

## 1. E2E test plan

### 1.1 E2E-001 – Register and Super Admin Verify Account Function

Test ID: E2E-001

ชื่อการทดสอบ: การลงทะเบียนผู้ใช้ใหม่ Student/Recruiter และการยืนยันบัญชีโดย Super Admin

เป้าหมาย: เพื่อทดสอบว่าผู้ใช้สามารถลงทะเบียนใหม่ Student/Recruiter ได้สำเร็จ และ Super Admin สามารถตรวจสอบ/อนุมัติ/ปฏิเสธบัญชีได้

#### Preconditions

- ผู้ใช้งานเข้าสู่หน้า Login ของเว็บไซต์ได้
- ผู้ใช้งานเตรียมไฟล์รูปภาพที่ถูกต้อง (.jpg, .png) สำหรับยืนยันตัวตน (บัตรนักศึกษา หรือ บัตรพนักงาน)
- ผู้ใช้งานยังไม่ได้เข้าสู่ระบบ (อยู่ในสถานะ Guest)
- มีบัญชีผู้ใช้ระดับ Super Admin ที่ถูกต้องอยู่ในระบบ
- มีบัญชีผู้ใช้ใหม่ (Student หรือ Recruiter) ที่ลงทะเบียนเข้ามาแล้ว และมีสถานะเป็น "Pending" (รอการอนุมัติ) อายุไม่เกิน 1 ปี

#### Steps

- ผู้ใช้งานเข้าสู่หน้าเว็บไซต์ (หน้า Login) จากนั้นกดที่ลิงก์ "Sign-up here" ด้านล่างปุ่ม Log in
- จากนั้นเลือก role โดยมี 2 role คือ Student และ Recruiter
- เมื่อเลือก role เรียบร้อยแล้วให้กรอกข้อมูลให้ครบถ้วนโดยใช้อีเมล @kmutt.ac.th เท่านั้น
- อัปโหลดรูปภาพ หากเป็น Student ให้อัปโหลดบัตรนักเรียน หากเป็น Recruiter ให้อัปโหลดบัตรพนักงาน
- เมื่อกรอกข้อมูลครบแล้วกด Submit
- เข้าสู่ระบบด้วยบัญชี Super Admin
- กดที่เมนู "Verify Account New User" เพื่อไปยังหน้ารายการผู้รออนุมัติ
- ค้นหาชื่อหรืออีเมลของผู้ใช้งานที่ต้องการปฏิเสธ แล้วกดปุ่ม "View" เพื่อดูรายละเอียด
- หากบัญชีผู้ใช้ใหม่ตรงตามเงื่อนไขที่กำหนด Super Admin กด Approve และหากผู้ใช้ใหม่ผิดกฎหมาย กำหนด Reject

## Expected Result

- ระบบแสดงหน้าต่างแจ้งเตือน (Browser Alert) ข้อความว่า "สมัครสมาชิกสำเร็จ! 🎉 รอการอนุมัติจาก ระบบ"
- เมื่อกดปิดแจ้งเตือน ระบบจะย้ายหน้าผู้ใช้งานไปยังหน้า Pending โดยอัตโนมัติ
- รายชื่อผู้ใช้งานที่ถูกกด Approve ต้องหายไปจากหน้ารายการรออนุมัติ (Pending List) หรือสถานะในระบบเปลี่ยนเป็น Approve
- รายชื่อผู้ใช้งานที่ถูกกด Reject ต้องหายไปจากหน้ารายการรออนุมัติ (Pending List) หรือสถานะเปลี่ยนเป็น Reject ในฐานข้อมูล
- หากผู้ใช้ที่ Super Admin กด Approve แล้ว ผู้ใช้งานคนดังกล่าวจะสามารถ Login เข้าสู่ระบบได้แล้ว

## 1.2 E2E-002 – Login (Email and Password)

Test ID: E2E-002

วัตถุประสงค์: การเข้าสู่ระบบด้วยบัญชีที่ได้รับการอนุมัติแล้ว

เป้าหมาย: เพื่อตรวจสอบว่าผู้ใช้งานที่มีสถานะ "Approve" สามารถเข้าสู่ระบบได้สำเร็จ และระบบทำการเปลี่ยนหน้าไปยังหน้าแรกได้ถูกต้องตามสิทธิ์การใช้งาน (Role) ของตนเอง

### Preconditions

- ผู้ใช้งานเข้าถึงหน้า Login ของเว็บไซต์
- มีบัญชีผู้ใช้ (Student หรือ Recruiter) ที่ผ่านการลงทะเบียนและได้รับการอนุมัติ (Approve) จาก Super Admin เรียบร้อยแล้ว
- ผู้ทดสอบทราบ อีเมล (Email) และ รหัสผ่าน (Password) ที่ถูกต้องของบัญชีดังกล่าว

### Steps

- เปิดหน้าเว็บไซต์
- กรอก Email และ Password ของบัญชีที่ได้รับการอนุมัติแล้ว
- กดปุ่ม "Log in"
- รอระบบประมวลผลสักครู่ จนกว่าจะมีหน้าต่างแจ้งเตือน (Browser Alert) เด้งขึ้นมา
- กดปุ่ม "OK" บนหน้าต่างแจ้งเตือนที่แสดงข้อความ "เข้าสู่ระบบสำเร็จ!"

### Expected Result

- ระบบแสดง Pop-up แจ้งเตือนข้อความว่า "เข้าสู่ระบบสำเร็จ!"
- ระบบพาผู้ใช้งานไปยังหน้าแรกได้ถูกต้องตาม Role ของบัญชีนั้น หากเป็น Student ต้องไปที่หน้า /student/home , หากเป็น Recruiter ต้องไปที่หน้า /recruiter/home

### 1.3 E2E-003 – Forget Password Login

Test ID: E2E-003

ชื่อการทดสอบ: การรีเซ็ตรหัสผ่านใหม่ด้วย OTP

เป้าหมาย: เพื่อตรวจสอบว่าผู้ใช้งานสามารถขอรหัส OTP ทางอีเมล, ยืนยันตัวตนด้วย OTP และทำการตั้งค่ารหัสผ่านใหม่ (Reset Password) ได้สำเร็จ

### Preconditions

- ผู้ใช้งานอยู่ที่หน้าเข้าสู่ระบบ
- มีบัญชีผู้ใช้ (Student หรือ Recruiter) ที่ลงทะเบียนไว้ในระบบแล้วอย่างถูกต้อง
- (กรณีทดสอบจริง) ระบบ Backend สามารถส่งอีเมลออกได้จริง หรือ (กรณี Automation) มีการ Mock API เพื่อตอบกลับค่า OTP ที่ถูกต้อง

### Steps

- คลิกที่ปุ่ม "forgot password" ในหน้า Login
- เมื่อเข้าสู่หน้ากรอกอีเมล ให้ระบุ อีเมลที่ลงทะเบียนไว้ และกดปุ่ม "Send OTP"
- ระบบเปลี่ยนหน้าไปที่หน้ากรอก OTP ให้ทำการกรอก รหัส OTP 6 หลัก ที่ถูกต้อง (หรือที่ Mock ไว้) และกดปุ่ม "Verify OTP"
- เมื่อระบบตรวจสอบ OTP ผ่านและพาไปหน้าตั้งรหัสผ่านใหม่ ให้กรอก รหัสผ่านใหม่ (New Password) และ ยืนยันรหัสผ่านใหม่ (Confirm Password) ให้ตรงกัน
- กดปุ่ม "Submit"

### Expected Result

- หน้าเรียบมีการเปลี่ยนหน้าตามลำดับอย่างถูกต้อง (Input Email -> Input OTP -> Set New Password -> Back to Login)

- ระบบต้องพานักเรียนกลับมาหน้า Login เพื่อให้เข้าสู่ระบบด้วยรหัสผ่านใหม่

#### 1.4 E2E-004 – Upload Portfolio

Test ID: E2E-004

ชื่อการทดสอบ: Student ส่งผลงาน (Portfolio) ครั้งแรกสำเร็จ

เป้าหมาย: ตรวจสอบว่า Student สามารถอัปโหลด Portfolio และบันทึกลง DB ได้

##### Preconditions

- ผู้ใช้ล็อกอินเป็น Student แล้ว
- หน้า home มีปุ่ม Upload Portfolio
- Backend endpoint สำหรับสร้าง Portfolio ทำงาน

##### Steps

- คลิกปุ่ม Upload Portfolio ที่หน้า home
- กรอกข้อมูลทุกช่องที่จำเป็น (ชื่อผลงาน, คำอธิบาย, ปี ฯลฯ)
- เลือกไฟล์ภาพปกผลงาน และไฟล์อื่นๆ ที่ต้องการแนบในผลงาน เช่น .pdf, .jpg, .png
- หากกดปุ่ม Upload ผลงานจะขึ้นสถานะ Pending ใน Profile page หรือ ถ้าเลือกปุ่ม Draft ผลงานจะขึ้นสถานะ Draft ใน Profile page
- รอให้หน้าโหลดเสร็จ

##### Expected Result

- ระบบแสดงข้อความส่งผลงานสำเร็จ
- ในหน้า Profile มีการ์ด หรือรายการผลงานใหม่เพิ่มขึ้น
- สถานะแรกของผลงานเป็น Pending / Draft ตามที่ออกแบบ
- มีข้อมูลผลงานและไฟล์ถูกบันทึกลง DB

#### 1.5 E2E-005 – Edit & Resubmit

Test ID: E2E-005

ชื่อการทดสอบ:แก้ไข Portfolio ที่ถูกติกลับและส่งใหม่

เป้าหมาย: ตรวจสอบว่า Student สามารถแก้ไขผลงานที่ถูก Reject และ Resubmit ได้

### Preconditions

1. มี Portfolio ของ Student อย่างน้อย 1 ผลงาน
2. สถานะผลงานอยู่ใน Reject หรือ Draft
3. Student ล็อกอินอยู่

### Steps

1. ไปที่หน้า Profile
2. เลือกผลงานที่สถานะ Reject หรือ Draft
3. คลิกสัญญาลักษณ์ดินสอเพื่อทำการ Edit หรือ Resubmit
4. แก้ไขข้อมูลบางส่วน (เช่น แก้คำอธิบาย หรืออัปโหลดไฟล์ใหม่)
5. กดปุ่ม Submit หรือ Draft
6. รอให้ระบบประมวลผล

