

การยกระดับกระบวนการทดสอบซอฟต์แวร์ด้วยเครื่องมืออัตโนมัติ Katalon Studio กรณีตัวอย่างระบบบริหารจัดการการให้บริการในแผนกผู้ป่วยนอกโรงพยาบาลศิริราช

ศุภรัช จามจรีกุล* และ ธนบดีนทร์ เขียรเจริญธนกิจ

ฝ่ายสารสนเทศ คณะแพทยศาสตร์ศิริราชพยาบาล มหาวิทยาลัยมหิดล กรุงเทพมหานคร 10700

*อีเมลของผู้ประพันธ์บทความ: suppatuch.jam@mahidol.ac.th

บทคัดย่อ

การทดสอบซอฟต์แวร์ (Software Testing) ถือเป็นกระบวนการสำคัญในวงจรการพัฒนาซอฟต์แวร์ เพื่อให้มั่นใจในคุณภาพและประสิทธิภาพการทำงานของซอฟต์แวร์ให้เป็นไปตามความต้องการที่กำหนดไว้ โดยเฉพาะในยุคปัจจุบันที่ซอฟต์แวร์มีโครงสร้างและฟังก์ชันการทำงานที่มีความซับซ้อนมากขึ้น การทดสอบด้วยมนุษย์ (Manual Testing) อาจไม่สามารถตอบสนองความต้องการในด้านความเร็วและความแม่นยำได้อย่างเพียงพอ การนำเครื่องมือทดสอบอัตโนมัติ (Automated Testing Tools) มาใช้ในการทดสอบซอฟต์แวร์ จึงเป็นแนวทางที่ได้รับความนิยมและถูกนำมาใช้อย่างแพร่หลายโดยบทความนี้จะกล่าวถึงประสิทธิภาพของเครื่องมืออัตโนมัติในการทดสอบซอฟต์แวร์โดยใช้ Katalon Studio เป็นเครื่องมือหลักในการทดสอบ ผ่านกรณีตัวอย่าง ระบบบริหารจัดการการให้บริการในแผนกผู้ป่วยนอกของโรงพยาบาลศิริราช ซึ่งดำเนินการทดสอบซ้ำ 10 รอบ จากผลการทดสอบพบว่าเครื่องมือ Katalon Studio ใช้เวลาเฉลี่ย 37.24 วินาที ในขณะที่การทดสอบโดยมนุษย์ใช้เวลาเฉลี่ย 120.6 วินาที แสดงให้เห็นว่าการใช้เครื่องมือทดสอบอัตโนมัติสามารถลดระยะเวลาการดำเนินการได้อย่างมีนัยสำคัญ ในบทความนี้ยังวิเคราะห์ข้อดีและข้อจำกัดของเครื่องมือทดสอบอัตโนมัติเปรียบเทียบกับ การทดสอบด้วยมนุษย์ รวมถึงอภิปรายสถานการณ์ที่ต้องการความยืดหยุ่นและการวิเคราะห์เชิงลึก เพื่อเสนอแนวทางในการผสมผสานทั้งสองรูปแบบให้ทำงานร่วมกันอย่างมีประสิทธิภาพ และยกระดับกระบวนการทดสอบให้ครอบคลุมและแม่นยำยิ่งขึ้น

คำสำคัญ: การทดสอบซอฟต์แวร์, การทดสอบด้วยมนุษย์, เครื่องมือการทดสอบอัตโนมัติ, ระบบบริหารจัดการผู้ป่วยนอก, วงจรชีวิตการทดสอบซอฟต์แวร์

Enhancing the Software Testing Process with Katalon Studio: A Case Study of the Outpatient Service Management System at Siriraj Hospital

Suppatuch Jamjareekul* and Tanabodin Chiencharoentanakij

Siriraj Information Technology Department, Faculty of Medicine Siriraj Hospital, Mahidol University, Bangkok 10700, Thailand

