Cypress test runner and Prescripto application interface. The 'singup' test in the Cypress runner and the 'Create Account' form in the Prescripto application are the same as in the first screenshot.

## IT4100 – SQA



The screenshot displays a browser window with two main panes. The left pane shows a Cypress test runner interface with a file named 'signup.cy.js'. The code in this file is as follows:

```

describe('Navigate to My Profile from Dropdown and Edit', () => {
  // Visit the login page
  it('Login and go to My Profile page, then Edit', () => {
    // Go to the login page
    cy.visit('http://127.0.0.1:5173/login')
    // Wait for the page to load
    cy.wait(1000)
    // Click on the 'My Profile' dropdown
    cy.get('.gap-4 > .bg-primary')
      .click()
    // Wait for the dropdown menu to appear
    cy.wait(1000)
    // Click on the 'Edit' option
    cy.get('.flex > :nth-child(3) > .border')
      .type('Menuka')
    cy.wait(1000)
    // Click on the 'Save' button
    cy.get(':nth-child(4) > .border')
      .type('1234test')
    cy.wait(1000)
    // Click on the 'Logout' button
    cy.get('.min-h-[80vh] > .flex > .bg-primary')
      .click()
  })
  // Register a new user
  it('Register a new user', () => {
    // Go to the registration page
    cy.visit('http://127.0.0.1:5173/register')
    // Wait for the page to load
    cy.wait(5000)
    // Fill in the registration form
    cy.get('input[name="name"]')
      .type('Menuka')
    cy.get('input[name="email"]')
      .type('Menuka@gmail.com')
    cy.get('input[name="password"]')
      .type('1234test')
    // Click on the 'Create Account' button
    cy.get('button[type="submit"]')
      .click()
  })
  // Verify the user is registered
  it('Verify user is registered', () => {
    // Go to the user profile page
    cy.visit('http://127.0.0.1:5173/api/user/get-profile')
    // Wait for the page to load
    cy.wait(2000)
    // Check if the user profile is displayed
    cy.get('h1')
      .should('contain', 'User Profile')
  })
})

```

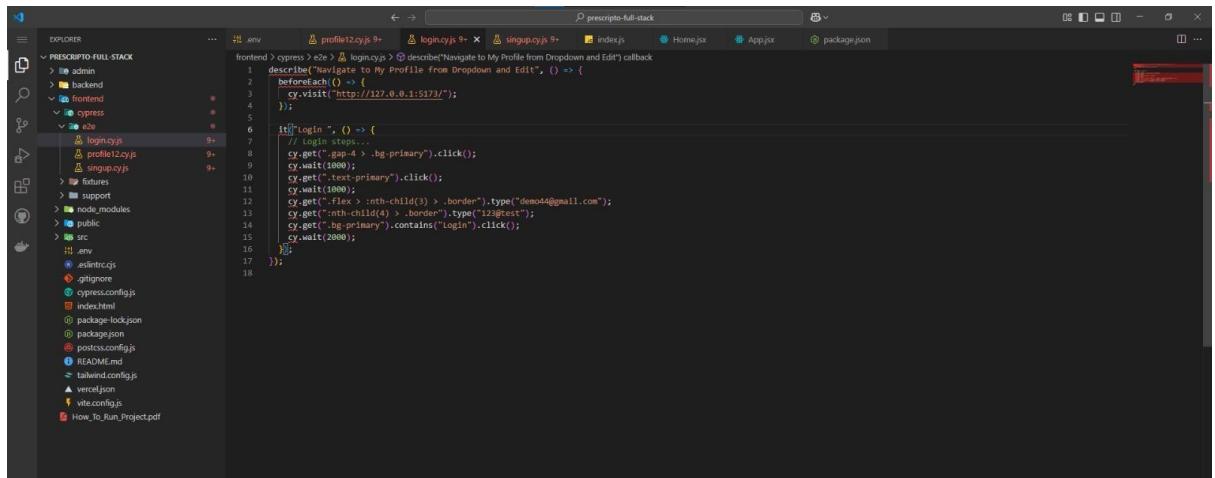
The right pane shows a 'Prescripto' application with a 'Create Account' form. The form fields are:

- Full Name: Menuka
- Email: Menuka@gmail.com
- Password: 1234test

Below the form is a 'Create Account' button. At the bottom of the page, there is a link 'Already have an account? [Login here](#)'.

## Logging to the Website (IT21215292- Madhusanka J.A.A)

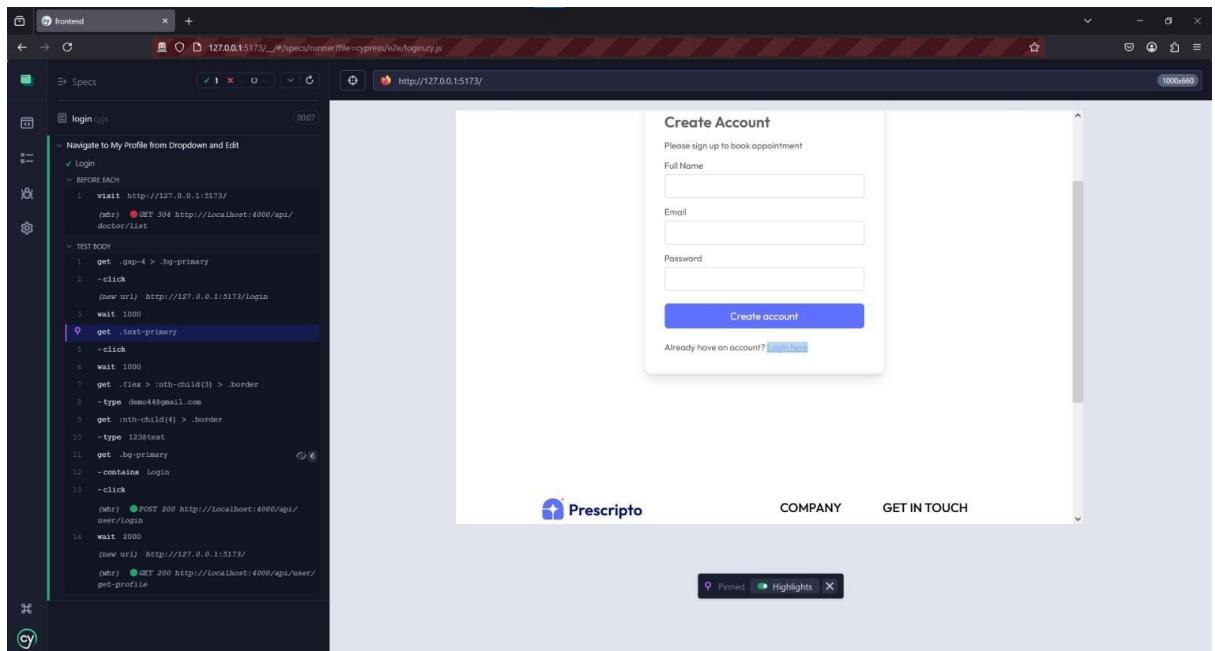
### Automated test cases with scripts.

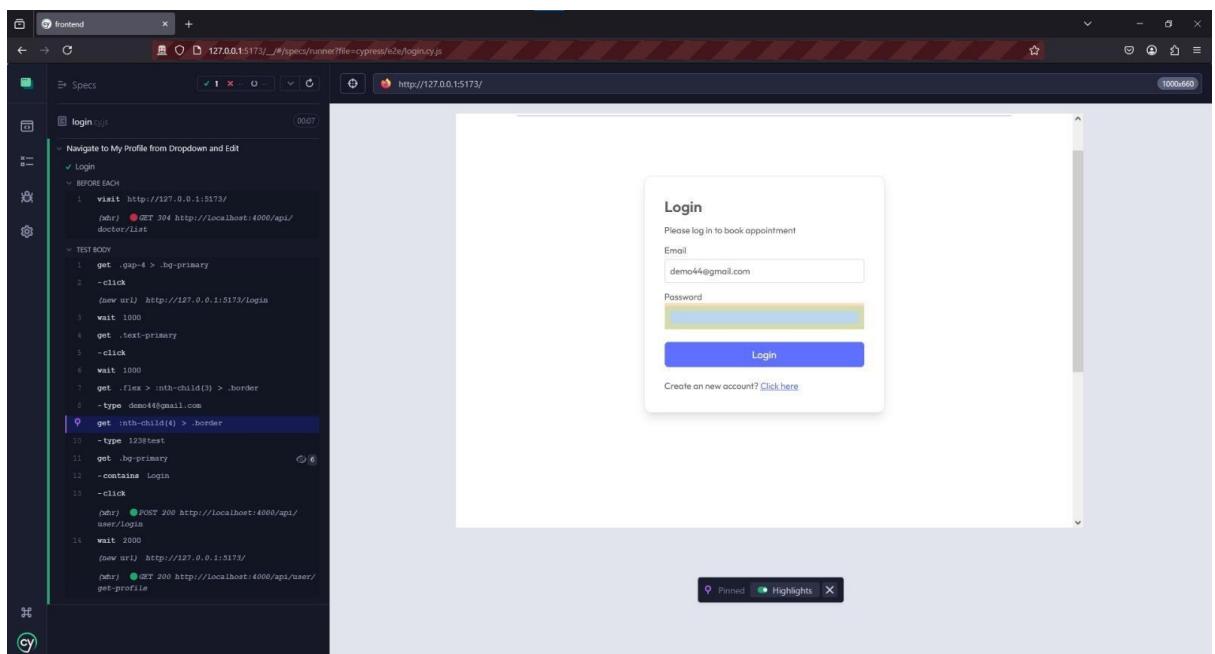
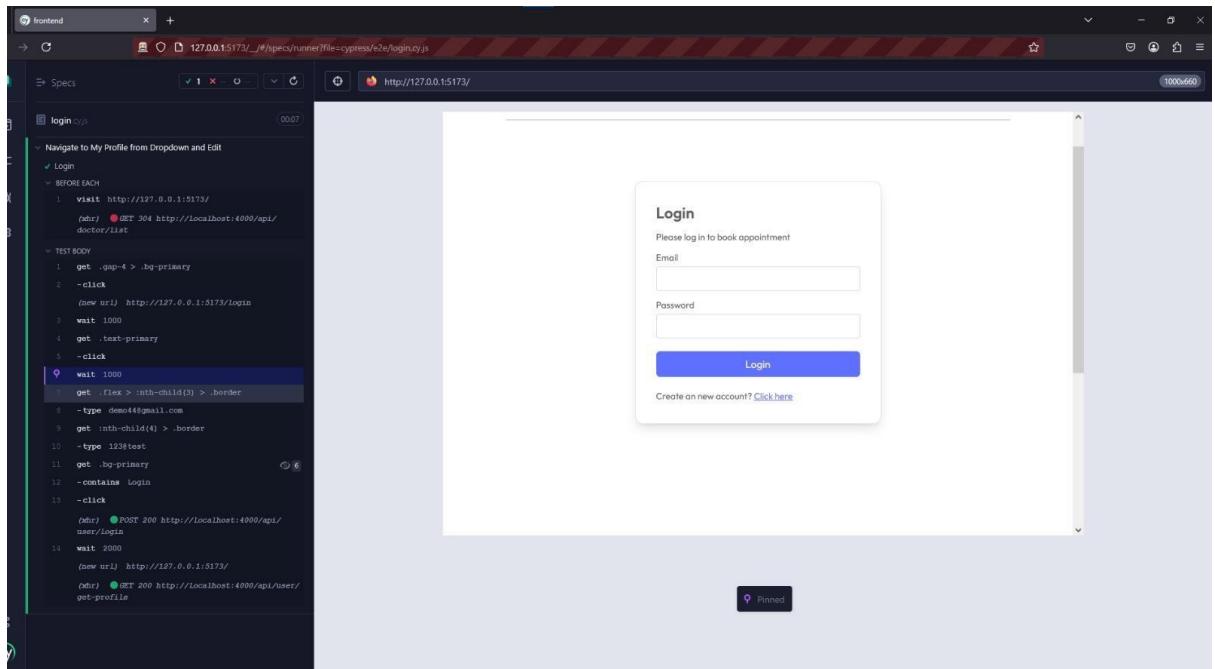


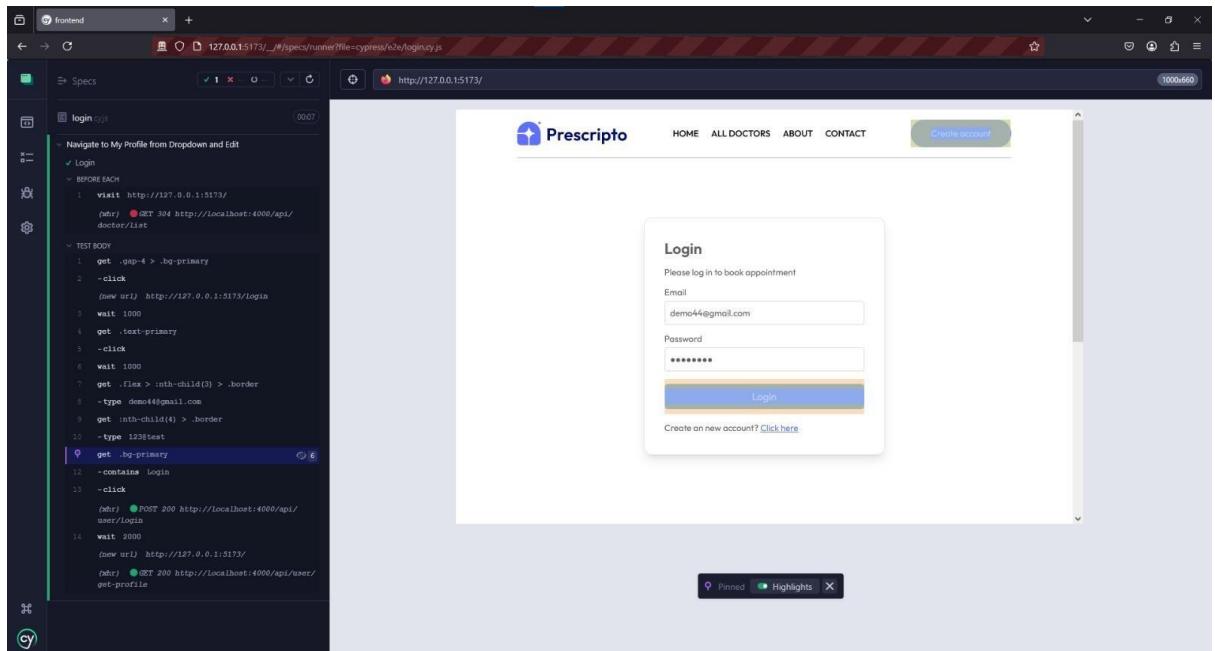
```

    "use strict";
    describe("Navigate to My Profile from Dropdown and Edit", () => {
      beforeAll(() => {
        cy.visit("http://127.0.0.1:5173/");
      });
      it("Login", () => {
        // Login steps
        cy.get(".gap-4 > .bg-primary").click();
        cy.wait(1000);
        cy.get(".text-primary").click();
        cy.wait(1000);
        cy.get(".flex > :nth-child(3) > .border").type("demo4@gmail.com");
        cy.get(":nth-child(4) > .border").type("123@test");
        cy.get(".bg-primary").contains("Login").click();
        cy.wait(2000);
      });
    });
  