### Expected Result

1. ระบบบันทึกข้อมูลเวอร์ชันใหม่ของ Portfolio
2. สถานะของผลงานเปลี่ยนเป็น Pending หรือ Draft
3. Admin Advisor เห็นผลงานขึ้นในรายการที่ต้องตรวจสอบ (Verify Portfolio)

## 1.6 E2E-006 – Verify Portfolio Workflow (Advisor → Super Admin)

Test ID: E2E-006

ชื่อการทดสอบ: การตรวจสอบผลงานระหว่าง Admin Advisor และ Super Admin

เป้าหมาย: ตรวจสอบว่าทั้ง Admin Advisor และ Super Admin สามารถตรวจสอบและอนุมัติหรือปฏิเสธผลงานได้ครบถ้วน

### Preconditions

1. มี Portfolio ที่สถานะ Pending จาก Student
2. มีบัญชี Admin Advisor และ Super Admin
3. ทั้งสอง role สามารถเข้าสู่ระบบได้

### Steps

1. ล็อกอินเป็น Admin Advisor
2. ไปหน้า Verify Portfolio (Admin Advisor)
3. คลิกปุ่ม review เพื่อเปิดดู Portfolio ที่ Pending
4. หากต้องการ Reject ผลงาน จำเป็นต้องใส่ feedback หรือเหตุผลเพิ่มเติม แต่ถ้าหากต้องการ Approve สามารถส่งต่อให้ Super Admin เพื่อ Verify Portfolio เป็นครั้งสุดท้าย
5. Logout
6. ล็อกอินเป็น Super Admin
7. ไปหน้า Verify Portfolio (Super Admin)
8. คลิกปุ่ม view เพื่อเปิด Portfolio เดียวกัน
9. หากต้องการ Reject ผลงาน จำเป็นต้องใส่ feedback หรือเหตุผลเพิ่มเติม แต่ถ้าหากต้องการ Approve สามารถส่งเพื่อยืนยันสถานะ Approved ในหน้า Profile ของ Student ว่าผลงานนี้ผ่านการ Approve และสามารถแสดงผลงานได้

### Expected Result

1. Admin Advisor สามารถให้คำแนะนำและส่งต่อผลงานได้
2. Super Admin เห็นข้อมูลที่ Advisor ส่งมา และสามารถ Approve หรือ Reject ได้
3. สถานะผลงานเปลี่ยนไปตาม action สุดท้าย (เช่น Approved หรือ Reject)
4. Student เห็นสถานะและ feedback หากถูก Reject ผลงาน

## 1.7 E2E-007 – Status Tracking

Test ID: E2E-007

ชื่อการทดสอบ: Student ติดตามสถานะผลงานได้

เป้าหมาย: ตรวจสอบว่าระบบแสดงสถานะผลงานถูกต้องตาม workflow

### Preconditions

1. Student มี Portfolio อย่างน้อย 1 ชิ้น
2. ผลงานเคยถูกเปลี่ยนสถานะ เช่น Draft → Pending → In Process → Approved / Reject

### Steps

1. ล็อกอินเป็น Student
2. ไปที่หน้า home → Profile
3. ดูรายการ Portfolio ทั้งหมด
4. ตรวจสอบแท็กสถานะของแต่ละผลงาน

#### Expected Result

1. ระบบแสดงสถานะถูกต้อง เช่น Draft, Pending, In Process, Approved, Reject
2. สถานะสอดคล้องกับ action ล่าสุดของ Advisor หรือ Super Admin

### 1.8 E2E-008 – Search & Public Feed

Test ID: E2E-008

ชื่อการทดสอบ: ค้นหาและแสดงผลใน Public Feed หรือ home

เป้าหมาย: ตรวจสอบว่า Student หรือ Recruiter สามารถค้นหาผลงานที่เผยแพร่แล้วได้

#### Preconditions

1. มี Portfolio อย่างน้อย 1 ผลงานที่สถานะ Approved และตั้งค่า Visibility เป็น Public
2. มีหน้า Public Feed หรือ home

#### Test Data

1. ล็อกอินเป็น Recruiter หรือ Student
2. อยู่หน้า home
3. พิมพ์คำค้นในช่อง Search = DB (ชื่อผลงาน)
4. สังเกตผลลัพธ์ที่แสดง

#### Steps

1. ล็อกอินเป็น Recruiter หรือ Student
2. อยู่หน้า home
3. พิมพ์คำค้นในช่อง Search (เช่น ชื่อผลงาน หรือ username)
4. สังเกตผลลัพธ์ที่แสดง

#### Expected Result

- ระบบแสดงรายการผลงานที่ตรงกับคำค้นหา
- ไม่แสดงผลงานที่เป็น Private หรือยังไม่ Approved
- สามารถคลิกเข้าไปดูรายละเอียด Portfolio ได้

### 1.9 E2E-009 – Filter (Year / Category)

Test ID: E2E-009

ชื่อการทดสอบ: การกรองผลงานใน home ด้วย Filter

เป้าหมาย: ตรวจสอบว่าผู้ใช้สามารถกรองผลงานตาม Year หรือ Category ได้

#### Preconditions

- มีผลงานหลายชิ้น ที่มีค่า Year / Category ต่างกัน
- หน้า home มีส่วน Filter ให้เลือก

#### Test Data

- ล็อกอินเป็น Recruiter หรือ Student
- ไปหน้า home
- เลือกค่าจาก Filter ที่กำหนดให้ Year = 2025 , Category = Database
- สังเกตผลลัพธ์

#### Steps

- ล็อกอินเป็น Recruiter หรือ Student
- ไปหน้า home
- เลือกค่าจาก Filter
- สังเกตผลลัพธ์

#### Expected Result

- ระบบแสดงเฉพาะผลงานที่ตรงกับเงื่อนไขที่เลือก
- เมื่อเคลียร์ Filter → รายการกลับมาเป็นทั้งหมด

## 1.10 E2E-010 – Reviewer Comment

Test ID: E2E-010

ชื่อการทดสอบ: Recruiter หรือ Student ใส่ความคิดเห็นในผลงานที่ซึ่งขอบได้

เป้าหมาย: ตรวจสอบว่า Recruiter หรือ Student สามารถเพิ่มความคิดเห็นและ Recruiter หรือ Student เท่านั้นได้

### Preconditions

1. มี Portfolio อย่างน้อย 1 ผลงานที่สถานะ Approved และตั้งค่า Visibility เป็น Public
2. Recruiter หรือ Student สามารถเข้าหน้า comment ได้
3. Recruiter หรือ Student สามารถดูหรือพิมพ์ความคิดเห็นลงในผลงานได้

### Steps

1. ล็อกอินเป็น Recruiter หรือ Student
2. ไปหน้า home
3. คลิกเปิด Portfolio ที่ต้องการ
4. พิมพ์ความคิดเห็นในช่อง Comment
5. กด post comment

### Expected Result

1. Comment ของ Recruiter หรือ Student ถูกบันทึกในระบบ
2. Recruiter หรือ Student เท่านั้นข้อความความคิดเห็นของ Recruiter หรือ Student ที่เขียนไว้

## 1.11 E2E-011 – Visibility Setting (Public / Private)

Test ID: E2E-011

ชื่อการทดสอบ: ตั้งค่าการเผยแพร่ผลงาน (Public/Private)

เป้าหมาย: ตรวจสอบว่า Student กำหนดความเป็นส่วนตัวของผลงานได้

### Preconditions

1. Student ล็อกอินอยู่ในระบบ

- มี Portfolio อย่างน้อย 1 ผลงานที่สถานะ Approved และตั้งค่า Visibility เป็น Public หรือ Private ได้

#### Test Data

- ไปที่หน้า Profile
- เลือกผลงานที่สถานะ Approved และมีชื่อผลงานว่า test
- เปลี่ยนค่า Visibility จาก Private → Public
- ทดสอบด้วยการเข้า Public Feed หรือ home

#### Steps

- ไปที่หน้า Profile
- เลือกผลงานที่ผ่านการ Approve ที่ต้องการตั้งค่า
- เปลี่ยนค่า Visibility จาก Private → Public (หรือกลับกัน)
- ทดสอบด้วยการเข้า Public Feed หรือ home

#### Expected Result

- ค่า Visibility ของผลงานถูกอัปเดต
- ถ้าเป็น Public → Recruiter หรือ Student เห็นใน Feed หรือ home
- ถ้าเป็น Private → เห็นเฉพาะเจ้าของ (Student)

## E2E-012 – Logout

Test ID: E2E-011

ชื่อการทดสอบ: Logout ออกจากระบบ

เป้าหมาย: เพื่อตรวจสอบว่าผู้ใช้งานที่เข้าสู่ระบบอยู่ สามารถกดปุ่ม Logout เพื่ออกจากระบบและกลับไปยังหน้า Login ได้ถูกต้อง

#### Preconditions

- ผู้ใช้งานได้ทำการเข้าสู่ระบบ (Login) เรียบร้อยแล้ว
- ผู้ใช้งานกำลังอยู่ที่หน้าหลัก (เช่น /student/home หรือ /recruiter/home) หรือหน้าที่มีเมนูแสดงปุ่ม Logout

## Steps

1. ค้นหาปุ่มหรือลิงก์ที่มีคำว่า "Log out" (หรือ "ออกจากระบบ") บนหน้าจอ
2. คลิกที่ปุ่ม "Log out"
3. รอให้ระบบทำการประมวลผลและเปลี่ยนหน้า

## Expected Result

1. ระบบพาผู้ใช้งานกลับไปยังหน้า เข้าสู่ระบบ (Login Page) หน้าแรกของเว็บไซต์ได้ถูกต้อง

## 2. Evidence of implemented tests

- E2E-001 – Register and Super Admin Verify Account Function ทำ E2E (Playwright) Test โดยใช้โค้ดดังนี้