*Corresponding author's e-mail: Suppatuch.jam@mahidol.ac.th

Abstract

Software testing is a pivotal process in the software development lifecycle to guarantee the quality and performance of software in accordance with specified requirements. As modern software becomes increasingly complex in structure and functionality, manual testing may not adequately meet the demands for speed and accuracy. The adoption of automated testing tools has therefore gained significant traction and is widely implemented. This article examines the effectiveness of automated testing tools by utilizing Katalon Studio as the primary testing tool in a case study involving the outpatient service management system at Siriraj Hospital. The study was conducted over 10 test iterations, revealing that Katalon Studio achieved an average execution time of 37.24 seconds, whereas manual testing required an average of 120.6 s. This result demonstrates that automated testing tools can significantly reduce execution time. Furthermore, this article analyzes the advantages and limitations of automated testing compared to manual testing, which remains the standard practice in various scenarios. It also explores situations that require greater flexibility and in-depth analysis, proposing an integrated approach that combines both testing methods to enhance efficiency and improve the accuracy and comprehensiveness of the software testing process.

Keywords: Software testing, Manual testing, Automated testing tools, Outpatient management system, Software testing life cycle

บทนำ

ในยุคปัจจุบันที่เทคโนโลยีสารสนเทศและการสื่อสารมีการพัฒนาอย่างรวดเร็ว ซอฟต์แวร์ได้กลายเป็นองค์ประกอบสำคัญในหลายภาคส่วนของชีวิตประจำวัน ตั้งแต่ธุรกิจ การศึกษา ไปจนถึงการดูแลสุขภาพ ส่งผลให้กระบวนการพัฒนาซอฟต์แวร์ต้องคำนึงถึงคุณภาพ ความแม่นยำ และความถูกต้องของระบบซอฟต์แวร์ ซึ่งระบบบริหารจัดการการให้บริการในแผนกผู้ป่วยนอก เป็นระบบสารสนเทศที่มีความสำคัญต่อโรงพยาบาลศิริราช เพราะถูกออกแบบมาเพื่อช่วยเพิ่มประสิทธิภาพในการบริหารจัดการผู้ป่วยนอก โดยมุ่งเน้นการลดระยะเวลาการรอคอย เพิ่มความสะดวกสบายให้แก่ผู้ป่วยนอก และช่วยให้บุคลากรทางการแพทย์สามารถทำงานได้อย่างเป็นระบบมากขึ้น ฟังก์ชันหลักของระบบนี้คือ ระบบคิวแบบไดนามิกซึ่งสามารถทำได้ทั้งในรูปแบบออนไลน์และออฟไลน์ การลงทะเบียนนัดหมายผู้ป่วย การนัดหมายแพทย์ การเรียกผู้ป่วยเข้าห้องตรวจ การจัดการข้อมูลเวชระเบียน ไปจนถึงการออกใบเสร็จและการตรวจสอบสิทธิประกันสุขภาพ เป็นต้น

การทดสอบซอฟต์แวร์ (Software Testing) เป็นกระบวนการสำคัญที่ช่วยมั่นใจว่าซอฟต์แวร์ที่พัฒนาขึ้นมีความถูกต้อง ความสมบูรณ์ มีความปลอดภัย และมีคุณภาพที่ดีตรงตามความต้องการของผู้ใช้งาน (Jinda & Amornmontien, 2023) โดยมุ่งเน้นการตรวจสอบและเปิดเผยข้อบกพร่องที่แฝงอยู่ในซอฟต์แวร์ เพื่อให้สามารถระบุข้อบกพร่องและแก้ไขได้ทันเวลาที่ (Kumar & Mishra, 2016) อย่างไรก็ตามจากการศึกษาพบว่า การทดสอบด้วยมนุษย์ (Manual Testing) แม้ว่าจะมีประสิทธิภาพในการตรวจพบข้อบกพร่องที่ซับซ้อนสูง แต่กลับมีข้อจำกัดสำคัญด้านระยะเวลาและต้นทุนที่สูงเกินไป โดยเฉพาะในระบบซอฟต์แวร์ขนาดใหญ่หรือมีความซับซ้อนมาก ซึ่งอาจส่งผลกระทบต่อการบริหารจัดการทรัพยากรขององค์กรในระยะยาว (Thant & Tin, 2023)