```

### Test execution results with screenshots.





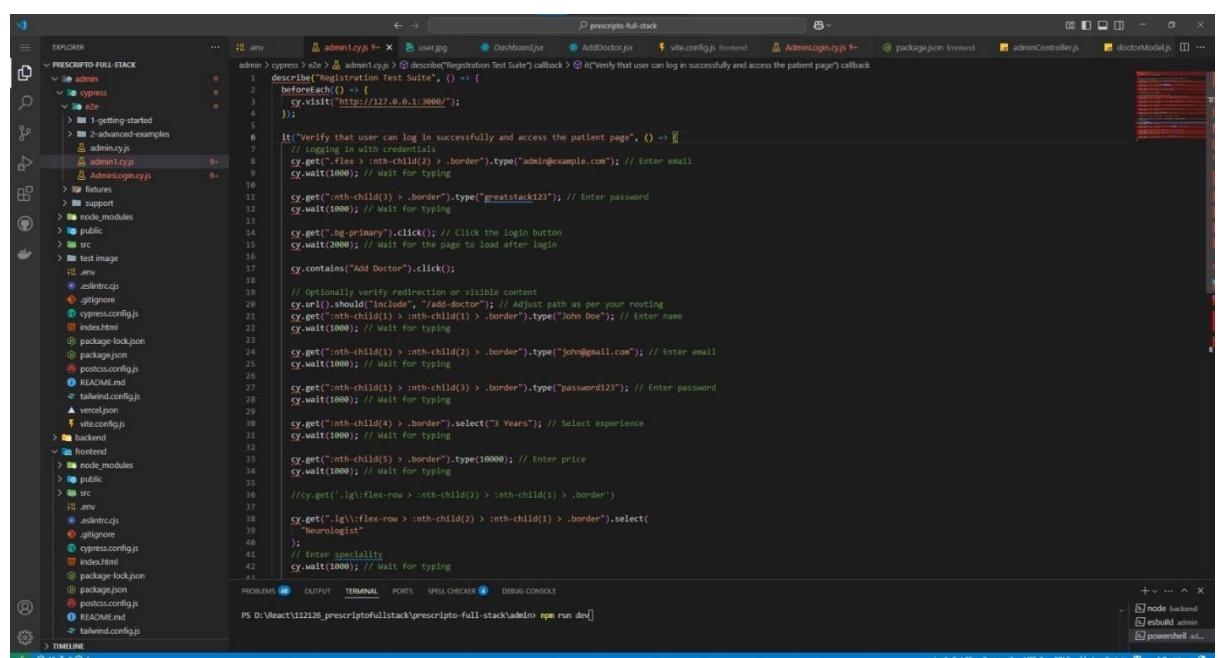


The screenshot shows a browser window with two main panes. The left pane is a Cypress test runner titled 'frontend' with the URL 'http://127.0.0.1:5173/\_/#/specs/runner?file=cypress/e2e/login.cy.js'. It displays a test script for a 'login' scenario. The right pane shows a login page for 'Prescripto' with the URL 'http://127.0.0.1:5173/'. The page has a header with the 'Prescripto' logo, 'HOME', 'ALL DOCTORS', 'ABOUT', 'CONTACT', and a 'Create account' button. Below the header is a 'Login' form with fields for 'Email' (containing 'demo44@gmail.com') and 'Password' (containing '\*\*\*\*\*'). A 'Login' button is highlighted in blue. Below the form is a link 'Create on new account? [Click here](#)'. At the bottom of the browser window, there are status indicators: 'Tracked' and 'Highlights'.

```
login.cy.js
  ✓ Login
    ✓ Navigate to My Profile from Dropdown and Edit
      ✓ Login
        ✓ BEFORE EACH
          ✓ visit http://127.0.0.1:5173/
            ✓ GET 304 http://localhost:4000/api/doctors/list
          ✓ TEST BODY
            ✓ get .gap-4 > .bg-primary
            ✓ -click
              ✓ new url: http://127.0.0.1:5173/login
            ✓ wait 1000
            ✓ get .text-primary
            ✓ -click
            ✓ wait 1000
            ✓ get .flex > :nth-child(3) > .border
            ✓ -type demo44@gmail.com
            ✓ get :nth-child(4) > .border
            ✓ -type 123456
            ✓ get .bg-primary
            ✓ -contains Login
            ✓ -click
              ✓ POST 200 http://localhost:4000/api/users/login
            ✓ wait 2000
            ✓ new url: http://127.0.0.1:5173/
            ✓ GET 200 http://localhost:4000/api/users/get-profile
```

## Add doctor records (IT21225024-Bhagya P.S)

### Automated test cases with scripts.

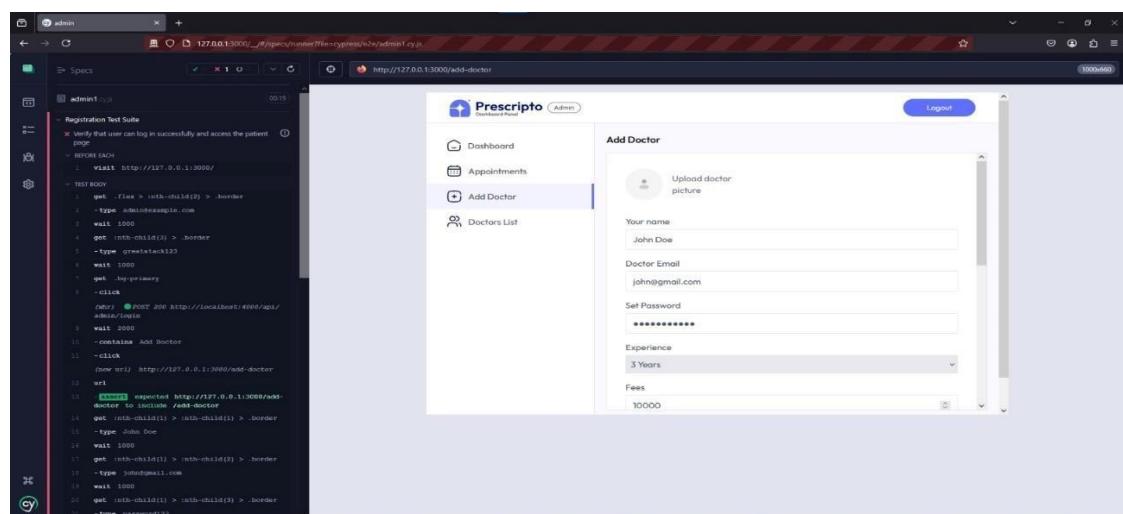


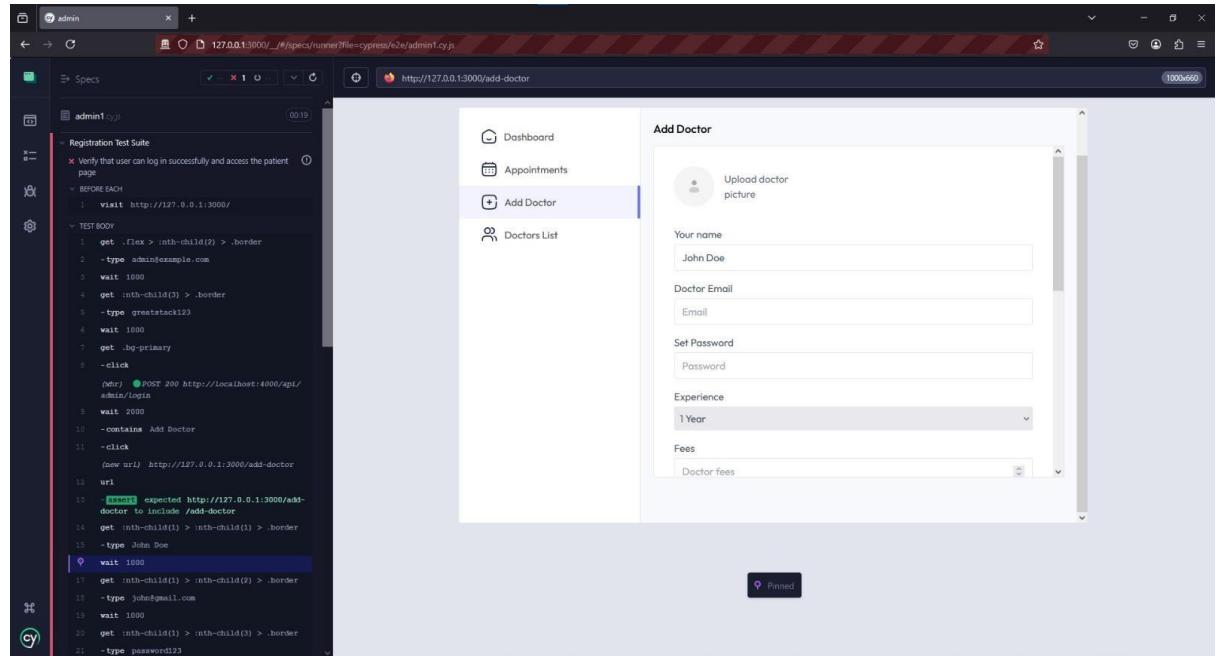
```

admin > cy.get('#username').type('admin@example.com'); // Enter email
cy.wait(1000); // Wait for typing
cy.get('#password').type('greatstack123'); // Enter password
cy.wait(1000); // Wait for typing
cy.get('#name').type('John Doe'); // Enter name
cy.wait(1000); // Wait for typing
cy.get('#email').type('john@gmail.com'); // Enter email
cy.wait(1000); // Wait for typing
cy.get('#password').type('password123'); // Enter password
cy.wait(1000); // Wait for typing
cy.get('#experience').select('3 Years'); // Select experience
cy.wait(1000); // Wait for typing
cy.get('#price').type('10000'); // Enter price
cy.wait(1000); // Wait for typing
cy.get('#speciality').select('Neurologist'); // Enter speciality
cy.wait(1000); // Wait for typing

```

### Test execution results with screenshots.





Specs

admin1.e2e

Registration Test Suite

Verify that user can log in successfully and access the patient page

BEFORE EACH

visit <http://127.0.0.1:3000/>

TEST BODY

```

1  get .flex > :nth-child(2) > .border
2  -type admin@example.com
3  wait 1000
4  get :nth-child(3) > .border
5  -type greatstack123
6  wait 1000
7  get .bg-primary
8  -click
  (either) POST 200 http://localhost:4000/api/admin/login
9  wait 2000
10 -contains Add Doctor
11 -click
  (new url) http://127.0.0.1:3000/add-doctor
12 url
13 expect http://127.0.0.1:3000/add-doctor to include /add-doctor
14 get :nth-child(1) > :nth-child(1) > .border
15 -type John Doe
16 -type John Doe
17 wait 1000
18 get :nth-child(1) > :nth-child(2) > .border
19 -type john@gmail.com
20 wait 1000
21 get :nth-child(1) > :nth-child(3) > .border
22 -type password123

```

Add Doctor

Dashboard

Appointments

Add Doctor

Doctors List

Your name

John Doe

Doctor Email

Email

Set Password

Password

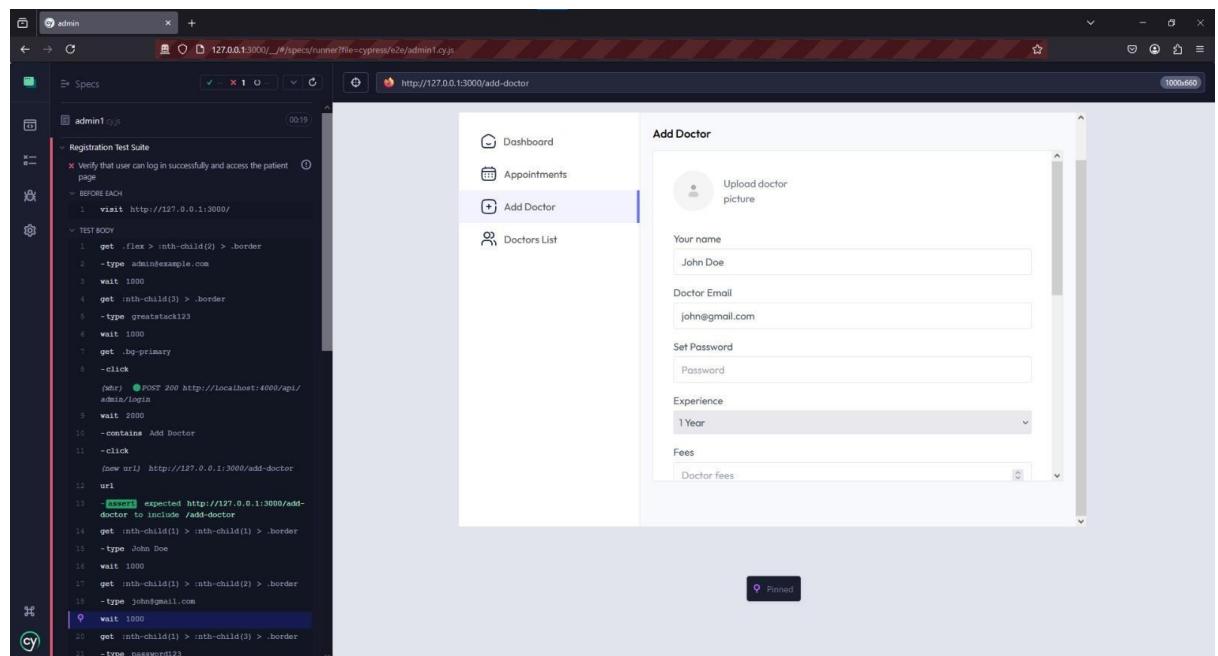
Experience

1 Year

Fees

Doctor fees

Pinned



Specs

admin1.e2e

Registration Test Suite

Verify that user can log in successfully and access the patient page

BEFORE EACH

visit <http://127.0.0.1:3000/>

TEST BODY

```

1  get .flex > :nth-child(2) > .border
2  -type admin@example.com
3  wait 1000
4  get :nth-child(3) > .border
5  -type greatstack123
6  wait 1000
7  get .bg-primary
8  -click
  (either) POST 200 http://localhost:4000/api/admin/login
9  wait 2000
10 -contains Add Doctor
11 -click
  (new url) http://127.0.0.1:3000/add-doctor
12 url
13 expect http://127.0.0.1:3000/add-doctor to include /add-doctor
14 get :nth-child(1) > :nth-child(1) > .border
15 -type John Doe
16 -type John Doe
17 wait 1000
18 get :nth-child(1) > :nth-child(2) > .border
19 -type john@gmail.com
20 wait 1000
21 get :nth-child(1) > :nth-child(3) > .border
22 -type password123