- New Account : Student Register

```
// tests/e2e-register.spec.js
const { test, expect } = require('@playwright/test');

// ● 1. ใช้ test.describe.serial เพื่อ串ต่อ TC1 => TC2 => TC3 ตามลำดับ
test.describe.serial('E2E Registration and Approval Flow', () => {

    let uniqueEmail; // ตัวแปรglobal

    // =====
    // TC1: Student Register
    // =====
    test('TC1: Navigate to Student Register', async ({ page }) => {
        await page.goto('/studentport/login');
        await page.click('text=Sign-up here');
        await expect(page).toHaveURL('/select-role');
        await page.click('text=Student');

        const randomId = Date.now();
        // ▲ ข้อสังเกต: คำนี้จะถูก TC2 เผยแพร่ในภายหลัง
        uniqueEmail = `Somyhing${randomId}@kmutt.ac.th`;

        await expect(page).toHaveURL('/register/student/');
        await expect(page.locator('h2')).toContainText('Student Account Registration');
        await page.fill('input[name="firstName"]', 'Somyhing');
        await page.fill('input[name="surname"]', 'Jaidum');
        await page.fill('input[name="password"]', 'pass123');
        await page.fill('input[name="email"]', uniqueEmail);
    });
});
```

```

const fakeImageBuffer = Buffer.from('fake-image-content');

await page.setInputFiles('input[type="file"]', {
  name: 'student_card.png',
  mimeType: 'image/png',
  buffer: fakeImageBuffer,
});

page.once('dialog', async dialog => {
  await dialog.accept();
});

await page.click('.submit-btn');
await expect(page).toHaveURL(/.*pending/);
});

```

The screenshot shows the Playwright interface during a test run. The top part displays a timeline of the test execution, with various actions like 'Passed', 'Fill "surname"', 'Click "submit-btn"', and 'Expect "toHaveURL"' marked with their respective execution times. Below the timeline is a list of test cases, all of which have passed. To the right of the timeline is a screenshot of a web browser window titled 'Student Account Registration'. The form contains fields for 'First name' (Somying), 'Last name' (Adee), 'Password' (password123), 'Email Address (Organization)' (somying176560601285@chromium), and a file input field. A preview button is visible. At the bottom of the interface is a detailed 'Network' tab showing a list of network requests and responses, including the main page load and various CSS, JS, and image files.

(E2E-001 Student Register Test Run Screenshot/Output image)

## O New Account : Recruiter Register

```
// TC3: Recruiter Register
// =====
test('TC2: Navigate to Recruiter Register and Submit', async ({ page }) => {
  await page.goto('/studentport/login');
  await page.click('text=Sign-up here');
  await expect(page).toHaveURL(/.*select-role/);
  await page.click('text=Recruiter');

  await expect(page).toHaveURL(/.*register/recruiter/i);
  await expect(page.locator('h2')).toContainText('Recruiter Account Registration');

  const randomId = Date.now();
  // ▲ គាន់ថា id នឹងត្រួវតិចជាកំណែង Recruiter នៃការបង្កើតគម្រោង
  uniqueEmail = `Jack${randomId}@kmutt.ac.th`;

  await page.fill('input[name="firstName"]', 'Jack');
  await page.fill('input[name="surname"]', 'Jaidum');
  await page.fill('input[name="password"]', 'pass123');
  await page.fill('input[name="email"]', uniqueEmail);
  await page.fill('input[name="organization"]', "KMUUT");

  const fakeImageBuffer = Buffer.from('fake-image-content');
  await page.setInputFiles('input[type="file"]', {
    name: 'employee_card.png',
    mimeType: 'image/png',
    buffer: fakeImageBuffer,
  });
});

page.once('dialog', async dialog => {
  await dialog.accept();
});

await page.click('.submit-btn');
await expect(page).toHaveURL(/.*pending/);

console.log(`✓ Recruiter registered with email: ${uniqueEmail}`);
});
```

PLAYWRIGHT

Filter (e.g. text, @tag)

Status: all Projects: chromium

4/4 passed (100%)

	Action	Metadata	6 hidden	Before	After
✓ Passed	2.6s				
Fill "Jack1765606018538@k...	411ms	Fill(locator("input[name="password"]))			
Fill "KMUUT"	4ms	Fill(locator("input[name="organization"]))			
Set input files	5ms	Set(locator("input[type="file"]))			
Click locator("submit-btn")	40ms	Click(locator("submit-btn"))			
Expect "toHaveURL"	857ms	Expect(toHaveURL)			
Accept dialog	24ms	Accept(dialog)			

Actions

Metadata

6 hidden

Action

Before

After

Locator

Source

Call

Log

Errors

Console

Network

Attachments

Annotations

Filter network

All

Fetch

HTML

JS

CSS

Font

Image

Name	Method	Status	Content Type	Duration	Size	Start	Route
login	GET	404	text/html	285ms	721	269ms	
main.a283fba5.js	GET	200	application/javascript	149ms	105.0K	510ms	/
main.1a722b82.css	GET	200	text/css	155ms	6.1K	511ms	/
css2?family=Poppins:wght@...	GET	200	text/css	413ms	1.0K	664ms	/
studentport_logo.431b9067c...	GET	200	image/png	77ms	51.4K	1.1s	/
pxiByp8kv8JHgFVrLj6Z1xF...	GET	200	font/woff2	111ms	7.8K	1.1s	/
pxiEyp8kv8JHgFVrJjfecg.wo...	GET	200	font/woff2	510ms	7.7K	1.1s	/
pxiByp8kv8JHgFVrC27Z1xF...	GET	200	font/woff2	104ms	7.7K	1.2s	/
e827bb74-18cd-4777-b645...	GET	200	image/png	2ms	18	1.6s	/
register	POST	201	application/json	683ms	410	1.6s	/
e1d49401-0a06-47d8-9b13...	GET	200	image/png	13ms	18	1.6s	/
1e1c2f50-5290-4779-9f54...	GET	200	image/png	3ms	18	2.2s	/
pxiByp8kv8JHgFVrLGT9Z1xL...	GET	200	font/woff2	87ms	7.6K	2.2s	/

TESTING OPTIONS

SETTINGS

(E2E-001 Recruiter Register Test Run Screenshot/Output image)

## O Super Admin Approve New Account

```
// TC2: Admin Verify-Approve(User ผู้ดูแล = Student จาก TC1)
// =====
test('TC3: Verify account by Super admin', async ({ page }) => {
  // เช็คกันหน่อยว่ามี email ฟรีมาให้
  if (!uniqueEmail) throw new Error("Unique Email is undefined!");

  await page.goto('/studentport/login');

  const ADMIN_EMAIL = "super_admin@kmutt.ac.th";
  const ADMIN_PASS = "123";

  await page.fill('input[type="email"]', ADMIN_EMAIL);
  await page.fill('input[type="password"]', ADMIN_PASS);
  await page.click('.btn');

  await expect(page).toHaveURL(/.*super\/verify/);

  await page.click('text=Verify Account New User');
  await expect(page).toHaveURL(/.*super\/verify-acc/);

  // 🔍 รอให้มีผลตารางใหม่ต่อท้าย (สำคัญมาก กับ Error หากไม่เจء)
  const emailLocator = page.locator('tr').filter({ hasText: uniqueEmail }).first();
  await expect(emailLocator).toBeVisible({ timeout: 10000 });
});

// คลิก View
await emailLocatorlocator('text=View').click();

const approveBtn = page.locator('button', { hasText: 'Approve' });

// ✅ ของใหม่: รอให้มีขึ้นจริง
await expect(approveBtn).toBeVisible();

await approveBtn.click();
page.once('dialog', async dialog => await dialog.accept());
await page.waitForURL(/.*super\/verify-acc/)

});
```

The screenshot shows the Playwright Test Run interface with the following details:

- Timeline:** A horizontal timeline at the top showing execution times for each step.
- Test Summary:** Shows 4/4 passed (100%) with a green status bar.
- Test Tree:** A tree view of the test cases:
  - register-verify.spec.js
  - E2E Registration and Approval
  - TC1: Navigate to Student Port
  - TC3: Verify account (highlighted in blue)
  - TC2: Navigate to Registration
  - TC5: Reject account...
- Step Details:** A detailed view of the current step (TC3):
  - Action: Click on 'Approve' button
  - Duration: 23ms
  - Locator: locator('button').filter({ hasText: 'Approve' })
- Network Tab:** Shows a table of network requests with columns: Name, Method, Status, Content Type, Duration, Size, Start, and Route.
- Screenshot:** A screenshot of the 'Student Account Verification' dialog box, showing fields for First name, Surname, Email Address (registered), Password, and Attach your student ID card.

(E2E-001 Super Admin Approve New Account Test Run Screenshot/Output image)

## O Super Admin Reject New Account

```
// TC4: Admin Verify-Reject(User ผู้ดูแล = Student จาก TC3)
// =====
test('TC5: Reject account by Super admin', async ({ page }) => {
  // ให้ uniqueEmail ตัวใหม่จาก TC4
  if (!uniqueEmail) throw new Error("Email undefined!");

  // 1. Login Admin
  await page.goto('/studentport/login');
  const ADMIN_EMAIL = "super_admin@kmutt.ac.th";
  const ADMIN_PASS = "123";
  await page.fill('input[type="email"]', ADMIN_EMAIL);
  await page.fill('input[type="password"]', ADMIN_PASS);
  await page.click('.btn');

  // 2. ไปหน้า List
  await expect(page).toHaveURL(/.*super\/verify/);
  await page.click('text=Verify Account New User');

  // 3. ตรวจสอบ User (BadUser)
  const emailLocator = page.locator('tr').filter({ hasText: uniqueEmail }).first();
  await expect(emailLocator).toBeVisible({ timeout: 1000 });

  // 4. นำ View
  await emailLocatorlocator('text=View').click();
  await page.waitForURL(/.*super\/user-approval/, { timeout: 10000 });

  // 6. ทำปุ่ม Reject
  const rejectBtn = page.locator('button', { hasText: 'Reject' });
  await expect(rejectBtn).toBeVisible();

  // 7. กด Reject (ตัว Dialog ต้องมี แน่นใน code React อาจจะไม่มี confirm)
  // ไม่รู้กันเห็นยา
  page.once('dialog', async dialog => await dialog.accept());
  // กดปุ่ม Reject ไปเลย
  await rejectBtn.click();

  // บรรทัดนี้ Playwright จะ "รอ" ให้ URL เปลี่ยนเป็น verify-acc ให้เอง
  // ถ้าอังกฤษเด็กซ่าๆ ทุกเสี้ยววินาที จนกว่าจะเรื่อย (ภายใน 30 วิ)
  await expect(page).toHaveURL(/.*super\/verify-acc/, { timeout: 30000 });
});

});
```

The screenshot shows the Playwright test run interface. At the top, there's a timeline bar with time markers from 0.0s to 3.5s. Below it, the test results show 4/4 passed (100%). The test file is register-verify.spec.js, and the test case is TC5: Reject account by Super admin. The test steps include navigating to the login page, logging in as super\_admin, going to the verify page, finding the user row, clicking 'View', and then clicking the 'Reject' button. A modal dialog for 'Accept dialog' is shown, and the final step is waiting for the URL to change to verify-acc. The bottom part of the interface shows the Network tab with a table of requests, including a large 404 error for 'login' with a duration of 328ms.