ด้วยเหตุนี้ อุตสาหกรรมซอฟต์แวร์ในปัจจุบันจึงมีแนวโน้มที่จะนำเครื่องมือทดสอบอัตโนมัติ (Automated Testing Tools) มาใช้งานร่วมกับการทดสอบด้วยมนุษย์มากขึ้น เนื่องจากเครื่องมือเหล่านี้สามารถช่วยลดเวลาและเพิ่มความแม่นยำในการตรวจจับข้อผิดพลาดของซอฟต์แวร์ และดำเนินการทดสอบซ้ำได้อย่างรวดเร็ว โดยมีเครื่องมือที่หลากหลายสำหรับทดสอบในแต่ละสภาพแวดล้อม (Mahajan et al., 2016) อย่างไรก็ตาม เครื่องมือทดสอบอัตโนมัติก็มีข้อจำกัดบางประการ เช่น ขาดความยืดหยุ่น และไม่สามารถทดสอบส่วนที่ต้องการการวิเคราะห์เชิงลึกได้ดีเท่าการทดสอบด้วยมนุษย์ ดังนั้น การผสมผสานจุดแข็งของทั้งสองวิธีเข้าด้วยกัน จึงเป็นแนวทางสำคัญที่จะช่วยยกระดับกระบวนการทดสอบให้มีประสิทธิภาพ และตอบสนองความต้องการที่ซับซ้อนของอุตสาหกรรมซอฟต์แวร์ในปัจจุบันได้ดีที่สุด

ความหมายของการทดสอบซอฟต์แวร์ (Software Testing)

การทดสอบซอฟต์แวร์ เป็นกระบวนการที่ช่วยให้อมั่นใจว่าซอฟต์แวร์ที่พัฒนาขึ้นมีความถูกต้อง สมบูรณ์ และสามารถทำงานได้อย่างมีประสิทธิภาพ ตรงตามความต้องการของผู้ใช้งาน เพื่อให้อมั่นใจในคุณภาพก่อนที่จะส่งมอบให้ลูกค้า (Iamkhajornchai, 2023) โดยเป้าหมายหลักของการทดสอบนี้ไม่เพียงแต่ช่วยค้นหาข้อผิดพลาดหรือข้อบกพร่อง (Defects) ที่อาจส่งผลกระทบต่อการทำงานของซอฟต์แวร์ แต่ยังช่วยให้สามารถปรับปรุงและแก้ไขได้อย่างทันเวลาที่ เพื่อปรับปรุงคุณภาพของซอฟต์แวร์ให้เป็นไปตามมาตรฐานอุตสาหกรรม

โดยทั่วไปการทดสอบระบบซอฟต์แวร์มีการพัฒนากระบวนการเป็นแนวคิดหลักในการทำทดสอบซอฟต์แวร์ ซึ่งแบบดั้งเดิมคือ วงจรชีวิตการทดสอบซอฟต์แวร์ (Software Testing Life Cycle: STLC) เป็นขั้นตอนพื้นฐานการทำทดสอบด้วยมนุษย์ และ วงจรชีวิตการทดสอบอัตโนมัติ (Automated Testing Life Cycle: ATLC) ที่ถูกพัฒนาขึ้นต่อมาโดยมีการนำเครื่องมือการทดสอบอัตโนมัติเข้ามาช่วยเพิ่มประสิทธิภาพ (Skooldio, 2024)

กรณีทดสอบ (Test Cases) คือ การสร้างสถานการณ์หรือเงื่อนไขต่าง ๆ เพื่อใช้ในการทดสอบระบบซอฟต์แวร์ โดยออกแบบให้สอดคล้องกับความต้องการ (Requirements) และเงื่อนไขการใช้งาน (Conditions) ที่กำหนดไว้ เป้าหมายของกรณีทดสอบไม่ใช่แค่การตรวจสอบว่าฟังก์ชันต่าง ๆ ในระบบสามารถทำงานได้หรือไม่ แต่ยังรวมถึงการตรวจสอบว่าระบบทำงานได้ตรงตามความต้องการและเงื่อนไขที่กำหนดไว้ด้วย (Jobsdb, 2023)