```

Add Doctor

Dashboard

Appointments

Add Doctor

Doctors List

Your name

John Doe

Doctor Email

john@gmail.com

Set Password

Password

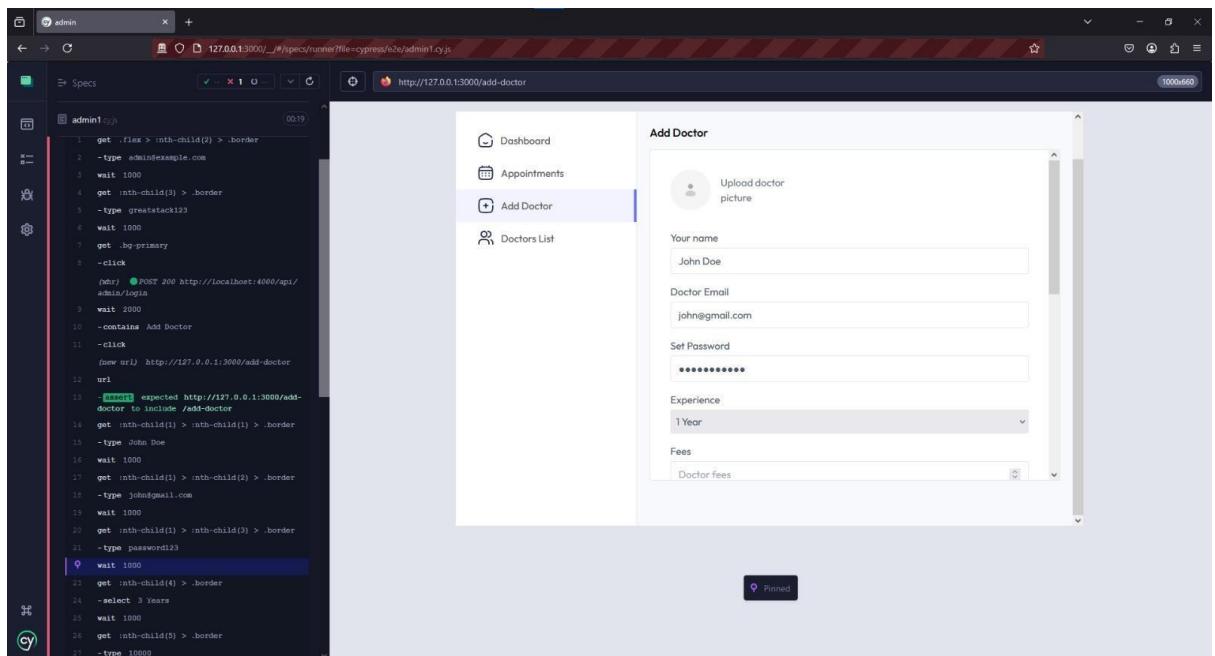
Experience

1 Year

Fees

Doctor fees

Pinned

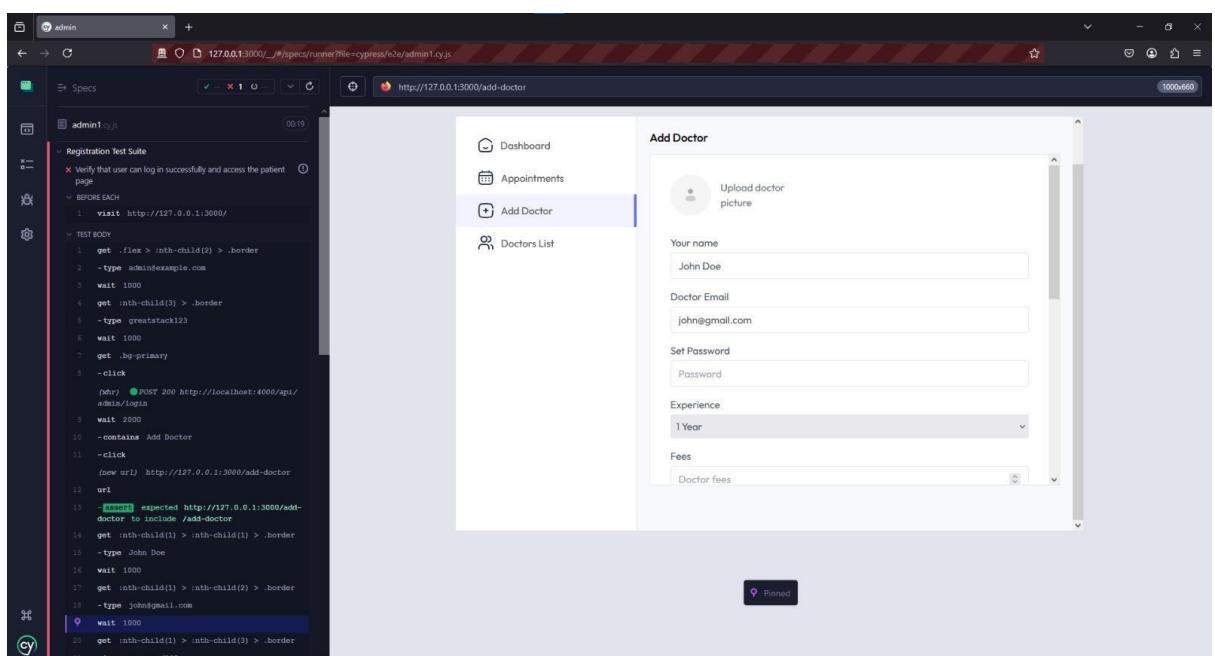


admin1.cy.js

```

1  get '.flex > :nth-child(2) > .border'
2  -type adminexample.com
3  wait 1000
4  get ':nth-child(3) > .border'
5  -type greatstack123
6  wait 1000
7  get '.bg-primary'
8  -click
9  (her) POST 200 http://localhost:4000/api/admin/login
10 wait 2000
11 -contains Add Doctor
12 -click
13 (new url) http://127.0.0.1:3000/add-doctor
14 url
15 -assert expected http://127.0.0.1:3000/add-doctor to include /add-doctor
16 get ':nth-child(1) > :nth-child(1) > .border'
17 -type John Doe
18 wait 1000
19 get ':nth-child(1) > :nth-child(2) > .border'
20 -type john@gmail.com
21 wait 1000
22 get ':nth-child(1) > :nth-child(3) > .border'
23 -type password123
24 wait 1000
25 get ':nth-child(4) > .border'
26 -select 3 Years
27 wait 1000
28 get ':nth-child(5) > .border'
29 -type 10000
30

```



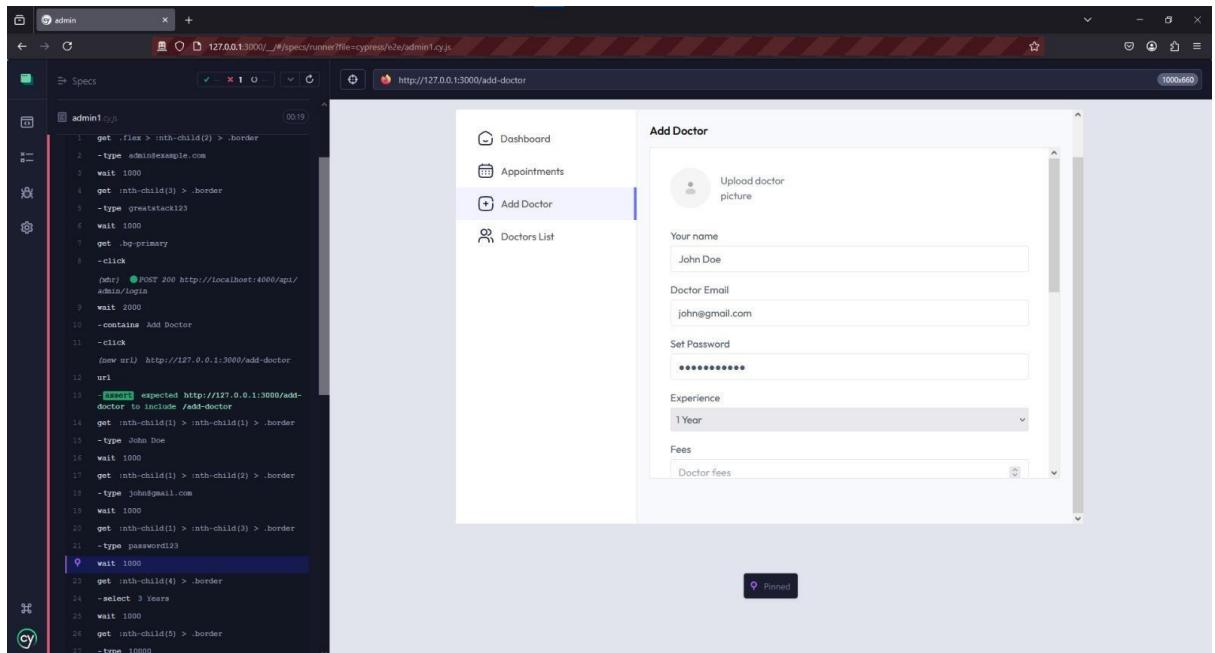
Registration Test Suite

- Verify that user can log in successfully and access the patient page

```

1  BEFORE EACH
2  visit http://127.0.0.1:3000/
3
4  TEST BODY
5  get '.flex > :nth-child(2) > .border'
6  -type adminexample.com
7  wait 1000
8  get ':nth-child(3) > .border'
9  -type greatstack123
10 wait 1000
11 get '.bg-primary'
12 -click
13 (her) POST 200 http://localhost:4000/api/admin/login
14 wait 2000
15 -contains Add Doctor
16 -click
17 (new url) http://127.0.0.1:3000/add-doctor
18 url
19 -assert expected http://127.0.0.1:3000/add-doctor to include /add-doctor
20 get ':nth-child(1) > :nth-child(1) > .border'
21 -type John Doe
22 wait 1000
23 get ':nth-child(1) > :nth-child(2) > .border'
24 -type john@gmail.com
25 wait 1000
26 get ':nth-child(1) > :nth-child(3) > .border'
27 -type password123
28 wait 1000
29

```

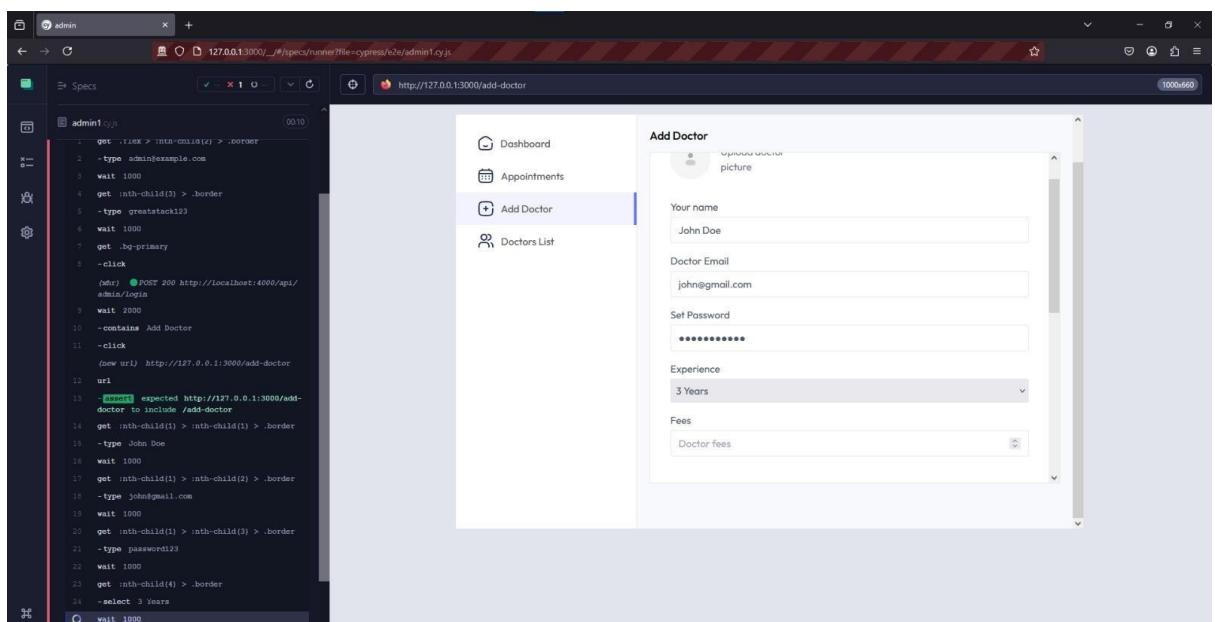


admin1.cy.js

```

1  get '.flex > :nth-child(2) > .border'
2  -type admin@example.com
3  wait 1000
4  get :nth-child(3) > .border
5  -type greatstack123
6  wait 1000
7  get .bg-primary
8  -click
9  (xhr) POST 200 http://localhost:4000/api/
10 admin/login
11 wait 2000
12 -contains Add Doctor
13 -click
14 (new url) http://127.0.0.1:3000/add-doctor
15 url
16 +assert expected http://127.0.0.1:3000/add-doctor to include /add-doctor
17 get :nth-child(1) > :nth-child(1) > .border
18 -type John Doe
19 wait 1000
20 get :nth-child(1) > :nth-child(2) > .border
21 -type john@gmail.com
22 wait 1000
23 get :nth-child(1) > :nth-child(3) > .border
24 -type password123
25 wait 1000
26 get :nth-child(4) > .border
27 -select 3 years
28 wait 1000
29 get :nth-child(5) > .border
30 -type 10000
31 wait 1000

```

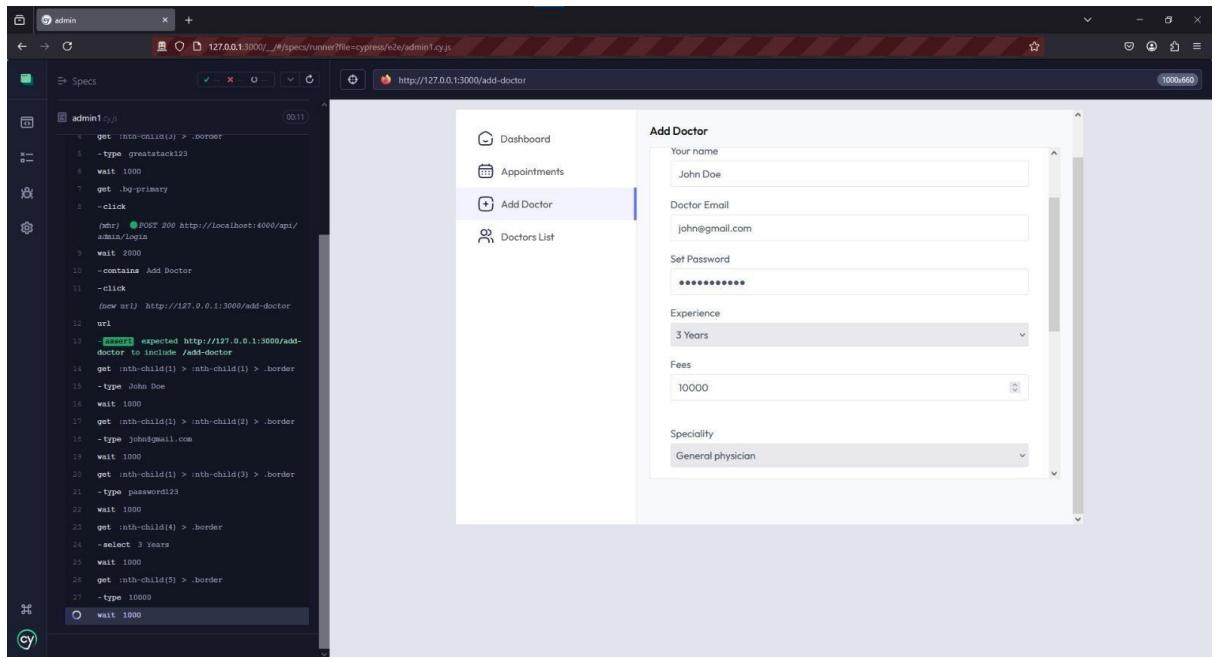


admin1.cy.js

```

1  get '.flex > :nth-child(2) > .border'
2  -type admin@example.com
3  wait 1000
4  get :nth-child(3) > .border
5  -type greatstack123
6  wait 1000
7  get .bg-primary
8  -click
9  (xhr) POST 200 http://localhost:4000/api/
10 admin/login
11 wait 2000
12 -contains Add Doctor
13 -click
14 (new url) http://127.0.0.1:3000/add-doctor
15 url
16 +assert expected http://127.0.0.1:3000/add-doctor to include /add-doctor
17 get :nth-child(1) > :nth-child(1) > .border
18 -type John Doe
19 wait 1000
20 get :nth-child(1) > :nth-child(2) > .border
21 -type john@gmail.com
22 wait 1000
23 get :nth-child(1) > :nth-child(3) > .border
24 -type password123
25 wait 1000
26 get :nth-child(4) > .border
27 -select 3 years
28 wait 1000
29 get :nth-child(5) > .border
30 -type 10000
31 wait 1000