Name	Method	Status	Content Type	Duration	Size	Start	Route
login	GET	404	text/html	328ms	722	263ms	
main.a283fba5.js	GET	200	application/javascript	186ms	105.0K	534ms	
main.1a72b2b82.css	GET	200	text/css	99ms	6.1K	535ms	
css?family=Poppins:wght@...;	GET	200	text/css	368ms	1.0K	629ms	
studentport_logo.431b98067...;	GET	200	image/png	77ms	51.3K	1.0s	
pxByp8v8JHgFvLxEjZ1xlf...;	GET	200	font/woff2	76ms	7.8K	1.0s	
pxEyP8v8JHgFvLJfegc.w...;	GET	200	font/woff2	318ms	7.7K	1.0s	
login	POST	200	application/json	560ms	465	1.1s	
inProcess	GET	200	application/json	255ms	753	1.7s	
pxByp8v8JHgFvLCz7Z1xF...;	GET	200	font/woff2	81ms	7.7K	2.0s	
pending	GET	200	application/json	293ms	985	2.0s	
693d02856a72f9d42287941b	GET	200	application/json	260ms	387	2.5s	
693d02856a72f9d42287941b	GET	200	application/json	149ms	37	2.8s	
1765606021331-employeeCa...	GET	200	image/png	479ms	18	2.9s	

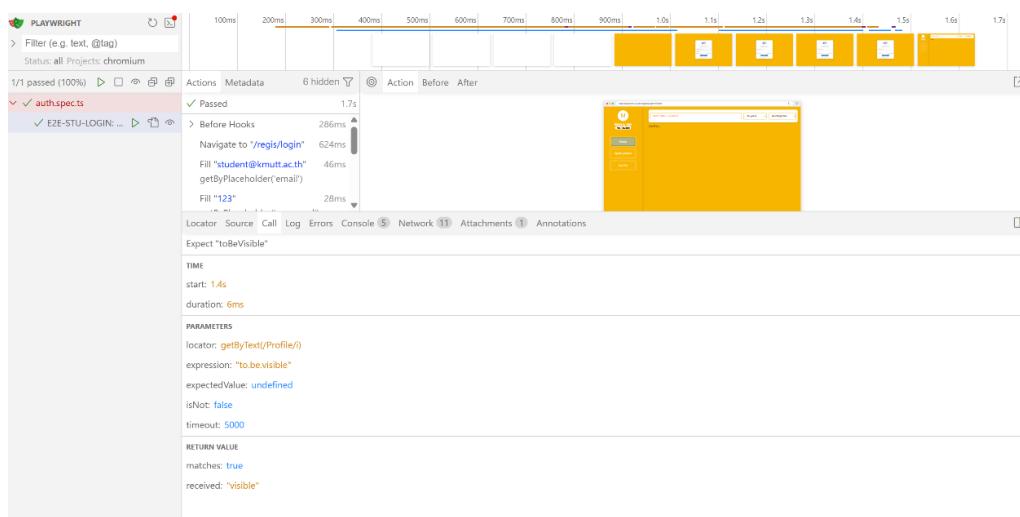
(E2E-001 Super Admin Reject New Account Test Run Screenshot/Output image)

- E2E-002-Login (Email, Password) ทำ E2E (Playwright) Test โดยใช้โค้ดดังนี้

- Login Email & Password Student

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

test('E2E-STU-LOGIN: Student login success → /regis/student/home', async ({ page }) => {
  await page.goto('http://localhost:5000/regis/login');
  await page.getByPlaceholder('email').fill('student@kmutt.ac.th');
  await page.getByPlaceholder('password').fill('123');
  await page.getByRole('button', { name: 'Log in' }).click();
  await page.waitForURL('http://localhost:5000/regis/student/home');
});
```



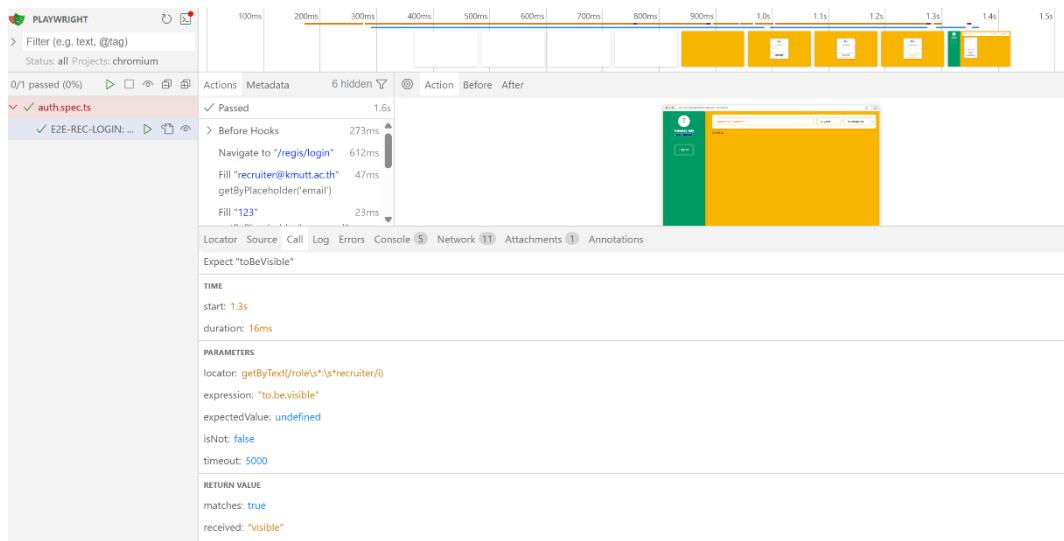
(E2E-002 Test Run Screenshot/Output image)

- Login Email & Password Recruiter

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

test('E2E-REC-LOGIN: Recruiter login success → /regis/recruiter/home', async ({ page }) => {
  await page.goto('http://localhost:5000/regis/login');

  await page.getByPlaceholder('email').fill('recruiter@kmutt.ac.th');
  await page.getByPlaceholder('password').fill('123');
  await page.getByRole('button', { name: 'Log in' }).click();
  await page.waitForURL('http://localhost:5000/regis/recruiter/home');
  await expect(page.getText(/role\s*: \s*recruiter/i)).toBeVisible();
});
```

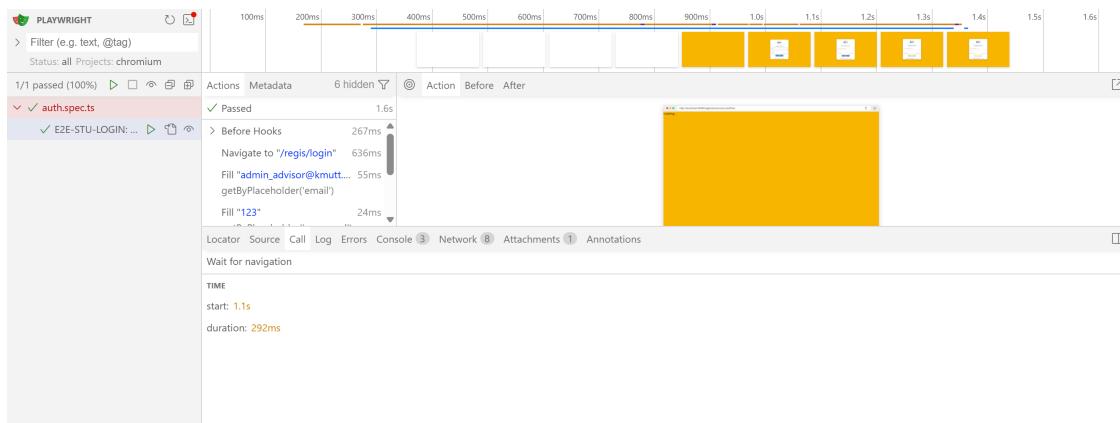


(E2E-002 Test Run Screenshot/Output image)

- Login Email & Password Admin Advisor

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

test('E2E-STU-LOGIN: Admin Advisor login success → /regis/advisor/home', async ({ page }) => {
  await page.goto('http://localhost:5000/regis/login');
  await page.getByPlaceholder('email').fill('admin_advisor@kmutt.ac.th');
  await page.getByPlaceholder('password').fill('123');
  await page.getByRole('button', { name: 'Log in' }).click();
  await page.waitForURL('http://localhost:5000/regis/advisor/veri-portfolio');
});
```



(E2E-002 Test Run Screenshot/Output image)

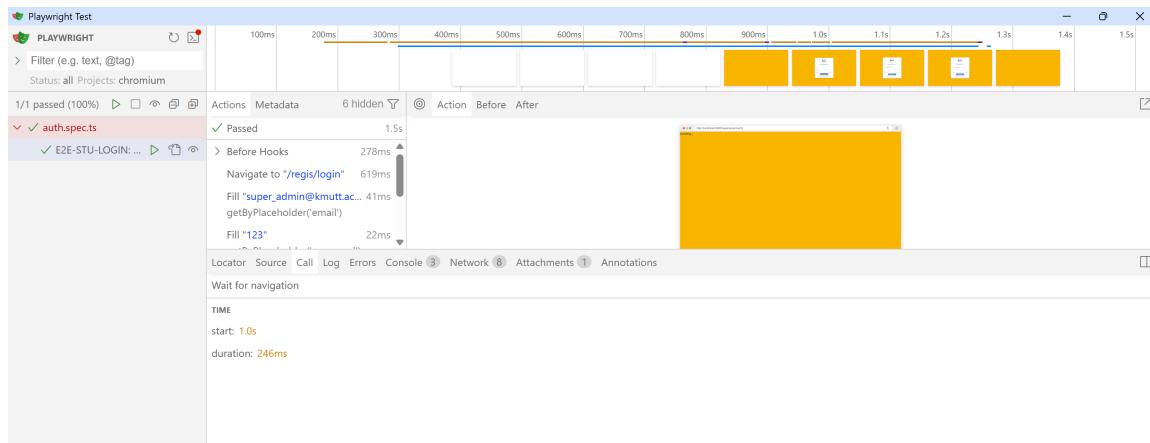
- Login Email & Password Super Admin