ความหมายของ Software Testing Life Cycle : STLC และ Automated Testing Life Cycle : ATLC

วงจรชีวิตการทดสอบซอฟต์แวร์ (Software Testing Life Cycle: STLC) เป็นกระบวนการที่มุ่งเน้นการทดสอบซอฟต์แวร์เพื่อระบุข้อบกพร่องหรือข้อผิดพลาด เมื่อตรวจพบข้อผิดพลาด จะมีการแจ้งให้ผู้พัฒนาทราบเพื่อดำเนินการแก้ไขและปรับปรุงซอฟต์แวร์ให้มีคุณภาพและประสิทธิภาพสูงสุด ก่อนส่งมอบและนำไปใช้งานจริงในสภาวะแวดล้อมใช้งานจริง (Production Environment) วงจรชีวิตการทดสอบซอฟต์แวร์ประกอบด้วยหลายขั้นตอนที่ช่วยให้การทดสอบเป็นระบบและมีประสิทธิภาพมากขึ้น (ภาพที่ 1) ดังนี้

1. การวิเคราะห์ความต้องการ (Requirements Analysis) ขั้นตอนแรกทีมทดสอบจะทำการวิเคราะห์ความต้องการของซอฟต์แวร์ที่ได้รับมอบหมาย โดยการวิเคราะห์นี้เป็นขั้นตอนสำคัญเพื่อให้ทราบถึงขอบเขตและรายละเอียดของระบบซอฟต์แวร์ที่ต้องทดสอบ ซึ่งประกอบด้วยข้อตกลงที่กำหนดไว้ในเอกสารความต้องการ (Requirements Document) การวิเคราะห์ความต้องการอย่างละเอียดช่วยให้ทีมทดสอบสามารถออกแบบกรณีทดสอบ (Test Cases) ที่ครอบคลุมและตรงตามความต้องการของผู้ใช้งาน

2. การวางแผนการทดสอบ (Test Planning) ในขั้นตอนนี้ทีมทดสอบจะวางแผนกลยุทธ์การทดสอบโดยระบุแนวทางและวิธีการทดสอบให้เป็นระบบและมีประสิทธิภาพ ตัวอย่างเช่น การกำหนดวัตถุประสงค์ของการทดสอบ การระบุทรัพยากรที่จำเป็น (เช่น บุคลากร อุปกรณ์ และเครื่องมือ) การกำหนดระยะเวลา และการจัดสรรงบประมาณ ทั้งนี้การวางแผนที่ดีจะช่วยให้กระบวนการทดสอบเป็นไปอย่างราบรื่นและบรรลุเป้าหมายที่ตั้งไว้

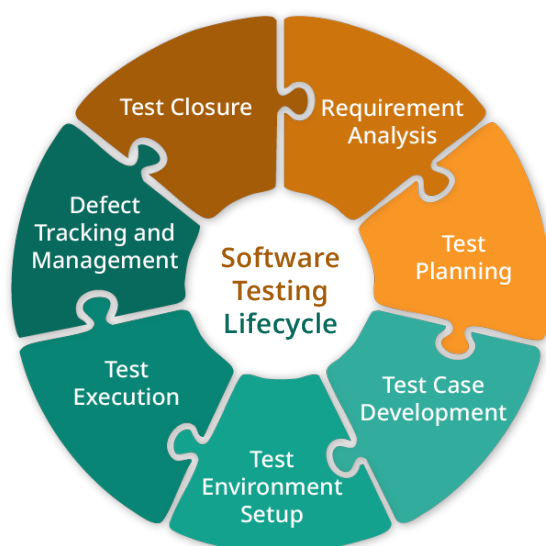
3. การพัฒนากรณีทดสอบ (Test Case Development) ในขั้นตอนนี้ทีมทดสอบจะเริ่มทำการพัฒนากรณีทดสอบ (Test Cases) ซึ่งเป็นเครื่องมือสำคัญในการทดสอบซอฟต์แวร์ โดยกรณีทดสอบจะถูกออกแบบมาเพื่อตรวจสอบว่าซอฟต์แวร์ทำงานได้ตามที่คาดหวังหรือไม่ การพัฒนากรณีทดสอบที่ครอบคลุมและมีประสิทธิภาพจะช่วยให้สามารถตรวจจับข้อผิดพลาดได้อย่างแม่นยำ