```



admin1.cy.js

```

1  get '#sign-up-form > .border'
2  -type greatstack123
3  wait 1000
4  get .bg-primary
5  -click
6  (xhr) POST 200 http://localhost:4000/api/admin/login
7  wait 2000
8  -contains Add Doctor
9  -click
10 (new url) http://127.0.0.1:3000/add-doctor
11 url
12 -assert expected http://127.0.0.1:3000/add-doctor to include /add-doctor
13 get :nth-child(1) > :nth-child(1) > .border
14 -type John Doe
15 wait 1000
16 get :nth-child(1) > :nth-child(2) > .border
17 -type john@gmail.com
18 wait 1000
19 get :nth-child(1) > :nth-child(3) > .border
20 -type password123
21 wait 1000
22 get :nth-child(4) > .border
23 -select 3 years
24 wait 1000
25 get :nth-child(5) > .border
26 -type 10000
27 wait 1000

```

Add Doctor

Your name: John Doe

Doctor Email: john@gmail.com

Set Password:

Experience: 3 Years

Fees: 10000

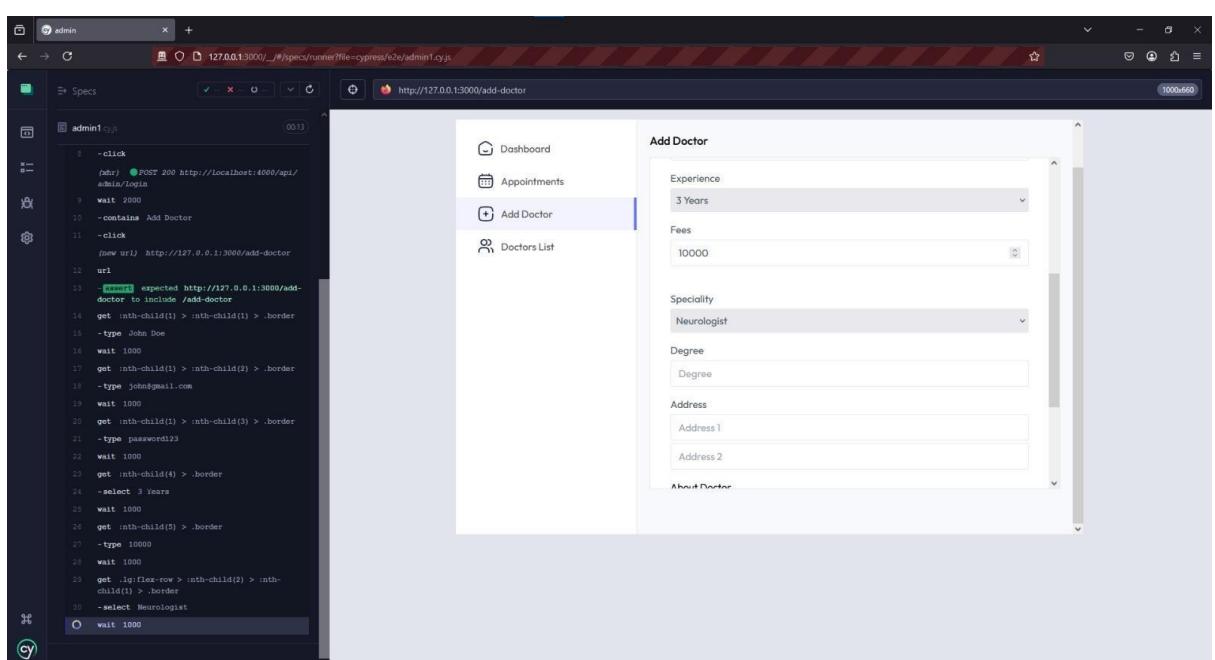
Speciality: General physician

Dashboard

Appointments

Add Doctor

Doctors List



admin1.cy.js

```

1  -click
2  (xhr) POST 200 http://localhost:4000/api/admin/login
3  wait 2000
4  -contains Add Doctor
5  -click
6  (new url) http://127.0.0.1:3000/add-doctor
7  url
8  -assert expected http://127.0.0.1:3000/add-doctor to include /add-doctor
9  get :nth-child(1) > :nth-child(1) > .border
10 -type John Doe
11 wait 1000
12 get :nth-child(1) > :nth-child(2) > .border
13 -type john@gmail.com
14 wait 1000
15 get :nth-child(1) > :nth-child(3) > .border
16 -type password123
17 wait 1000
18 get :nth-child(4) > .border
19 -select 3 years
20 wait 1000
21 get :nth-child(5) > .border
22 -type 10000
23 wait 1000
24 get :flex-row > :nth-child(2) > :nth-child(1) > .border
25 -select Neurologist
26 wait 1000

```

Add Doctor

Experience: 3 Years

Fees: 10000

Speciality: Neurologist

Degree:

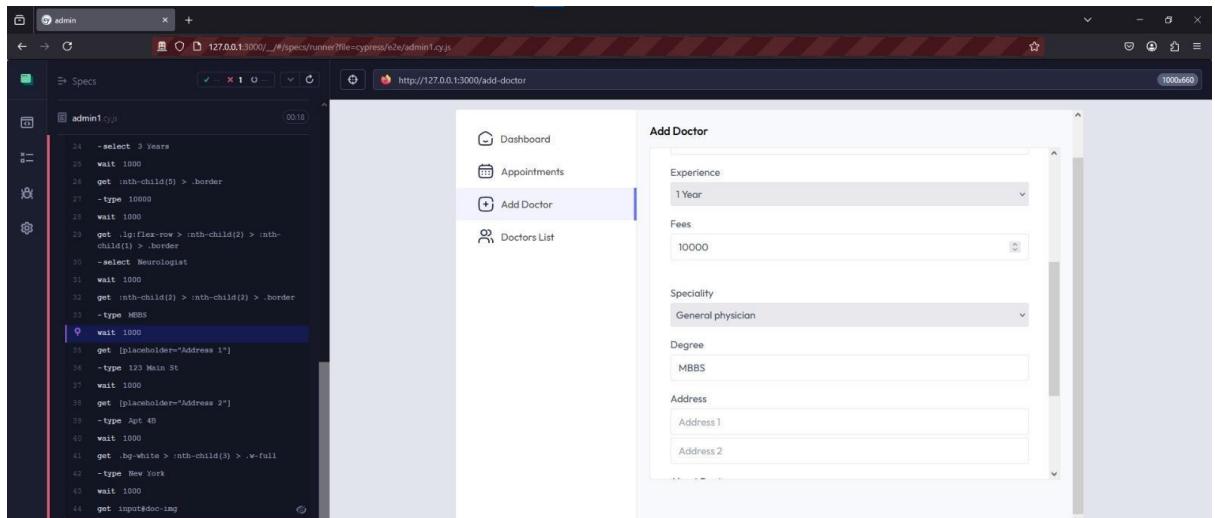
Address: Address 1  
Address 2

Dashboard

Appointments

Add Doctor

Doctors List



Specs

admin1.cy.js

```

24 - select 3 Years
25 wait 1000
26 get :nth-child(3) > .border
27 -type 10000
28 wait 1000
29 get :lg:flex-row > :nth-child(2) > :nth-child(1) > .border
30 -select Neurologist
31 wait 1000
32 get :nth-child(2) > :nth-child(2) > .border
33 -type MBBS
34 wait 1000
35 get [placeholder="Address 1"]
36 -type 123 Main St
37 wait 1000
38 get [placeholder="Address 2"]
39 -type Apt 4B
40 wait 1000
41 get :bg-white > :nth-child(3) > .w-full
42 -type New York
43 wait 1000
44 get input#doc-img

```

Add Doctor

Experience: 1 Year

Fees: 10000

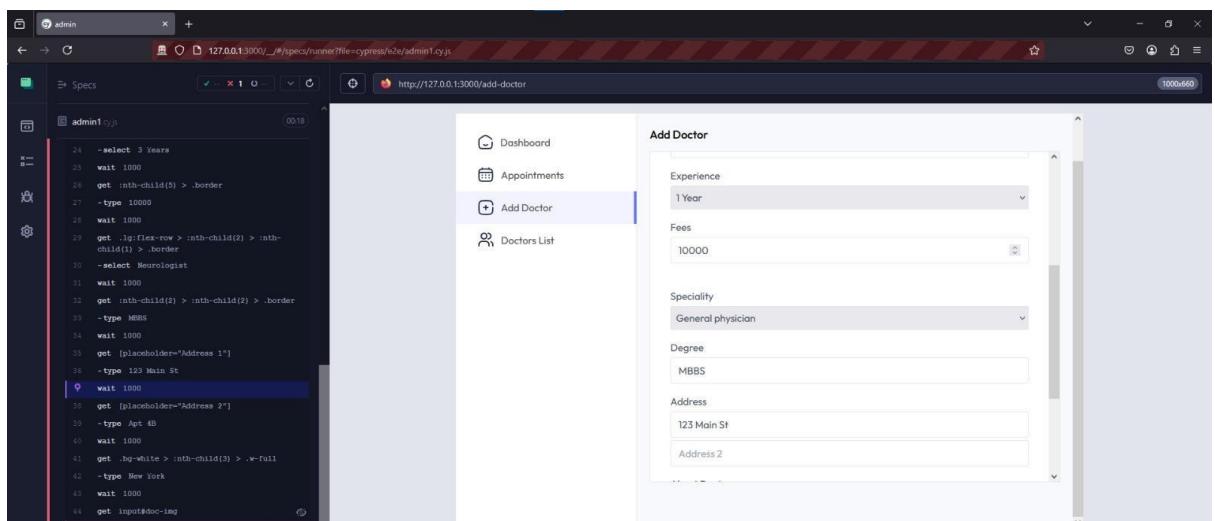
Specialty: General physician

Degree: MBBS

Address:

Address 1: 123 Main St

Address 2: Apt 4B, New York



Specs

admin1.cy.js

```

24 - select 3 Years
25 wait 1000
26 get :nth-child(3) > .border
27 -type 10000
28 wait 1000
29 get :lg:flex-row > :nth-child(2) > :nth-child(1) > .border
30 -select Neurologist
31 wait 1000
32 get :nth-child(2) > :nth-child(2) > .border
33 -type MBBS
34 wait 1000
35 get [placeholder="Address 1"]
36 -type 123 Main St
37 | wait 1000
38 | get [placeholder="Address 2"]
39 | -type Apt 4B
40 | wait 1000
41 | get :bg-white > :nth-child(3) > .w-full
42 | -type New York
43 | wait 1000
44 | get input#doc-img

```

Add Doctor

Experience: 1 Year

Fees: 10000

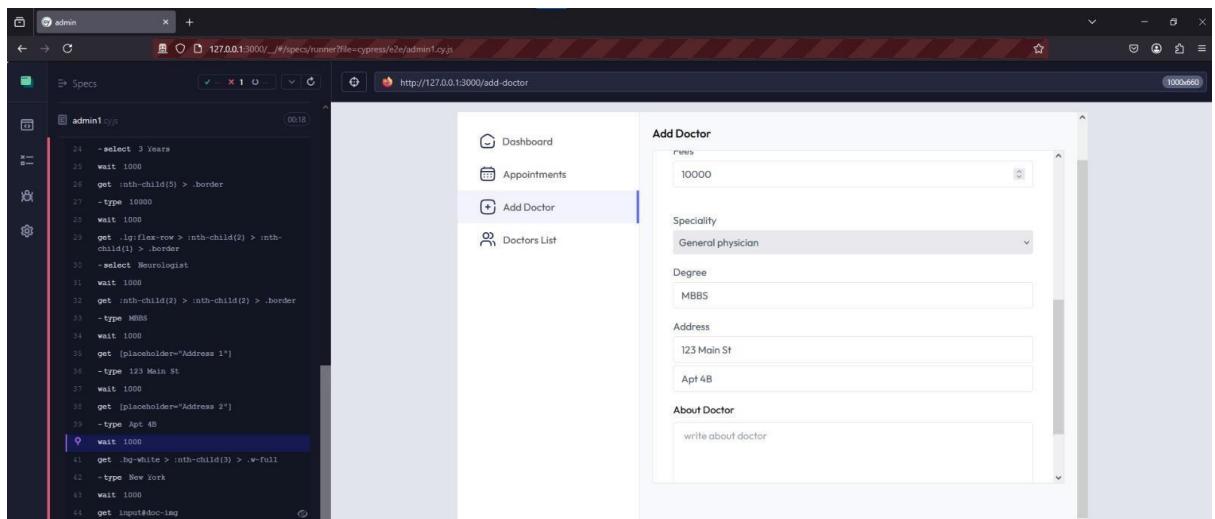
Specialty: General physician

Degree: MBBS

Address:

Address 1: 123 Main St

Address 2: Apt 4B, New York



Specs

admin1.cy.js

```

24 - select 3 Years
25 wait 1000
26 get :nth-child(3) > .border
27 -type 10000
28 wait 1000
29 get :lg:flex-row > :nth-child(2) > :nth-child(1) > .border
30 -select Neurologist
31 wait 1000
32 get :nth-child(2) > :nth-child(2) > .border
33 -type MBBS
34 wait 1000
35 get [placeholder="Address 1"]
36 -type 123 Main St
37 wait 1000
38 get [placeholder="Address 2"]
39 -type Apt 4B
40 | wait 1000
41 | get :bg-white > :nth-child(3) > .w-full
42 | -type New York
43 | wait 1000
44 | get input#doc-img

```

Add Doctor

Fees: 10000

Specialty: General physician

Degree: MBBS

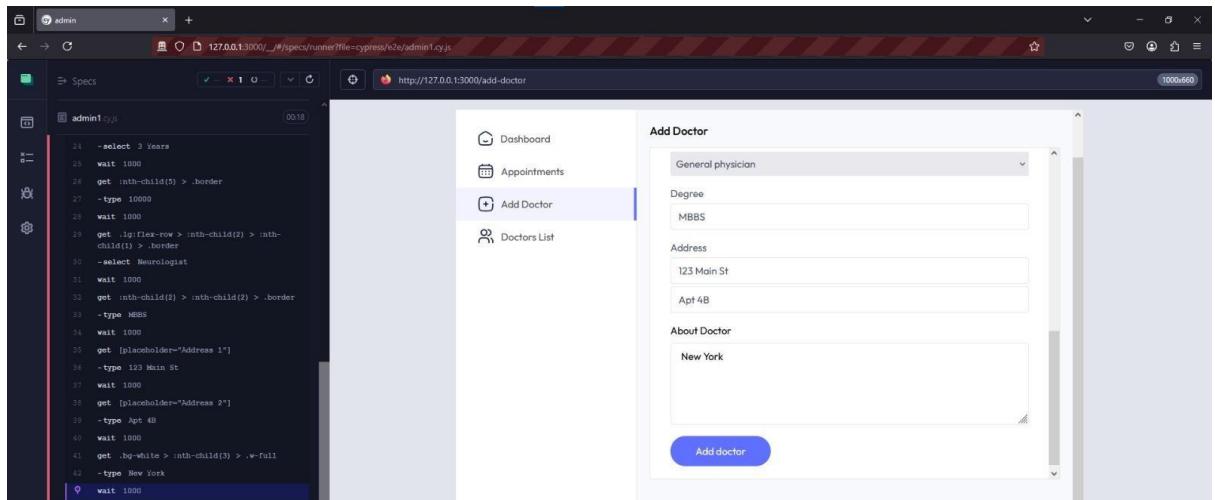
Address:

Address 1: 123 Main St

Address 2: Apt 4B, New York

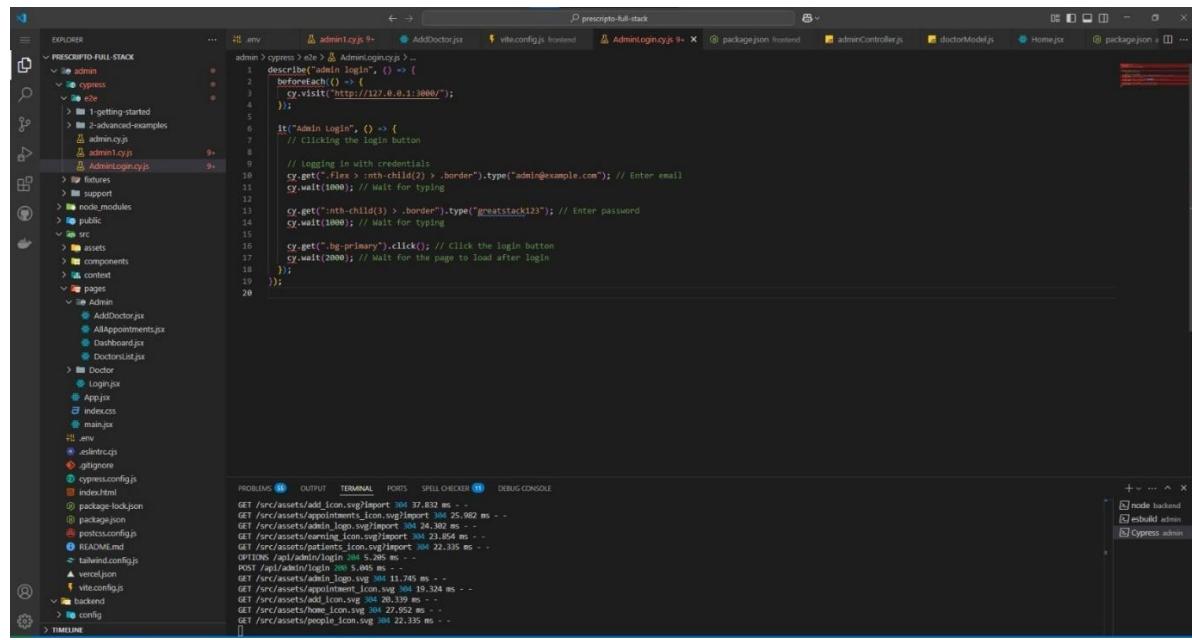
About Doctor:

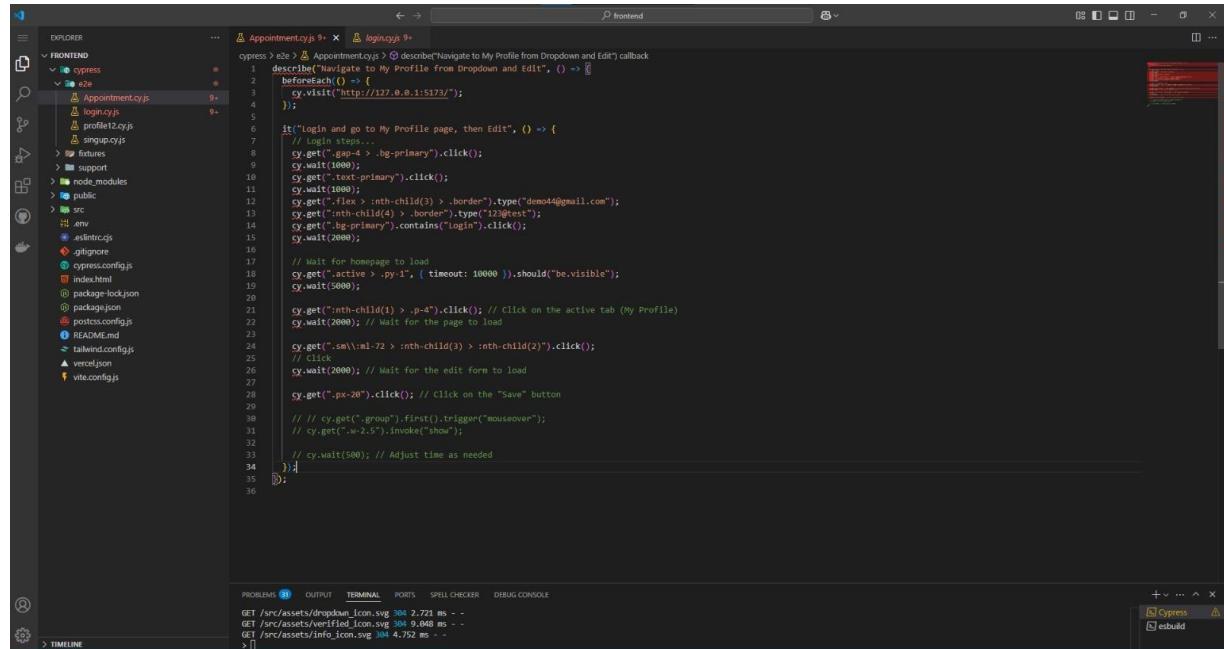
write about doctor



## Admin panel Logging and booking doctor (IT21228094-Mendis A.R.P)

### Automated test cases with scripts.





```

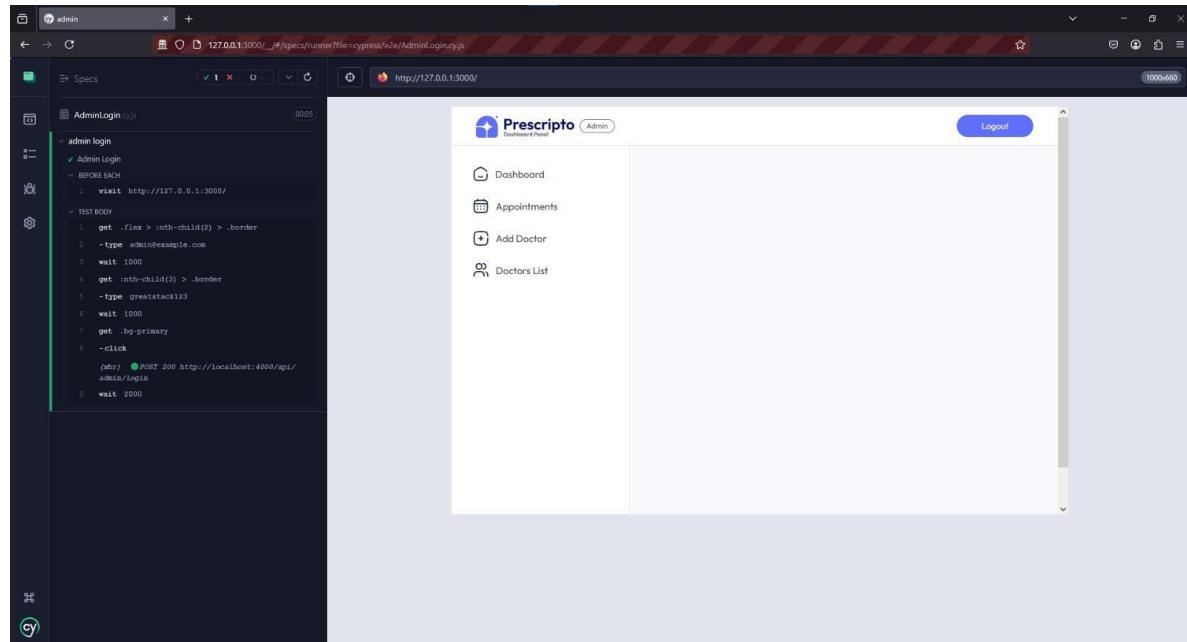
cypress > e2e > Appointment.cy.js > login.cy.js
1 describe("Navigate to My Profile from Dropdown and Edit") {
2   beforeEach(() => {
3     cy.visit("http://127.0.0.1:5173/");
4   });
5
6   it("Login and go to My Profile page, then Edit", () => {
7     // Login steps...
8     cy.get(".bg-primary").click();
9     cy.wait(1000);
10    cy.get(".bg-primary").click();
11    cy.wait(1000);
12    cy.get("flex > nth-child(3) > .border").type("demo4@gmail.com");
13    cy.get("nth-child(4) > .border").type("123@est");
14    cy.get(".bg-primary").contains("Login").click();
15    cy.wait(2000);
16
17    // Wait for homepage to load
18    cy.get("active > p-3", { timeout: 10000 }).should("be.visible");
19    cy.wait(5000);
20
21    cy.get("nth-child(1) > .p-4").click(); // Click on the active tab (My Profile)
22    cy.wait(2000); // Wait for the page to load
23
24    cy.get("sm\:ml-72 > :nth-child(3) > :nth-child(2)").click();
25    // Click
26    cy.wait(2000); // Wait for the edit form to load
27
28    cy.get("px-20").click(); // Click on the "Save" button
29
30    // cy.get(".group").first().trigger("mouseover");
31    // cy.get(".w-2.5").invoke("show");
32
33    // cy.wait(500); // Adjust time as needed
34  });
35}

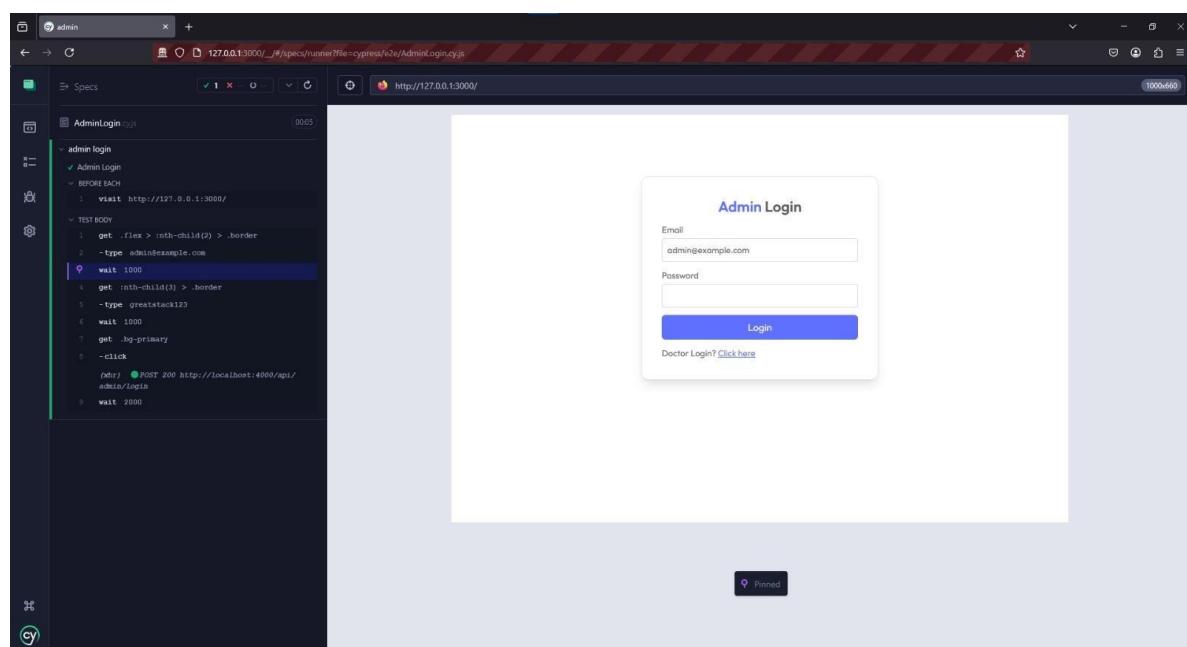
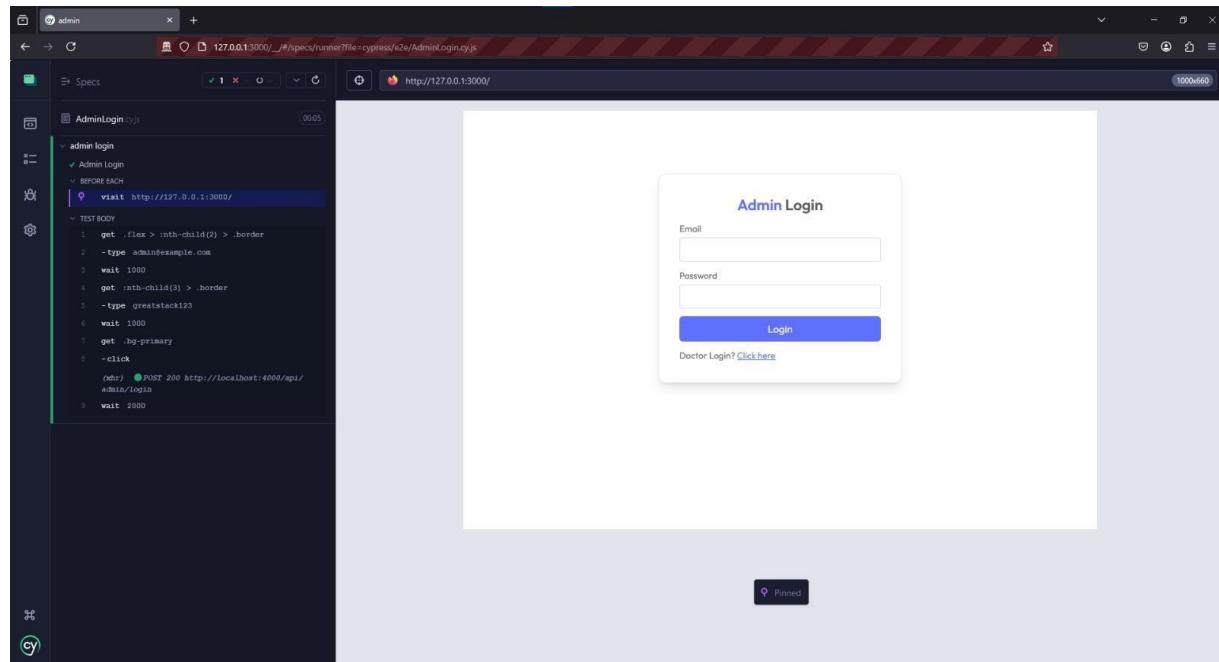
```

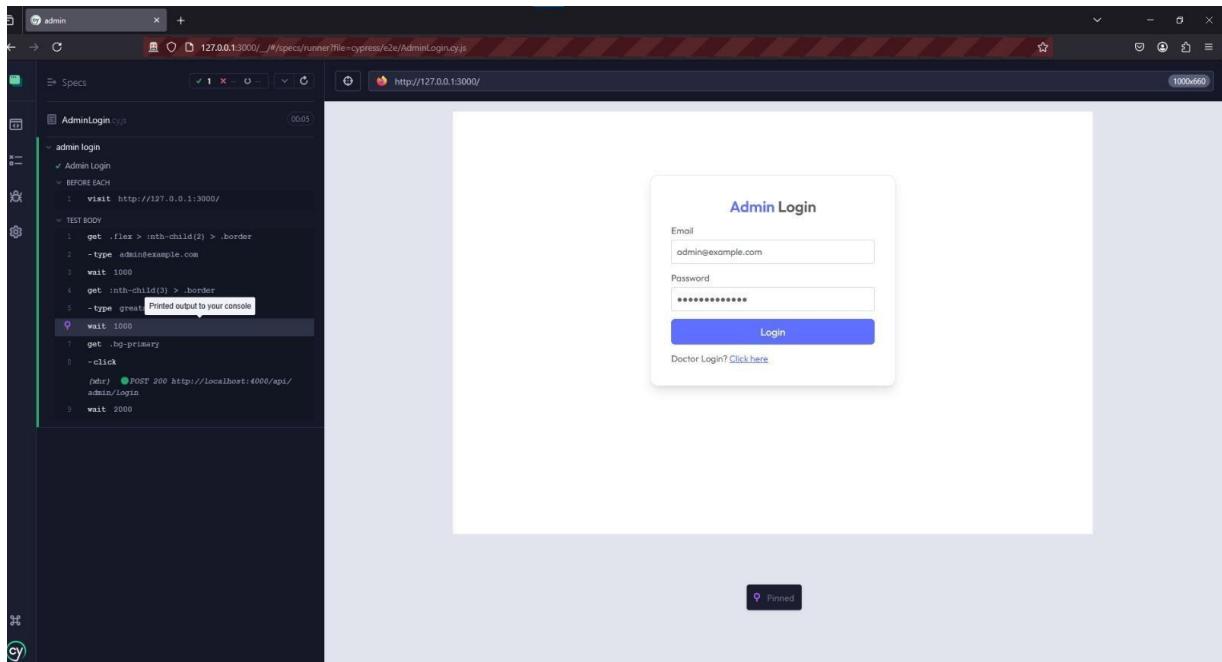
PROBLEMS OUTPUT TERMINAL PORTS SPELL CHECKER DEBUG CONSOLE

GET /src/assets/dropdown.icon.svg 304 2.727 ms - -
GET /src/assets/verified.icon.svg 304 9.049 ms - -
GET /src/assets/info.icon.svg 304 4.752 ms - -
> [ ]