```

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

test('E2E-STU-LOGIN: Super Admin login success → /regis/super/home', async ({ page }) => {
  await page.goto('http://localhost:5000/regis/login');
  await page.getByPlaceholder('email').fill('super_admin@kmutt.ac.th');
  await page.getByPlaceholder('password').fill('123');
  await page.getByRole('button', { name: 'Log in' }).click();
  await page.waitForURL('http://localhost:5000/regis/super/verify');
});

```



(E2E-002 Test Run Screenshot/Output image)

- E2E-003 – Forget Password Login ทำ E2E (Playwright) Test โดยใช้โค้ดดังนี้

```

const { test, expect } = require('@playwright/test');

test.describe('E2E-003 ┌ Forget Password Functionality', () => {
  test('TC1: User requests password reset flow (UI Test with Mock API)', async ({ page }) => {
    // =====
    // ● MOCK API: เตรียมการทดสอบเว็บ (ไม่ต้องมี Server จริง)
    // =====

    // 1. ตั้งการขอ OTP (POST /auth/forgot-password)
    await page.route('**/auth/forgot-password', async route => {
      await route.fulfill({
        status: 200,
        contentType: 'application/json',
        body: JSON.stringify({ message: "OTP sent to your email" })
      });
    });
  });
}

```

```

// 2. ตั้งการตรวจสอบ OTP (POST /auth/verify-otp)
await page.route('**/auth/verify-otp', async route => {
  // Backend คุณตั้ง resetToken กับมาเมื่อ OTP ถูก
  await route.fulfill({
    status: 200,
    contentType: 'application/json',
    body: JSON.stringify({
      message: "OTP verified",
      resetToken: "fake-reset-token-12345"
    })
  });
});

// 3. ตั้งการตั้งรหัสใหม่ (POST /auth/reset-password)
await page.route('**/auth/reset-password', async route => {
  await route.fulfill({
    status: 200,
    contentType: 'application/json',
    body: JSON.stringify({ message: "Password reset successful" })
  });
});

```

```

// =====
// ● START TEST: เริ่มทดสอบ UI จริง
// =====

// 1. ไปหน้า Login
await page.goto('/studentport/login');

// 2. กดลิ้งค์ Forgot Password
await page.click('text=forgot password'); // หรือ selector ที่ตรงกับหน้าเว็บคุณ
// ตรวจสอบว่ามาหน้ากรอกอีเมล
// (สมมติ URL คือ /forgot-password หรือ /request-otp)
await expect(page).toHaveURL(/.*forgot-password|.*request-otp/);

// 3. กรอกอีเมล
const emailInput = page.locator('input[type="email"]').or(page.getByPlaceholder('email'));
await emailInput.fill('kitty@kmutt.ac.th');

// 4. กดปุ่มส่ง (Send OTP)
// หากปุ่มมีคำว่า Send หรือ Submit
const sendBtn = page.locator('button', { hasText: /Send|Submit|OTP/i });
await sendBtn.click();

// 5. กรอก OTP (เราจะกรอกเลขมา 4 ไปเลย เพราะเรา Mock API ไว้แล้วว่าให้ผ่านเสมอ)
// เช็คความถูกต้องของ OTP ใหม่ (เน้น input name="otp" หรือ placeholder="OTP")
const otpInput = page.locator('input[name="otp"]').or(page.getByPlaceholder('OTP'));
await expect(otpInput).toBeVisible({ timeout: 10000 });
await otpInput.fill('123456'); // เลขอะไรมีได้

// 6. กด Verify OTP
const verifyBtn = page.locator('button', { hasText: /Verify|Check/i });
await verifyBtn.click();

// --- ตอนนี้หน้าเว็บควรจะเปลี่ยนไปหน้าตั้งรหัสผ่านใหม่ ---

// 7. กรอกรหัสผ่านใหม่
// ปกติจะมี 2 ช่อง: New Password และ Confirm Password
const newPassInput = page.locator('input[name="newPassword"]').or(page.getByPlaceholder('New Password'));
const confirmPassInput = page.locator('input[name="confirmPassword"]').or(page.getByPlaceholder('Confirm Password'));

await expect(newPassInput).toBeVisible({ timeout: 10000 });

await newPassInput.fill('newPass1234');
await confirmPassInput.fill('newPass1234');

```

```
// 8. ຖະໜົນລັບຜູ້ໃຫຍ່
const resetBtn = page.locator('button', { hasText: /Submit/i });

// ດັກ Dialog Success (ຄ້າສິນ)
page.once('dialog', async dialog => {
    console.log(`Success Dialog: ${dialog.message()}`);
    await dialog.accept();
});

await resetBtn.click();

// 9. ຕ່າງອີເມວວ່າກຳລັນມາແນ້າ Login ທີ່ຈະໄດ້
await expect(page).toHaveURL(/.*login/);

console.log("✅ Forgot Password Flow (UI Mock) Completed!");
});
```

The screenshot shows the Playwright test runner interface with the following details:

- Timeline:** A horizontal timeline at the top marks time from 0.0s to 5.0s. The test duration is approximately 4.8s.
- Test Status:** 1/1 passed (100%).
- Test Case:** E2E-003 – Forget Password ... (Passed).
- Actions:** A list of actions taken during the test:
  - Fill "123456"
  - Click
  - Expect "toBeVisible"
  - Fill "newPass1234"
  - Fill "newPass1234"
  - Click
  - Expect "toHaveURL"
- Metadata:** Includes the URL <https://michaelechromium.github.io/studentport/forgot-password> and a screenshot of the browser window showing the StudentPort.com login page with OTP verification.
- Network Tab:** Shows network requests for various assets like CSS, JS, and images.
- Annotations:** Annotations are present on the screenshot, notably around the "Set New Password" button.
- Call Log:** Shows the sequence of API calls made during the test.
- Console:** Shows the command "npm run test" being run.
- Attachments:** Shows a screenshot of the browser window.
- Annotations:** Annotations are present on the screenshot, notably around the "Set New Password" button.

(E2E-003 Test Run Screenshot/Output image)

- E2E-004-Upload Portfolio ทำ E2E (Playwright) Test โดยใช้โค้ดดังนี้

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

test('E2E-STU-UPLOAD: Student uploads portfolio successfully', async ({ page }) => {

  await page.goto('http://localhost:5000/regis/login');
  await page.getByPlaceholder('email').fill('student@kmutt.ac.th');
  await page.getByPlaceholder('password').fill('123');
  await page.getByRole('button', { name: 'Log in' }).click();
  await page.waitForURL('http://localhost:5000/regis/student/home');

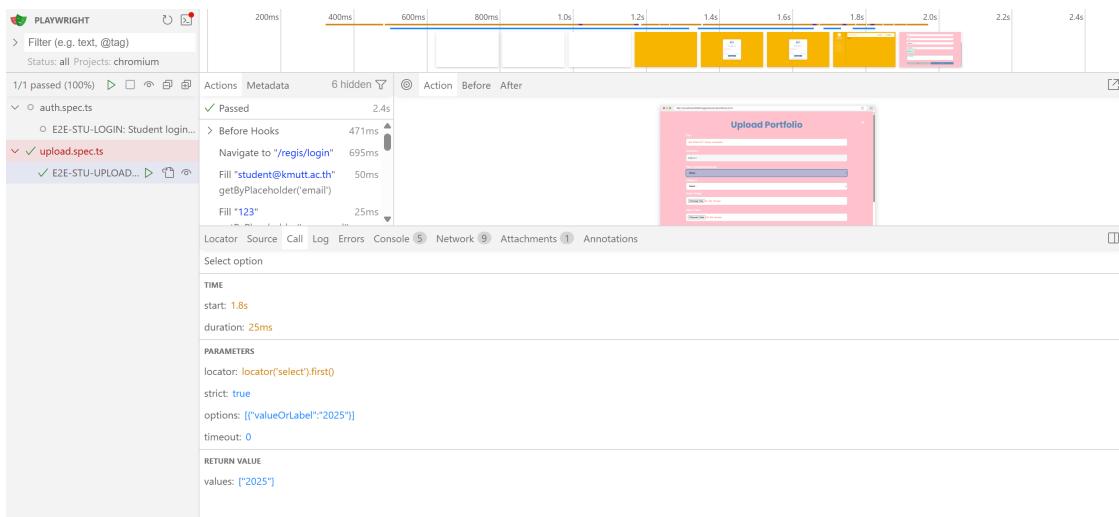
  await page.getByText(/upload portfolio/i).click();

  await page.waitForURL('**regis/student/portfolio-form');

  await page.locator('select').nth(0).selectOption('2025');
  await page.locator('select').nth(1).selectOption('Database');

  const fileInputs = page.locator('input[type="file"]');
  await fileInputs.nth(0).setInputFiles('tests/files/cover.png');
  await fileInputs.nth(1).setInputFiles('tests/files/portfolio.pdf');

  await page.locator('textarea').fill('E2E test upload');
  await page.getByRole('button', { name: '/Upload/i }).click();
});
```



(E2E-004 Test Run Screenshot/Output image)