4. การเตรียมสภาพแวดล้อมทดสอบ (Test Environment Setup) ในขั้นตอนนี้เกี่ยวข้องกับเตรียมสภาพแวดล้อมที่เหมาะสมสำหรับการทดสอบซอฟต์แวร์ เช่น การติดตั้งโปรแกรม การเตรียมฐานข้อมูล การตั้งค่าอุปกรณ์ และการจัดเตรียมเครื่องมือที่จำเป็น โดยการเตรียมสภาพแวดล้อมทดสอบที่ถูกต้องและใกล้เคียงกับสภาพแวดล้อมการใช้งานจริงจะช่วยให้ผลลัพธ์การทดสอบมีความน่าเชื่อถือ

5. การดำเนินการทดสอบ (Test Execution) ทีมทดสอบจะทำการทดสอบซอฟต์แวร์ตามแผนการทดสอบที่ได้วางไว้ โดยใช้กรณีทดสอบที่พัฒนาขึ้นมาเพื่อตรวจสอบการทำงานของระบบในสภาพแวดล้อมที่เตรียมไว้ ผลลัพธ์ที่ได้จากการทดสอบจะถูกบันทึกและวิเคราะห์เพื่อระบุข้อผิดพลาดหรือจุดที่ต้องปรับปรุง

6. การติดตามและบริหารจัดการข้อบกพร่อง (Defect Tracking and Management) ในขั้นตอนหลังจากดำเนินการทดสอบ ทีมทดสอบจะต้องติดตามข้อผิดพลาดที่พบ (Defects) โดยบันทึกรายละเอียดของปัญหา เช่น ประเภทของข้อผิดพลาด ระดับความรุนแรง (Severity) และลำดับความสำคัญ (Priority) เพื่อให้ทีมพัฒนาทราบถึงปัญหาที่ต้องแก้ไข

7. การปิดการทดสอบ (Test Closure) ขั้นตอนสุดท้ายของวงจรชีวิตการทดสอบซอฟต์แวร์ (STLC) คือการปิดการทดสอบ โดยทีมทดสอบจะทำการสรุปผลการทดสอบและจัดทำรายงานผลลัพธ์ รวมถึงข้อผิดพลาดที่พบให้กับทีมพัฒนาระบบเพื่อดำเนินการแก้ไขและปรับปรุงซอฟต์แวร์ต่อไป การปิดการทดสอบอย่างเป็นระบบช่วยให้สามารถประเมินประสิทธิภาพของกระบวนการทดสอบและนำไปสู่การพัฒนาปรับปรุงในอนาคต



ภาพที่ 1 ขั้นตอนการทำงานของ Software Testing Lift Cycle (STLC) (Skooldio, 2024)

วงจรชีวิตการทดสอบอัตโนมัติ (Automated Testing Life Cycle: ATLC) เป็นกระบวนการที่มุ่งเน้นการทดสอบซอฟต์แวร์โดยใช้เครื่องมืออัตโนมัติเพื่อลดภาระของการทดสอบด้วยมนุษย์และเพิ่มประสิทธิภาพการทดสอบ เมื่อทำการทดสอบด้วยเครื่องมืออัตโนมัติ ทีมทดสอบจะต้องเลือกเครื่องมือที่เหมาะสม เขียนสคริปต์การทดสอบ ดำเนินการทดสอบเพื่อหาข้อบกพร่อง โดยวงจรชีวิตการทดสอบอัตโนมัติ ประกอบด้วยหลายขั้นตอนที่ช่วยให้การทดสอบเป็นระบบและมีประสิทธิภาพมากขึ้น (ภาพที่ 2) ดังนี้