## Test execution results with screenshots







Admin Login

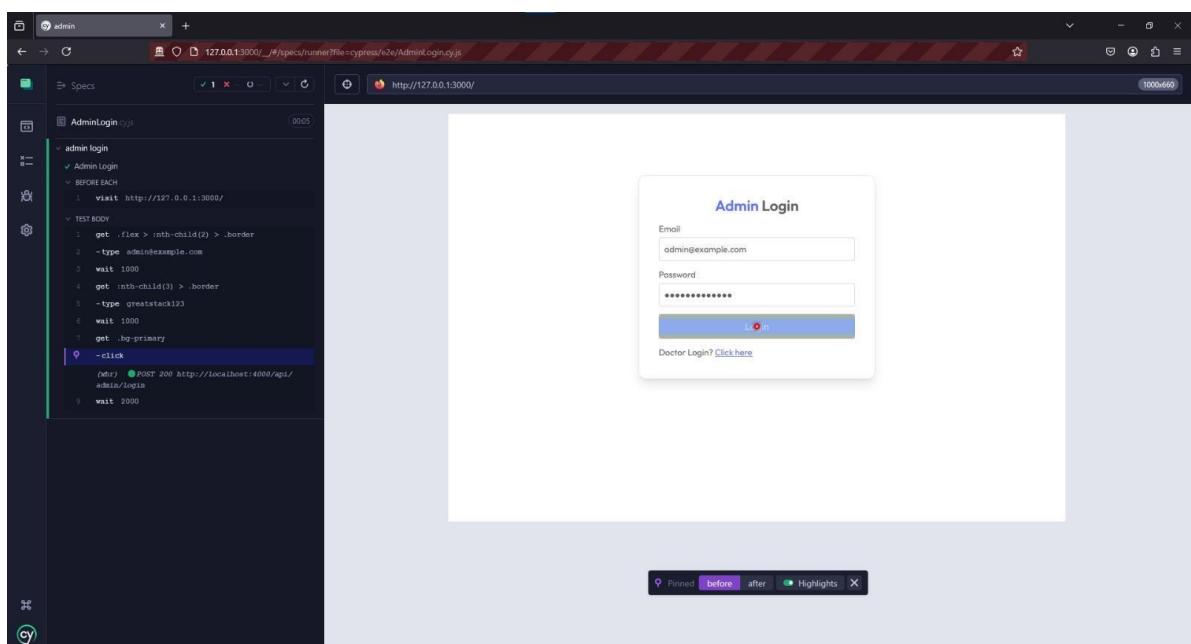
Email: admin@example.com

Password:

Login

Doctor Login? [Click here](#)

Pinned



Admin Login

Email: admin@example.com

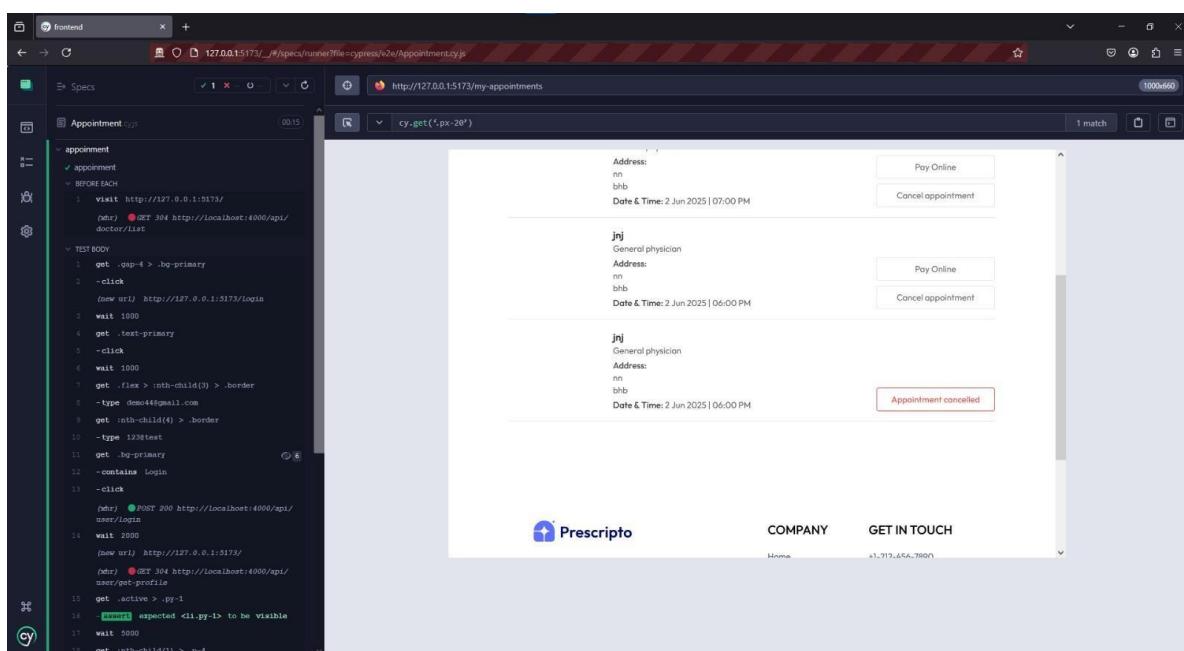
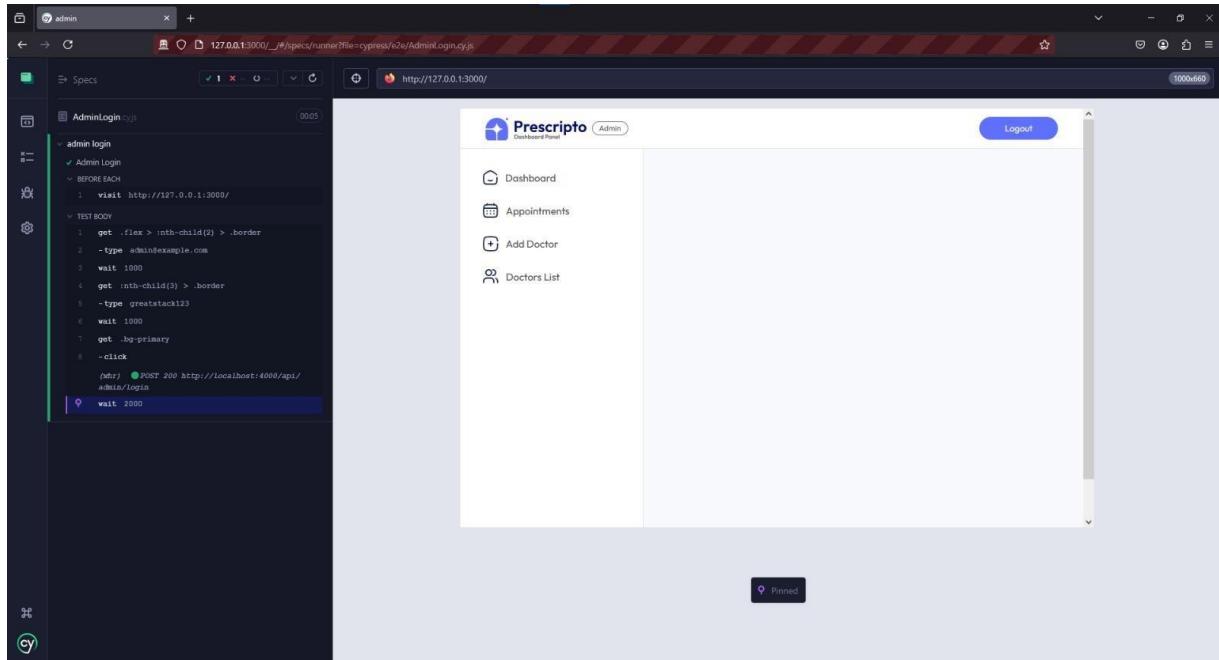
Password:

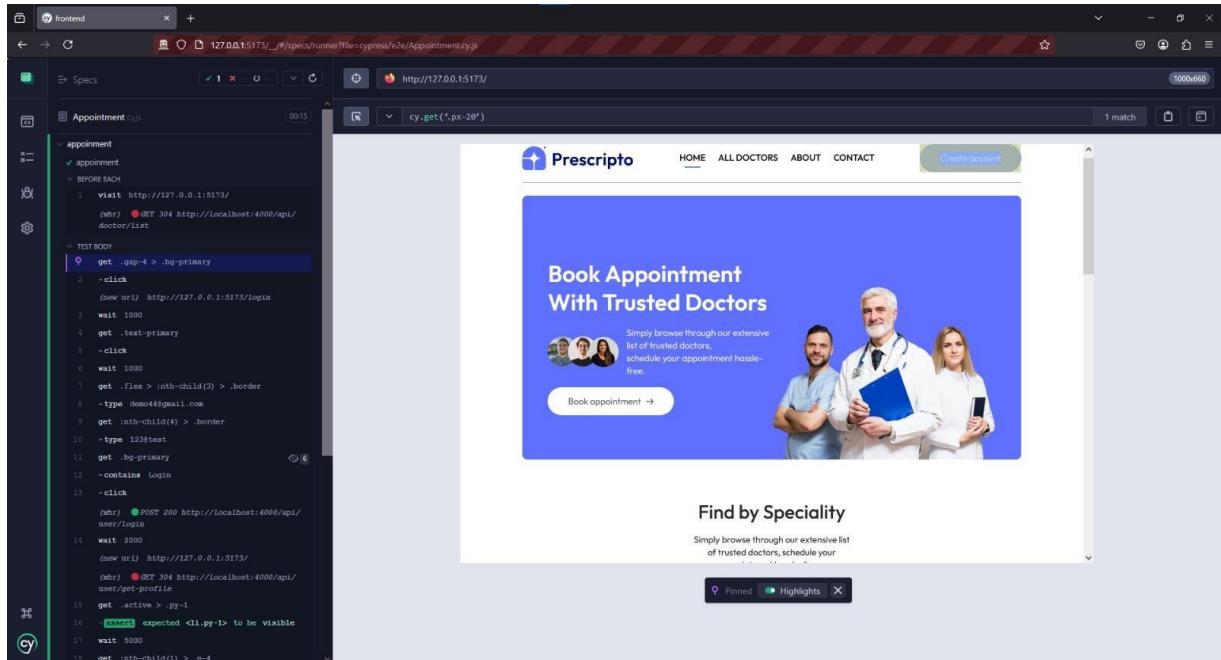
Login

Doctor Login? [Click here](#)

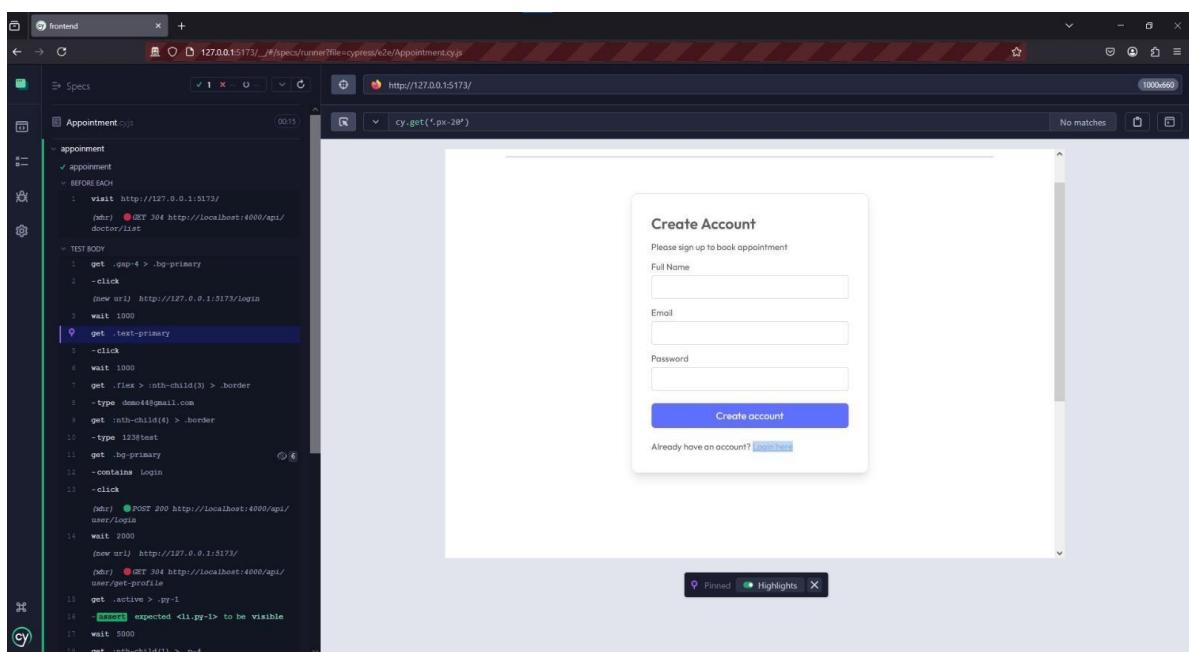
Pinned before after Highlights X

## IT4100 – SQA

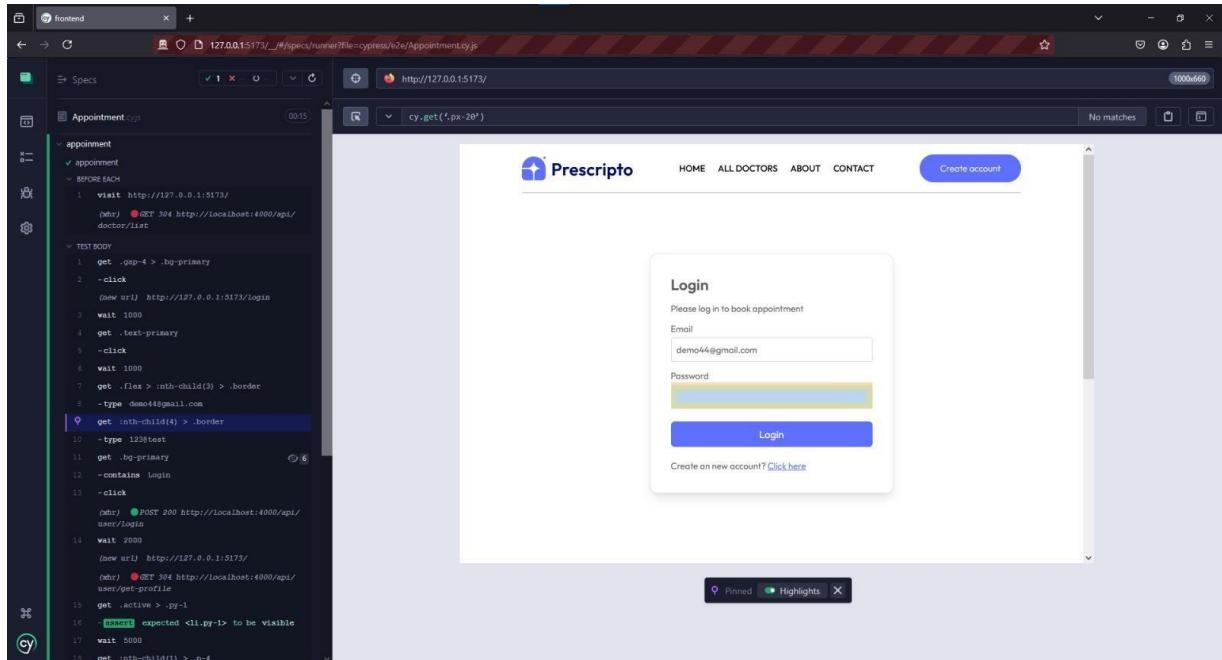




The screenshot shows a Cypress test runner interface on the left and a browser window on the right. The browser displays the 'Prescripto' website with a blue header and a 'Book Appointment With Trusted Doctors' section. The test runner shows a test file 'Appointment.cy.js' with a 'TEST BODY' block containing 18 lines of Cypress commands. The browser window shows the test results with 1 match found.