- E2E-005-Edit and Resubmit ทำ E2E (Playwright) Test โดยใช้โค้ดดังนี้

```

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

async function loginAsStudent(page: Page) {
  await page.goto('http://localhost:5000/regis/login');

  await page.getByPlaceholder('/email/i').fill('student@kmutt.ac.th');
  await page.getByPlaceholder('/password/i').fill('123');
  await page.getByRole('button', { name: /log in/i }).click();

  await page.waitForURL('**/regis/student/home**', { timeout: 10000 });
}

test('E2E-STU-RESUBMIT: Student edits rejected portfolio and resubmits', async ({ page }) => {
  const TARGET_TITLE = 'test'; // ชื่อการ์ดที่อยู่บนหน้าจอ

  await loginAsStudent(page);

  await page.getByText('/profile/i').click();
  await page.waitForURL('**/regis/student/status**', { timeout: 10000 });

  await page.getByRole('button', { name: /^edit$/i }).click();

  const rejectedCardLink = page
    .getByRole('link', { name: new RegExp(`${TARGET_TITLE}.*Rejected`, 'i') })
    .first();
  await expect(rejectedCardLink).toBeVisible();

  const pencil = rejectedCardLink.getByLabel('Edit');
  await expect(pencil).toBeVisible();
  await pencil.click();
  await page.waitForURL('**/regis/student/resubmit**', { timeout: 10000 });

  const NEW_DESC = 'Resubmitted after revision 1 E2E Test';
  await page.locator('textare').first().fill(NEW_DESC);

  await page.getByRole('button', { name: /resubmit|submit/i }).click();

  await page.waitForURL('**/regis/student/status**', { timeout: 10000 });
});

```

The screenshot shows the Playwright UI test runner interface. At the top, there's a timeline bar with a red highlight indicating the duration of the test execution. Below the timeline, the test results are listed under 'Actions' and 'Metadata'. The 'Actions' section details the steps taken, including navigating to the login page, filling in credentials, clicking the login button, and interacting with a 'Rejected' card to edit it. The 'Metadata' section shows the total duration of the test was 3.75 seconds. To the right of the timeline, a screenshot of the application interface is displayed, showing a 'Monday day' dashboard with a 'Rejected' card and other portfolio items. At the bottom of the interface, there are tabs for 'Locator', 'Source', 'Call', 'Log', 'Errors', 'Console', 'Network', 'Attachments', 'Annotations', and 'TESTING OPTIONS'.

(E2E-005 Test Run Screenshot/Output image)

- E2E-006-Verify Portfolio Workflow ทำ E2E (Playwright) Test โดยใช้คัดลับนี้
  - Admin Advisor Verify Portfolio (Approved)

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

const PASSWORD = '123';
async function login(page: any, email: string) {
  await page.goto('http://localhost:5000/regis/login');

  await page.getByPlaceholder(/email/i).fill(email);
  await page.getByPlaceholder(/password/i).fill(PASSWORD);
  await page.getByRole('button', { name: /log in/i }).click();
}

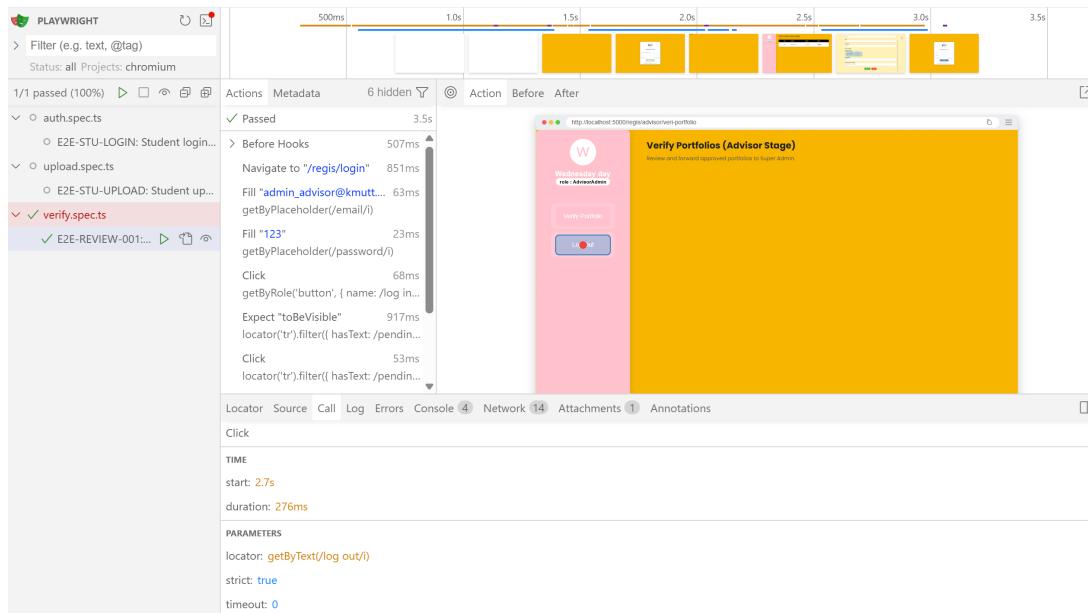
async function loginAsAdvisor(page: any) {
  await login(page, 'super_admin@kmutt.ac.th');
}

test('E2E-REVIEW-001: Advisor reviews approves portfolio', async ({ page }) => {

  await loginAsAdvisor(page);
  const pendingRow = page.locator('tr', { hasText: /pending/i }).first();
  await expect(pendingRow).toBeVisible();

  await pendingRow.getByRole('button', { name: /review/i }).click();

  await page.getByRole('button', { name: /send to super|approve|submit/i }).click();
  await page.getByText(/log out/i).click();
});
```



(E2E-006 Test Run Screenshot/Output image)

○ Admin Advisor Verify Portfolio (Rejected)

```

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

const PASSWORD = '123';
async function login(page: any, email: string) {
  await page.goto('http://localhost:5000/regis/login');

  await page.getByPlaceholder(/email/i).fill(email);
  await page.getByPlaceholder(/password/i).fill(PASSWORD);
  await page.getByRole('button', { name: '/log in/i'}).click();
}

async function loginAsAdvisor(page: any) {
  await login(page, 'admin_advisor@kmutt.ac.th');
}

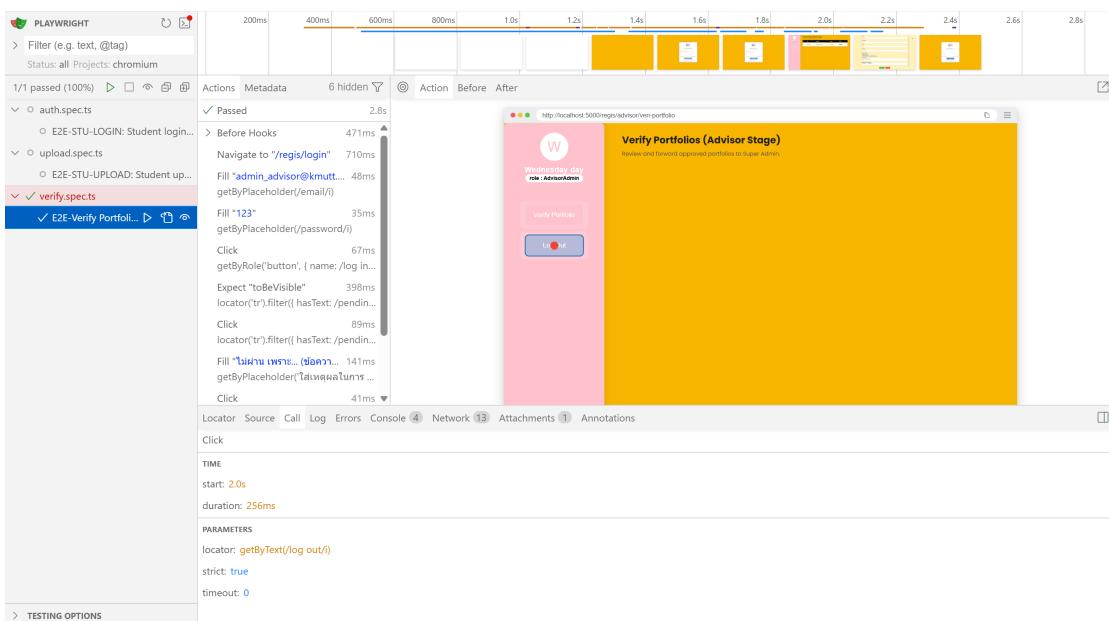
test('E2E-Verify Portfolio Workflow-005: Admin Advisor Verify portfolio (Reject)', async ({ page }) => {

  await loginAsAdvisor(page);
  const pendingRow = page.locator('tr', { hasText: /pending/i }).first();
  await expect(pendingRow).toBeVisible();

  await pendingRow.getByRole('button', { name: '/review/i'}).click();
  await page.getByPlaceholder('ใส่เหตุผลในการ Reject (ปั๊งบัน)').fill('ไม่เข้า เพระ... (ข้อความล้อป่าง)');
  await page.getByRole('button', { name: '/reject/i'}).click();

  await page.getByText('/log out/i').click();
});

```



(E2E-006 Test Run Screenshot/Output image)

○ Super Admin Verify Portfolio (Approved)

```

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

const PASSWORD = '123';
async function login(page: any, email: string) {
    await page.goto('http://localhost:5000/regis/login');

    await page.getByPlaceholder('/email/i').fill(email);
    await page.getByPlaceholder('/password/i').fill(PASSWORD);
    await page.getByRole('button', { name: '/log in/i'}).click();
}

async function loginAsAdvisor(page: any) {
    await login(page, 'super_admin@kmutt.ac.th');
}

test('E2E-REVIEW-001: Super Admin approves portfolio', async ({ page }) => {

    await loginAsAdvisor(page);
    const pendingRow = page.locator('tr', { hasText: /In_process/i }).first();
    await expect(pendingRow).toBeVisible();

    await pendingRow.getByRole('button', { name: '/View/i'}).click();
    await page.getByRole('button', { name: '/approve/i'}).click();

    await page.getByText('/log out/i').click();
});

```

The screenshot shows the Playwright Test Run interface. At the top, there's a timeline with various actions and their execution times. Below the timeline is a list of test files and their execution details. On the right, a browser window displays a 'Final Approval (Super Admin)' page. The page shows a table of pending portfolios for review. Each row in the table includes columns for Title, Student, Status, File, and Action (with a 'View' button). The browser window also has its own navigation and search bars at the top.

#	Title	Student	Status	File	Action
1	xxxx	Monday day	In_process	<a href="#">http://eegs-production-coal4.up.railway.app/uploads/1765019789602-portfoliles.png</a>	<a href="#">View</a>
2	ky	Monday day	In_process	<a href="#">http://eegs-production-coal4.up.railway.app/uploads/1765019789684-portfoliles.png</a>	<a href="#">View</a>
3	xxx	Monday day	In_process	<a href="#">http://eegs-production-coal4.up.railway.app/uploads/1765013302988-portfoliles.png</a>	<a href="#">View</a>
4	Health and science	Monday day	In_process	<a href="#">http://eegs-production-coal4.up.railway.app/uploads/1764935586242-portfoliles.pdf</a>	<a href="#">View</a>

(E2E-006 Test Run Screenshot/Output image)

## ○ Super Admin Verify Portfolio (Rejected)