1. การเลือกเครื่องมือ (Tool Selection) ขั้นตอนแรก คือการเลือกเครื่องมือทดสอบอัตโนมัติที่เหมาะสมกับลักษณะของซอฟต์แวร์ที่จะทดสอบ ทีมทดสอบจะพิจารณาเครื่องมือที่สามารถรองรับความต้องการ เช่น ความสามารถในการทดสอบแบบครอบคลุม รองรับแพลตฟอร์มต่าง ๆ และความสะดวกในการใช้งาน

2. การพัฒนาสคริปต์ทดสอบ (Test Script Development) ในขั้นตอนนี้ ทีมทดสอบจะเขียนสคริปต์ทดสอบโดยใช้เครื่องมือที่เลือกไว้ การพัฒนาสคริปต์ทดสอบจะเป็นการกำหนดลำดับขั้นตอนที่เครื่องมืออัตโนมัติจะทำตามเพื่อทดสอบการทำงานของระบบ สคริปต์ต้องถูกออกแบบอย่างระมัดระวังเพื่อให้ครอบคลุมและสามารถตรวจสอบข้อผิดพลาดได้แม่นยำ

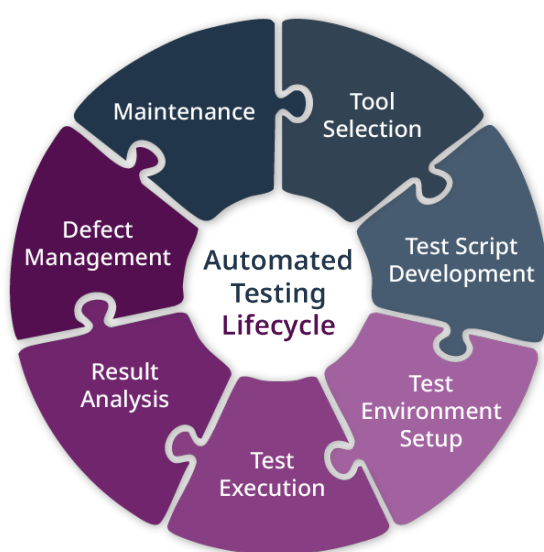
3. การเตรียมสภาพแวดล้อมทดสอบ (Test Environment Setup) ทีมทดสอบจะเตรียมสภาพแวดล้อมที่เหมาะสมสำหรับการทดสอบอัตโนมัติ เช่น การติดตั้งเครื่องมือที่จำเป็น การกำหนดการตั้งค่าฐานข้อมูล หรือการเตรียมเครื่องมืออื่น ๆ เพื่อให้สามารถทำการทดสอบได้อย่างมีประสิทธิภาพ

4. การดำเนินการทดสอบ (Test Execution) ทีมทดสอบจะรันสคริปต์ทดสอบบนระบบ โดยเครื่องมืออัตโนมัติจะทำการทดสอบซอฟต์แวร์ตามที่ได้ออกแบบไว้ ผลลัพธ์จากการทดสอบจะถูกบันทึกและนำมาวิเคราะห์ในขั้นตอนถัดไป

5. การวิเคราะห์ผลการทดสอบ (Result Analysis) หลังจากการดำเนินการทดสอบ ทีมทดสอบจะทำการวิเคราะห์ผลลัพธ์ที่ได้จากการทดสอบเพื่อระบุข้อผิดพลาดหรือจุดที่ต้องปรับปรุง การวิเคราะห์นี้ช่วยให้สามารถหาสาเหตุของข้อผิดพลาดได้และวางแผนการแก้ไข

6. การบริหารจัดการข้อบกพร่อง (Defect Management) ทีมทดสอบจะติดตามข้อบกพร่องที่พบจากการทดสอบ โดยบันทึกรายละเอียดเกี่ยวกับข้อบกพร่องนั้น เช่น ลำดับความสำคัญและประเภทของข้อผิดพลาด การบริหารจัดการข้อบกพร่องอย่างมีประสิทธิภาพจะช่วยให้สามารถแก้ไขได้รวดเร็วและมีประสิทธิผล