The screenshot shows a Cypress test runner interface on the left and a browser window on the right. The browser displays the 'Prescripto' website with a 'Create Account' form. The test runner shows a test file 'Appointment.cy.js' with a 'TEST BODY' block containing 18 lines of Cypress commands. The browser window shows the test results with 0 matches found.



Specs

Appointment.cy.js

```

appointment
  appointment
    BEFORE EACH
      1 visit http://127.0.0.1:5173/
      (new url) GET 304 http://localhost:4000/api/doctor/list
      2 click .text-primary
      (new url) http://127.0.0.1:5173/Login
      3 wait 1000
      4 get .text-primary
      5 click
      6 wait 1000
      7 get .flex > :nth-child(3) > .border
      8 type demo44@gmail.com
      9 get :nth-child(4) > .border
      10 type 123456
      11 get .bg-primary
      12 contains Login
      13 click
      (new url) POST 200 http://localhost:4000/api/user/Login
      14 wait 2000
      (new url) http://127.0.0.1:5173/
      (new url) GET 304 http://localhost:4000/api/user/get-profile
      15 get :active > _y-1
      16 expect <li>.py-1 to be visible
      17 wait 5000
      18 get :nth-child(1) > _y-4
  
```

Prescripto

HOME ALL DOCTORS ABOUT CONTACT Create account

Login

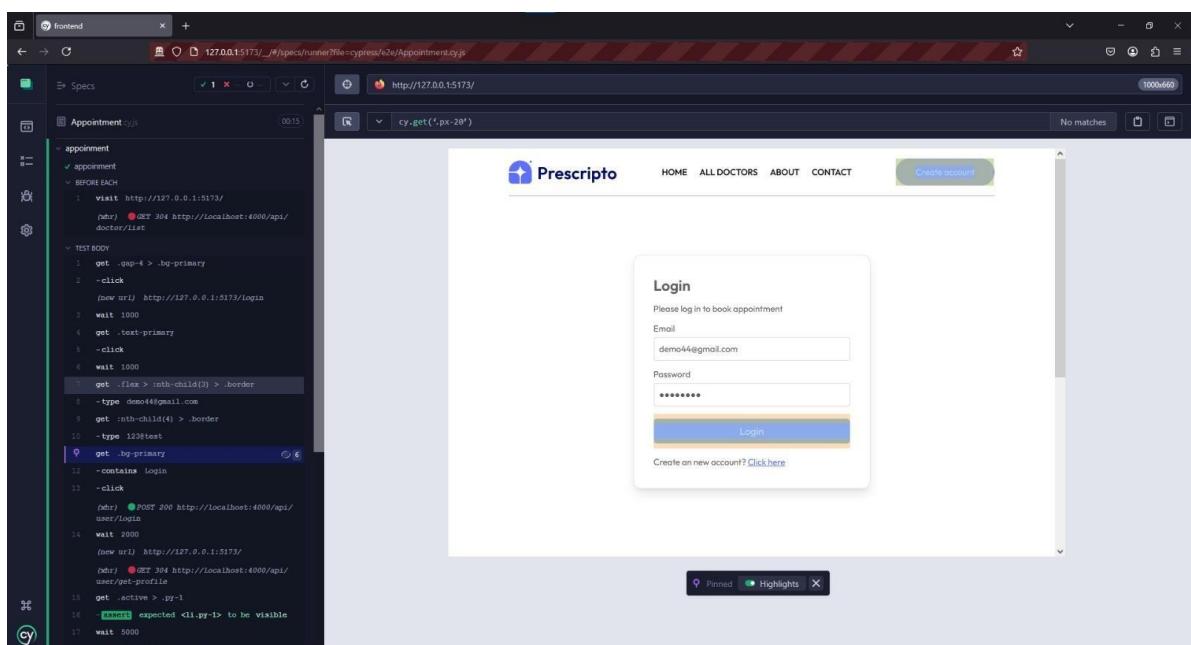
Please log in to book appointment

Email: demo44@gmail.com

Password:

Login

Create an new account? [Click here](#)



Specs

Appointment.cy.js

```

appointment
  appointment
    BEFORE EACH
      1 visit http://127.0.0.1:5173/
      (new url) GET 304 http://localhost:4000/api/doctor/list
      2 click .text-primary
      (new url) http://127.0.0.1:5173/Login
      3 wait 1000
      4 get .text-primary
      5 click
      6 wait 1000
      7 get .flex > :nth-child(3) > .border
      8 type demo44@gmail.com
      9 get :nth-child(4) > .border
      10 type 123456
      11 get .bg-primary
      12 contains Login
      13 click
      (new url) POST 200 http://localhost:4000/api/user/Login
      14 wait 2000
      (new url) http://127.0.0.1:5173/
      (new url) GET 304 http://localhost:4000/api/user/get-profile
      15 get :active > _y-1
      16 expect <li>.py-1 to be visible
      17 wait 5000
      18 get :nth-child(1) > _y-4
  
```

Prescripto

HOME ALL DOCTORS ABOUT CONTACT Create account

Login

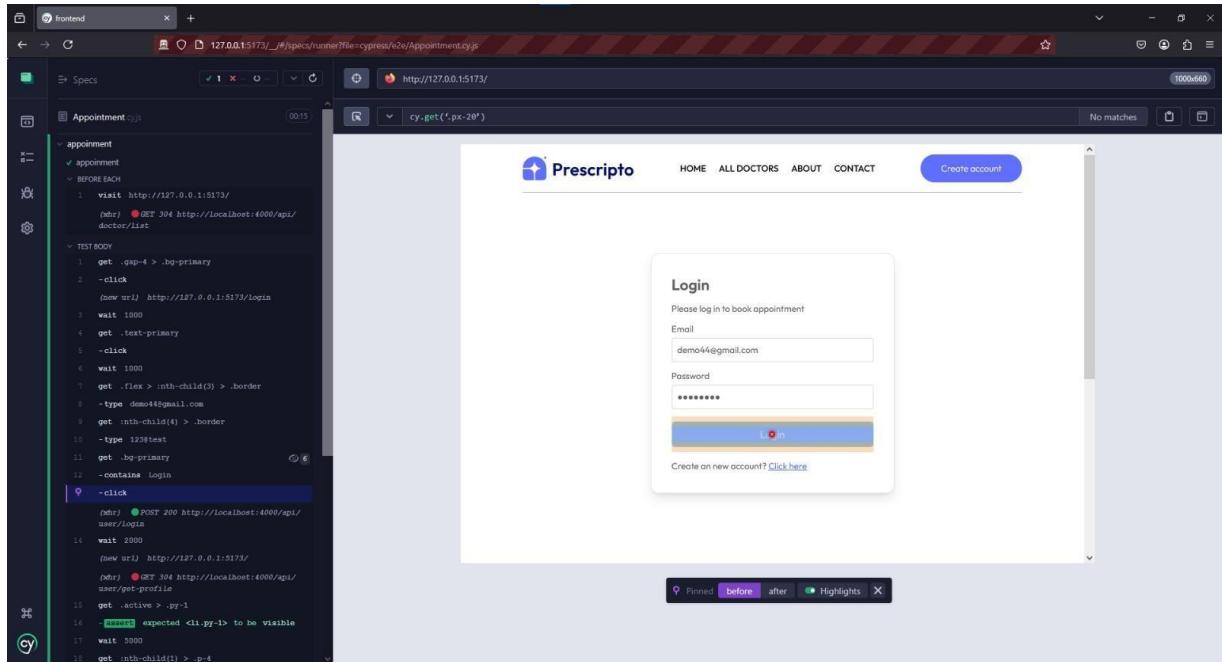
Please log in to book appointment

Email: demo44@gmail.com

Password:

Login

Create an new account? [Click here](#)



frontend

Specs

Appointment.cy.js

```

appointment
  appointment
    BEFORE EACH
      1 visit http://127.0.0.1:5173/
      (bdr) ① GET 304 http://localhost:4000/api/doctor/list

    TEST BODY
      1 get .gap-4 > .bg-primary
      2 -click
      (new url) http://127.0.0.1:5173/Login
      3 wait 1000
      4 get .text-primary
      5 -click
      6 wait 1000
      7 get .flex > :nth-child(3) > .border
      8 -type demo44@gmail.com
      9 get :nth-child(4) > .border
      10 -type 1234test
      11 get .bg-primary
      12 -contains Login
      13 -click
      (bdr) ② POST 200 http://localhost:4000/api/user/Login
      14 wait 2000
      (new url) http://127.0.0.1:5173/
      (bdr) ③ GET 304 http://localhost:4000/api/user/get-profile
      15 get .active > .py-1
      16 -assert expected <li.py-1> to be visible
      17 wait 5000
      18 get :nth-child(1) > .p-4
  
```

Prescripto

HOME ALL DOCTORS ABOUT CONTACT Create account

Login

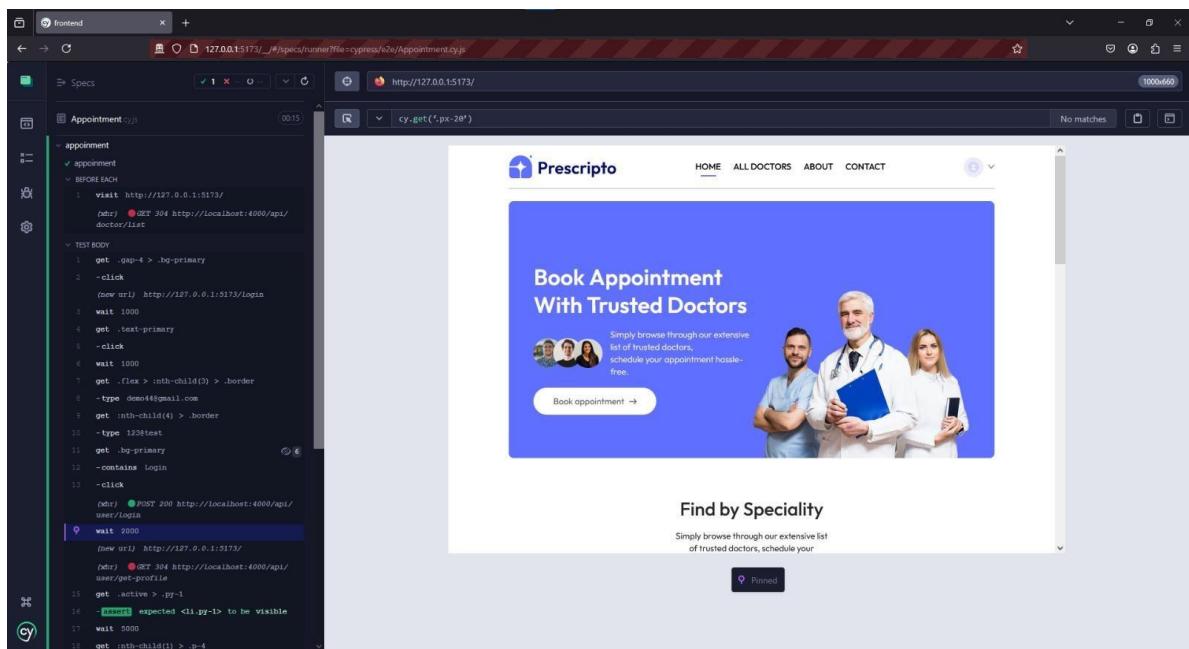
Please log in to book appointment

Email: demo44@gmail.com

Password: \*\*\*\*\*

Login

Create an new account? [Click here](#)



frontend

Specs

Appointment.cy.js

```

appointment
  appointment
    BEFORE EACH
      1 visit http://127.0.0.1:5173/
      (bdr) ① GET 304 http://localhost:4000/api/doctor/list

    TEST BODY
      1 get .gap-4 > .bg-primary
      2 -click
      (new url) http://127.0.0.1:5173/Login
      3 wait 1000
      4 get .text-primary
      5 -click
      6 wait 1000
      7 get .flex > :nth-child(3) > .border
      8 -type demo44@gmail.com
      9 get :nth-child(4) > .border
      10 -type 1234test
      11 get .bg-primary
      12 -contains Login
      13 -click
      (bdr) ② POST 200 http://localhost:4000/api/user/Login
      14 wait 2000
      (new url) http://127.0.0.1:5173/
      (bdr) ③ GET 304 http://localhost:4000/api/user/get-profile
      15 get .active > .py-1
      16 -assert expected <li.py-1> to be visible
      17 wait 5000
      18 get :nth-child(1) > .p-4
  