```

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

const PASSWORD = '123';
async function login(page: any, email: string) {
  await page.goto('http://localhost:5000/regis/login');

  await page.getByPlaceholder('/email/i').fill(email);
  await page.getByPlaceholder('/password/i').fill(PASSWORD);
  await page.getByRole('button', { name: /log in/i }).click();
}

async function loginAsAdvisor(page: any) {
  await login(page, 'super_admin@kmutt.ac.th');
}
}
test['E2E-REVIEW-001: Super Admin Verify portfolio (Reject)', async ({ page }) => {

  await loginAsAdvisor(page);
  const pendingRow = page.locator('tr', { hasText: /In_process/i }).first();
  await expect(pendingRow).toBeVisible();

  await pendingRow.getByRole('button', { name: /review/i }).click();
  await page.getByPlaceholder('ไม่เห็นด้วย Reject (มีผู้คน)').fill('ไม่เห็น เพราะ... (ขอความด้วยอย่าง)');
  await page.getByRole('button', { name: /reject/i }).click();

  await page.getByText(/log out/i).click();
}];

```

The screenshot shows the Playwright Test Run interface. On the left, a tree view lists test files: auth.specs, upload.specs, verify.specs (which contains the E2E-REVIEW-001 test), and E2E-STU-LOGIN: Student login... The main area displays a timeline of actions:

- Passed (3.4s): Fill "123" (29ms)
- Click (76ms): getByPlaceholder(/password/i)
- Click (76ms): getByRole('button', { name: /log in/i })
- Expect "toBeVisible" (918ms): locator('tr').filter({ hasText: /In\_proc... })
- Click (43ms): locator('tr').filter({ hasText: /In\_proc... })
- Fill "ไม่เห็น เพราะ... (มีผู้คน)" (186ms): getByPlaceholder("ไม่เห็นด้วย Reject (มีผู้คน)")
- Click (36ms): getByRole('button', { name: /reject/i })
- Click (243ms): getByText(/log out/i)

Below the timeline, a section titled "After Hooks" shows a Click action with a duration of 36ms, starting at 2.6s. At the bottom, parameters for the Click action are listed: locator: getByRole('button', { name: /reject/i }), strict: true, and timeout: 0.

On the right, a browser screenshot shows a form titled "http://localhost:5000/regis/supervise/review/6033736f0aef7b6fca9400". The form fields include Title (xx), University (KMITT), Year (2020), Category (Test Novice), Cover Image (Cover Attached Files: 1769013302488-portfolioFile.jpg), Description (description), and Feedback (for Reject: ไม่เห็นด้วย (ขอความด้วยอย่าง)). Buttons for Approve and Reject are visible at the bottom.

(E2E-006 Test Run Screenshot/Output image)

- E2E-008-Search & Public Feed ทำ E2E (Playwright) Test โดยใช้โค้ดดังนี้

```

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

async function loginAsStudent(page: Page) {
    await page.goto('http://localhost:5000/regis/login');

    await page.getByPlaceholder('/email/i').fill('student@kmutt.ac.th');
    await page.getByPlaceholder('/password/i').fill('123');

    await page.getByRole('button', { name: '/log in/i'}).click();
    await page.waitForURL('**/regis/student/home**', { timeout: 10000 });
}

test('E2E-STU-SEARCH: Student can search portfolio on home page', async ({ page }) => {
    const KEYWORD = 'DB';

    await loginAsStudent(page);

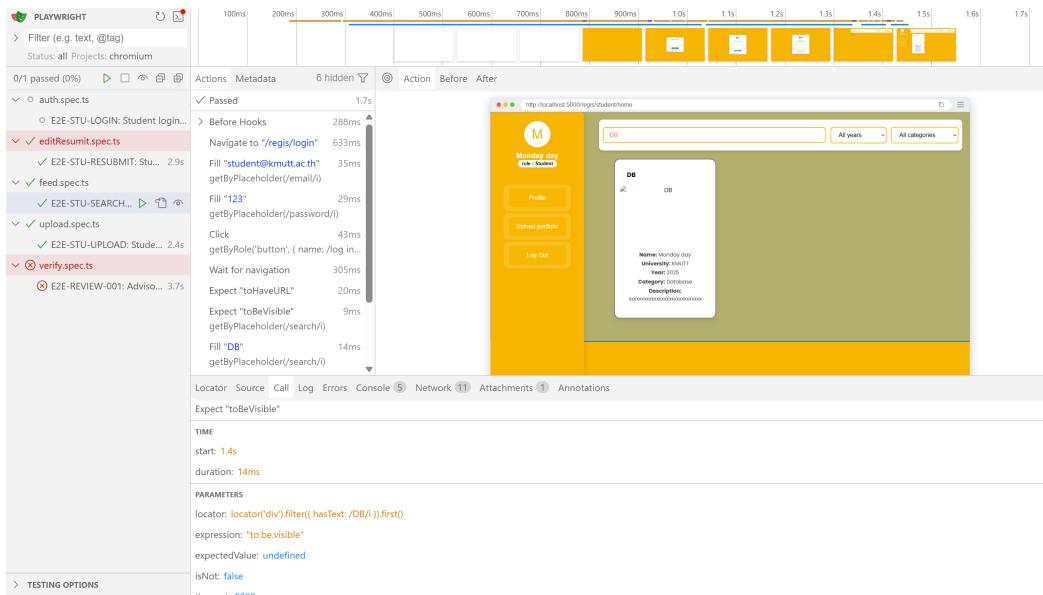
    await expect(page).toHaveURL('/regis/student/home');

    const searchInput = page.getByPlaceholder('/search/i');
    await expect(searchInput).toBeVisible();

    await searchInput.fill(KEYWORD);
    await searchInput.press('Enter');

    const resultCard = page.locator('div', { hasText: new RegExp(KEYWORD, 'i') }).first();
    await expect(resultCard).toBeVisible();
});

```



(E2E-008 Test Run Screenshot/Output image)

- E2E-009-Filter (Year/Category) ทำ E2E (Playwright) Test โดยใช้โค้ดดังนี้

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

async function loginAsStudent(page: Page) {
  await page.goto('http://localhost:5000/regis/login');

  await page.getByPlaceholder('/email/i').fill('student@kmutt.ac.th');
  await page.getByPlaceholder('/password/i').fill('123');
  await page.getByRole('button', { name: '/log in/i'}).click();

  await page.waitForURL('**/regis/student/home**', { timeout: 10000 });
}

test('E2E-FILTER-HOME: filter by year & category', async ({ page }) => {
  await loginAsStudent(page);
  await expect(page).toHaveURL('/regis/student/home');

  const yearSelect = page.locator('select').nth(0);
  await yearSelect.selectOption({ label: '2025' });

  const categorySelect = page.locator('select').nth(1);
  await categorySelect.selectOption({ label: 'Database' });

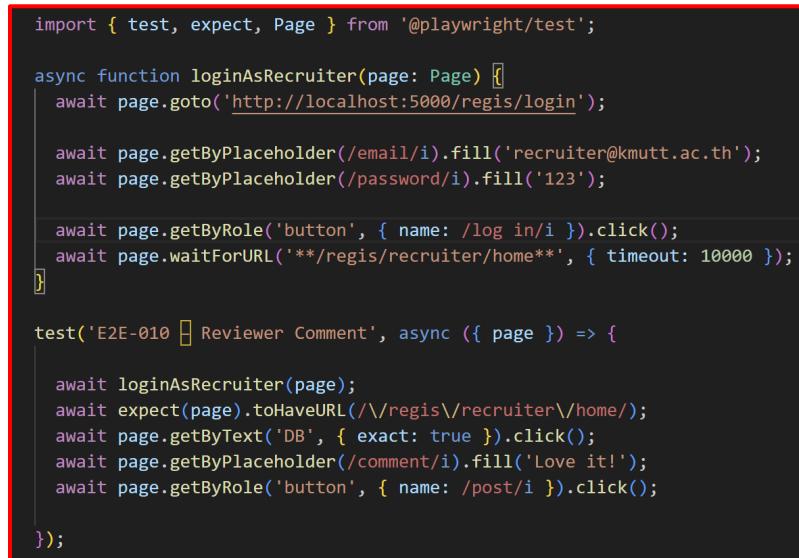
  const filteredCards = page
    .locator('div', { hasText: /2025/i })
    .filter({ hasText: /Database/i });

  await expect(filteredCards.first()).toBeVisible();
});
```

The screenshot shows the Playwright UI for a test run. At the top, there's a timeline with a yellow bar indicating the duration of each action. Below the timeline is a tree view of test results. A specific test, 'E2E-FILTER-HOME: filter by year & category', is expanded, showing its steps and assertions. To the right of the tree view is a browser screenshot of a login page. The user has entered 'student@kmutt.ac.th' and '123' into their respective fields. The dropdown menus for 'Year' (set to '2025') and 'Category' (set to 'Database') are open. The browser also displays the URL 'http://localhost:5000/regis/student/home'. At the bottom of the interface, there are sections for 'TIME', 'PARAMETERS', and 'TESTING OPTIONS'.