```

Prescripto

HOME ALL DOCTORS ABOUT CONTACT

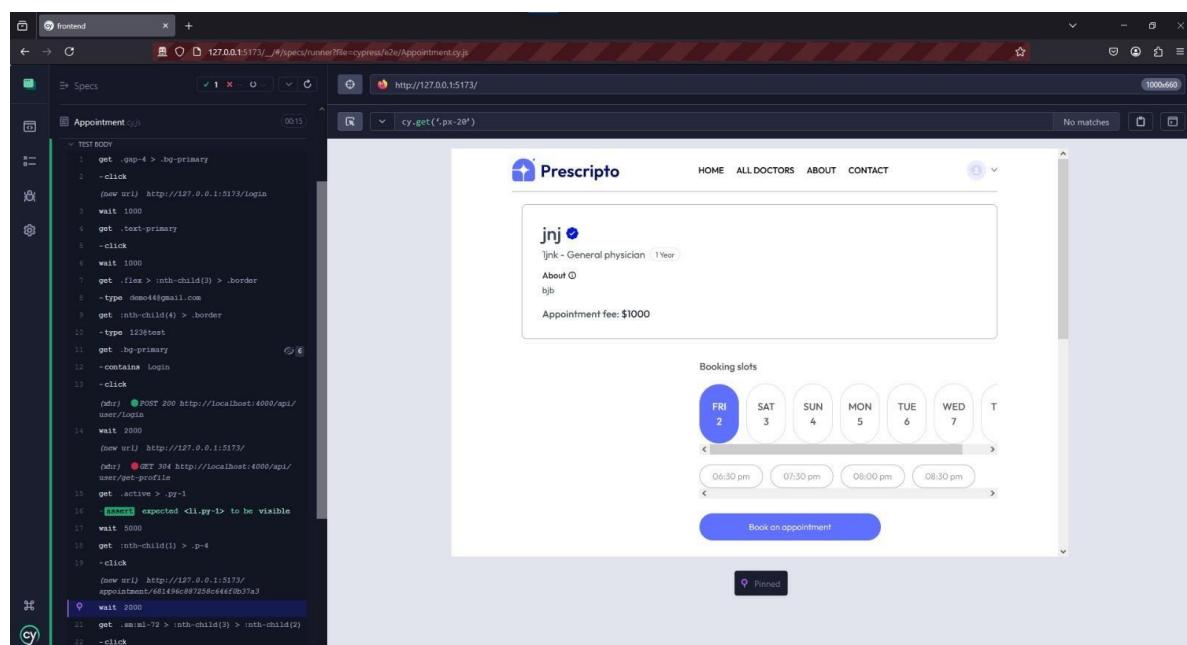
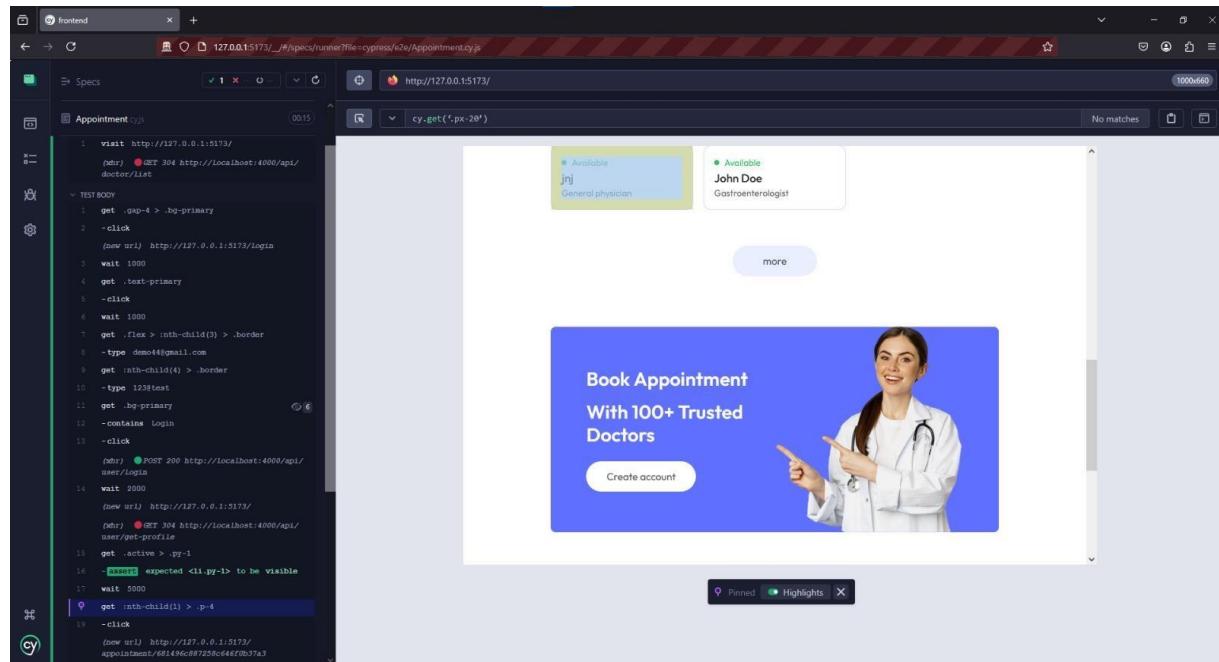
Book Appointment With Trusted Doctors

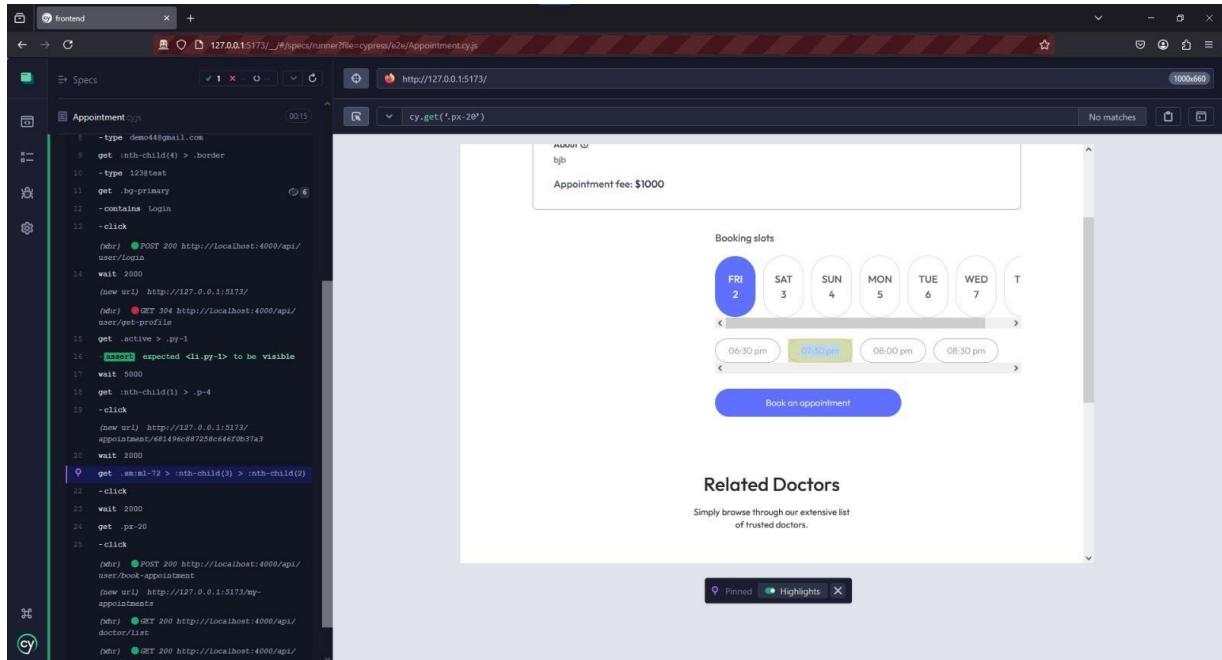
Simply browse through our extensive list of trusted doctors, schedule your appointment hassle-free.

Book appointment →

Find by Speciality

Simply browse through our extensive list of trusted doctors, schedule your





frontend

Specs

Appointment.cy.js

```

1  - type: demo44@gmail.com
2  get :nth-child(4) > .border
3  - type: 123test
4  get :bg-primary
5  - contains: Login
6  - click
7  (whr) POST 200 http://localhost:4000/api/user/login
8  wait 2000
9  (new url) http://127.0.0.1:5173/
10 (whr) GET 304 http://localhost:4000/api/user/get-profile
11 get :active > .py-1
12 - expect: expected <li>.py-1 to be visible
13 wait 5000
14 get :nth-child(1) > .p-4
15 - click
16 (new url) http://127.0.0.1:5173/appointments/681496c887258c646f0b37a3
17 wait 2000
18 get :sm:el-72 > :nth-child(3) > :nth-child(2)
19 - click
20 wait 2000
21 get :px-20
22 - click
23 wait 2000
24 get :px-20
25 - click
26 (whr) POST 200 http://localhost:4000/api/user/book-appointment
27 (new url) http://127.0.0.1:5173/my-appointments
28 (whr) GET 200 http://localhost:4000/api/doctors/list
29 (whr) GET 200 http://localhost:4000/api/doctors

```

Appointment fee: \$1000

Booking slots

FRI 2 SAT 3 SUN 4 MON 5 TUE 6 WED 7 T

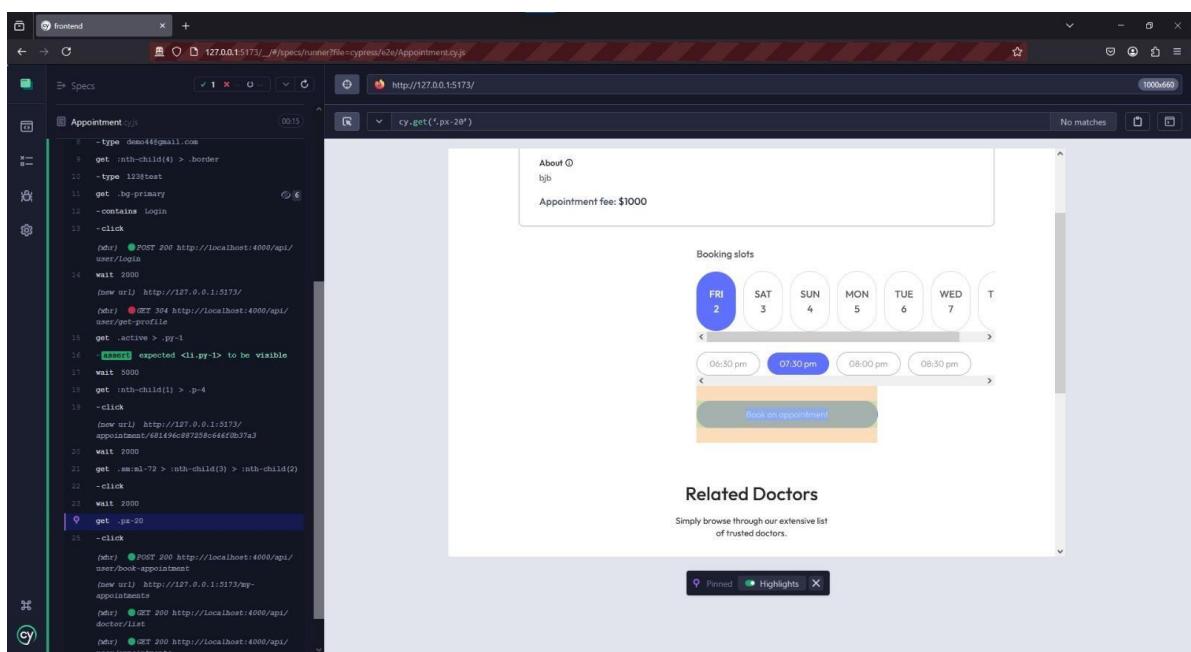
06:30 pm 07:30 pm 08:00 pm 08:30 pm

Book an appointment

Related Doctors

Simply browse through our extensive list of trusted doctors.

Pinned Highlights



frontend

Specs

Appointment.cy.js

```

1  - type: demo44@gmail.com
2  get :nth-child(4) > .border
3  - type: 123test
4  get :bg-primary
5  - contains: Login
6  - click
7  (whr) POST 200 http://localhost:4000/api/user/login
8  wait 2000
9  (new url) http://127.0.0.1:5173/
10 (whr) GET 304 http://localhost:4000/api/user/get-profile
11 get :active > .py-1
12 - expect: expected <li>.py-1 to be visible
13 wait 5000
14 get :nth-child(1) > .p-4
15 - click
16 (new url) http://127.0.0.1:5173/appointments/681496c887258c646f0b37a3
17 wait 2000
18 get :sm:el-72 > :nth-child(3) > :nth-child(2)
19 - click
20 wait 2000
21 get :px-20
22 - click
23 wait 2000
24 get :px-20
25 - click
26 (whr) POST 200 http://localhost:4000/api/user/book-appointment
27 (new url) http://127.0.0.1:5173/my-appointments
28 (whr) GET 200 http://localhost:4000/api/doctors/list
29 (whr) GET 200 http://localhost:4000/api/doctors

```

About bjb

Appointment fee: \$1000

Booking slots

FRI 2 SAT 3 SUN 4 MON 5 TUE 6 WED 7 T

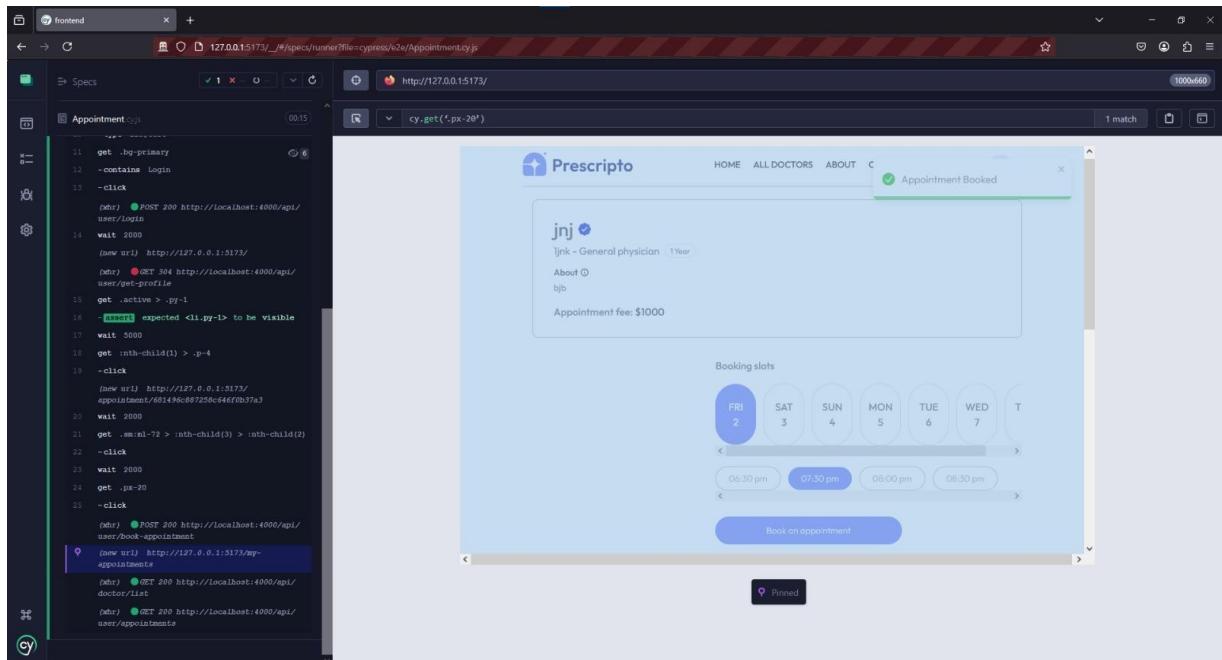
06:30 pm 07:30 pm 08:00 pm 08:30 pm

Book an appointment

Related Doctors

Simply browse through our extensive list of trusted doctors.

Pinned Highlights



### 3. List of identified defects with priority and severity levels.

4. Defect ID	Description	Module	Severity	Priority	Status
DEF-001	Profile picture upload allows unsupported file types (e.g., .exe, .bat)	Edit Profile	Medium	High	Fixed
DEF-002	No error message appears for incorrect	Login	Low	Medium	Fixed

	login credentials				
DEF-003	Admin "Add Doctor" form allows submission without required fields	Admin Dashboard	High	High	Fixed
DEF-004	Phone number field accepts alphabetic characters	Edit Profile	Medium	Medium	Fixed
DEF-005	No success message displayed after successful user registration	Sign-Up	Low	Low	Fixed
DEF-006	Newly added doctors do not show immediately in list without page refresh	Admin Dashboard	Medium	Medium	Fixed
DEF-007	On mobile view, form fields in admin dashboard overlap due to CSS issues	Admin Dashboard	Medium	Low	Pending
DEF-008	User can proceed to book an appointment without selecting a doctor	Appointment	High	High	In Progress
DEF-009	Gender dropdown in profile edit	Edit Profile	Low	Low	Fixed

	not pre-populated with existing data				
--	--------------------------------------	--	--	--	--

## 5. Conclusion

Testing has been a key consideration in deciding the effectiveness of healthcare solutions such as Prescripto. By employing a governed testing approach that entailed manual audit and Cypress-led automation, we ensured the platform meets quality parameters and delivers consistent user experiences.

By emphasizing actual-world interactions and important attributes, our group ensured the functional completeness and reliability of Prescripto. This extensive test atmosphere is an indicator of our commitment to healthcare perfection and enhancement.

In the future, we plan on expanding our coverage of testing across features like doctor reviews, role-based authorization, and appointment scheduling. Prescripto is developed with automation in its core and is best situated to scale up, grow, and be a trusted healthcare ally.