(E2E-009 Test Run Screenshot/Output image)

- E2E-010-Reviewer Comment ทำ E2E (Playwright) Test โดยใช้โค้ดดังนี้



```

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

async function loginAsRecruiter(page: Page) {
    await page.goto('http://localhost:5000/regis/login');

    await page.getByPlaceholder('/email/i').fill('recruiter@kmutt.ac.th');
    await page.getByPlaceholder('/password/i').fill('123');

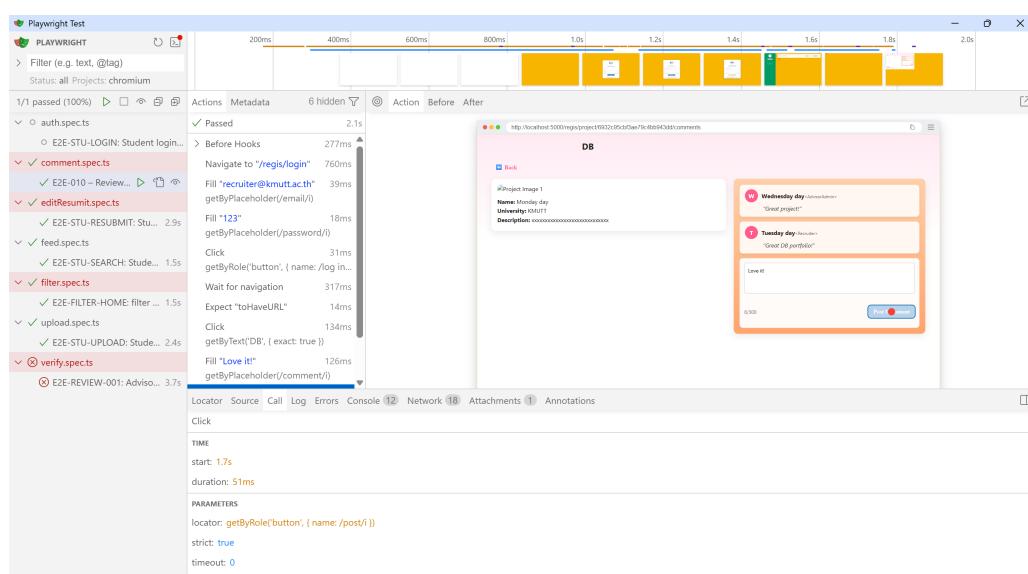
    await page.getByRole('button', { name: '/log in/i'}).click();
    await page.waitForURL('**/regis/recruiter/home**', { timeout: 10000 });
}

test('E2E-010 - Reviewer Comment', async ({ page }) => {

    await loginAsRecruiter(page);
    await expect(page).toHaveURL('/regis/recruiter/home');
    await page.getText('DB', { exact: true }).click();
    await page.getByPlaceholder('/comment/i').fill('Love it!');
    await page.getByRole('button', { name: '/post/i'}).click();

});

```



The screenshot shows the Playwright UI test runner interface. On the left, a tree view lists various test files and their execution status. A red box highlights the 'comment.spec.ts' file under the 'comment.spec.ts' folder, which contains the failing test case. On the right, a timeline shows the execution of the test steps, with a red box highlighting the step 'Expect "toHaveURL"' which failed at 1.7s. Below the timeline, detailed logs for the failed step are shown, including the locator used ('locator: getByRole(button, { name: /post/i })'), strict mode ('strict: true'), and timeout ('timeout: 0'). To the right of the timeline, a screenshot of the application's database interface is displayed, showing a table with two rows: 'Wednesday day' and 'Tuesday day', both with the description 'Great DB portfolio'.

(E2E-010 Test Run Screenshot/Output image)

- E2E-011-Visibility Setting (Public/Private) ทำ E2E (Playwright) Test โดยใช้โค้ดดังนี้

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

async function loginAsStudent(page: Page) {
    await page.goto('http://localhost:5000/regis/login');

    await page.getByPlaceholder('/email/i').fill('student@kmutt.ac.th');
    await page.getByPlaceholder('/password/i').fill('123');

    await page.getByRole('button', { name: '/log in/i }).click();
    await page.waitForURL('**/regis/student/**', { timeout: 1000 });
}

test('E2E-011 □ Visibility Setting (Public / Private)', async ({ page }) => {
    await loginAsStudent(page);

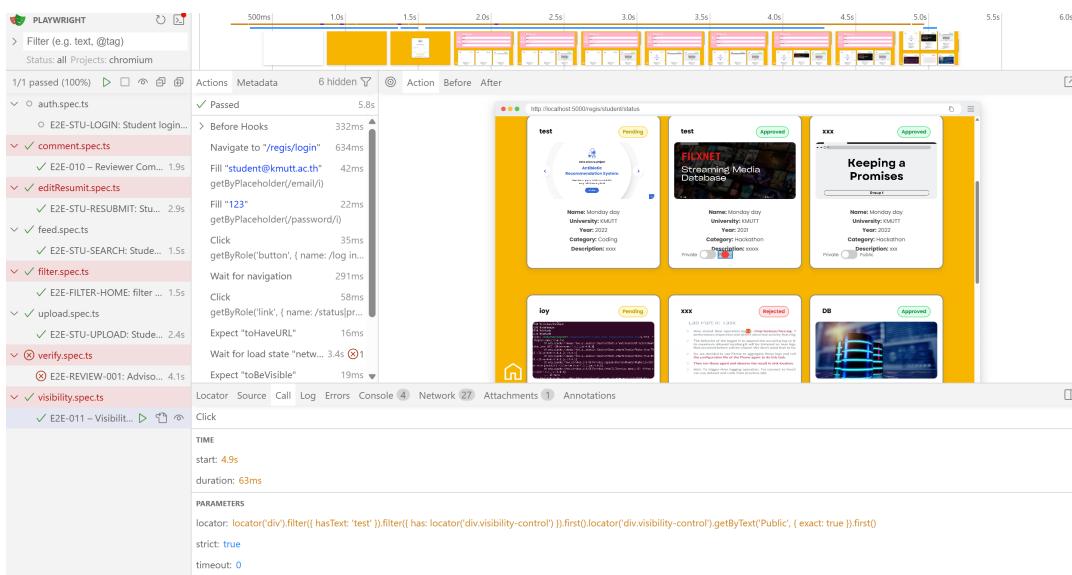
    await page.getByRole('link', { name: '/status|profile/i }).click();
    await expect(page).toHaveURL('/regis/student/status');

    await page.waitForLoadState('networkidle');

    const targetCard = page
        .locator('div', { hasText: 'test' })
        .filter({ has: page.locator('div.visibility-control') })
        .first();

    await expect(targetCard).toBeVisible();
    const publicButton = targetCard
        .locator('div.visibility-control')
        .getByText('Public', { exact: true })
        .first();

    await publicButton.click();
});
```



(E2E-011 Test Run Screenshot/Output image)

- E2E-012 – Logout

```
const { test, expect } = require('@playwright/test');
// =====
// TEST CASE: Logout (ตรวจสอบผลการเบลียนหน้า)
// =====
test('TC2: Logout successfully (UI Check)', async ({ page }) => {

    // -----
    // STEP 1: Pre-condition (Login เข้ามา ก่อน)
    // -----
    await page.goto('/studentport/login');
    await page.fill('input[type="email"]', "student@kmutt.ac.th");
    await page.fill('input[type="password"]', "123");

    page.once('dialog', async dialog => await dialog.accept());
    await page.click('.btn');

    // ตรวจสอบว่า เข้าหน้า Home ได้แล้ว
    await expect(page).toHaveURL(/.*student\/home/);

    // -----
    // STEP 2: กดปุ่ม Logout
    // -----
    console.log("--- Action: Clicking Logout ---");

    const logoutBtn = page.locator('text=Log out');

    // รอให้ปุ่มเข้มมา ก่อน
    await expect(logoutBtn).toBeVisible();

    // ตักจับ Alert (ເຜື່ອສີ)
    page.once('dialog', async dialog => {
        console.log(`Logout Dialog: ${dialog.message()}`);
        await dialog.accept();
    });

    // กดปุ่ม Logout
    await logoutBtn.click();

    // -----
    // STEP 3: ตรวจสอบผลลัพธ์ (ເຫັນວ່າ ກຳລັນມາหน้า Login ໄແນ)
    // -----

    // 1. ຮອດ URL ເປົ້ານກຳລັນເປັນหน้า Login
    await expect(page).toHaveURL(/.*login/, { timeout: 10000 });

    // 2. ເຫັນວ່າເຫັນປຸ່ມ Login ພິຈີ້ອ່ອງກາຣ໌ທັສຳຜ່ານໂພລ໌ຂຶ້ນມາແລ້ວ (ຍືນຍັນວ່າ ຍູ້ໜ້າ Login ຈົດ)
    await expect(page.locator('.btn')).toBeVisible(); // ພິຈີ້ອ່ອງເຫັນ input ຕີ່ໄດ້

    console.log("✅ TC2: Logout Successful (Redirected to Login Page)");
});
```

PLAYWRIGHT

> Filter (e.g. text, @tag)  
Status: all Projects: chromium

1/1 passed (100%)

Action	Metadata	6 hidden	Before	After
✓ Passed		5.0s		
locator('btn')				
Expect "toHaveURL"	858ms			
Accept dialog	25ms △4			
Expect "toBeVisible"	3ms			
locator('text=Log out')				
<b>Click locator('text=Log out')</b>	76ms			
Expect "toHaveURL"	1ms			
Expect "toBeVisible"	2ms			
locator('btn')				
> After Hooks	129ms			

Network 13 Attachments Annotations

Name	Method	Status	Content Type	Duration	Size	Start	Route
<b>login</b>	GET	<b>404</b>	text/html	<b>1.6s</b>	720	<b>596ms</b>	
main.a283fba5.js	GET	200	application/javascript	2.2s	105.0K	1.5s	
main.1a722b82.css	GET	200	text/css	761ms	6.1K	1.5s	
css?2family=Poppins:wght@...	GET	200	text/css	482ms	1.0K	2.3s	
studentport_logo.431b9067c...	GET	200	image/png	504ms	51.3K	3.8s	
pxiByp8kv8JHgFvRLj6Z1xIf...	GET	200	font/woff2	337ms	7.8K	3.8s	
pxiEyp8kv8JHgFvRjJfecg.w...	GET	200	font/woff2	86ms	7.7K	3.8s	
login	POST	200	application/json	614ms	451	3.9s	
public	GET	200	application/json	141ms	3.6K	4.6s	
pxiByp8kv8JHgFvRLc7Z1xF...	GET	200	font/woff2	83ms	7.7K	4.7s	
1765022896241-cover_img...	GET		x-unknown	-	-	4.7s	
1764936027494-cover_img.j...	GET		x-unknown	-	-	4.7s	

> TESTING OPTIONS  
> SETTINGS

(E2E-012 Test Run Screenshot/Output image)