7. การบำรุงรักษา (Maintenance) การทดสอบอัตโนมัติไม่ใช่กระบวนการที่ทำเพียงครั้งเดียวเท่านั้น เมื่อมีการเปลี่ยนแปลงซอฟต์แวร์หรือฟีเจอร์ใหม่ ๆ เข้ามา ทีมทดสอบจะต้องทำการบำรุงรักษาสคริปต์ทดสอบให้ทันสมัยและสอดคล้องกับการเปลี่ยนแปลงเหล่านั้น ซึ่งการบำรุงรักษาจะช่วยให้ระบบทดสอบอัตโนมัติยังคงทำงานได้อย่างมีประสิทธิภาพและรองรับการเปลี่ยนแปลงในอนาคต



ภาพที่ 2 ขั้นตอนการทำงานของ Automated Testing Life Cycle (ATLC) (Skooldio, 2024)

รูปแบบของการทดสอบซอฟต์แวร์

การทดสอบด้วยมนุษย์ (Manual Testing) คือกระบวนการทดสอบซอฟต์แวร์โดยใช้ความคิดของผู้ทดสอบ เพื่อตรวจสอบความถูกต้อง, ความสมบูรณ์ของซอฟต์แวร์ (Senna Labs, 2023) โดยผู้ทดสอบจะทำตามแผนการทดสอบที่วางไว้และในตอนท้ายจะจำลองเสมือนว่าเป็นผู้ใช้งานระบบ ทำหน้าที่กรอกข้อมูล ทำรายการที่ละขั้นตอนตามกรณีทดสอบที่ได้ออกแบบไว้ ถ้าเป็นโครงการใหญ่จะส่งผลทำให้ใช้เวลาในการทำงานมากขึ้น เพราะการทดสอบด้วยมนุษย์ จะต้องใช้แรงงานบุคลากรที่ทำหน้าที่ทดสอบเข้ามาเพื่อทำการทดสอบระบบเองทีละเรื่องทีละขั้นตอน ส่งผลทำให้การทดสอบใช้เวลานาน

ข้อดีของการทดสอบด้วยมนุษย์ คือการทดสอบจะมีความยืดหยุ่นสูง ผู้ทดสอบสามารถปรับเปลี่ยนวิธีการทดสอบได้ตามสถานการณ์จริง โดยไม่ต้องพึ่งพาสคริปต์หรือเครื่องมือ เหมาะสำหรับการทดสอบเชิงคุณภาพ เช่น การทดสอบประสบการณ์ของผู้ใช้งานในด้านความรู้สึกที่ตอบสนองต่อการใช้งานซอฟต์แวร์ (User Experience) หรือการทดสอบการใช้งานจริงที่ต้องการความคิดสร้างสรรค์ มีต้นทุนเริ่มต้นต่ำ ไม่จำเป็นต้องลงทุนในเครื่องมือหรือซอฟต์แวร์พิเศษ

ข้อเสียของการทดสอบด้วยมนุษย์ คือการทดสอบจะใช้เวลานานเนื่องจากต้องอาศัยกำลังคนในการทดสอบทีละขั้นตอน โดยเฉพาะในระบบที่มีความซับซ้อน และมีความเสี่ยงต่อความผิดพลาดได้ง่าย เนื่องจากการทดสอบด้วยมนุษย์จะมีความเหนื่อยล้าหรือความไม่สม่ำเสมอในการทำงาน รวมถึงไม่สามารถทำการทดสอบข้อมูลที่มีจำนวนเยอะมากหลายพันข้อมูลได้เท่ากับการทดสอบอัตโนมัติ

การทดสอบอัตโนมัติ (Automated Testing) คือกระบวนการทดสอบโปรแกรมหรือระบบ โดยใช้เครื่องมือพิเศษที่ออกแบบมาเพื่อทำการทดสอบโดยอัตโนมัติ แทนที่การทดสอบด้วยมนุษย์ สามารถแบ่งออกเป็นสี่ประเภทหลักได้แก่ การทดสอบความน่าเชื่อถือ (Reliability Testing), การทดสอบความปลอดภัย (Security Testing), การทดสอบความถูกต้อง (Correctness Testing), และการทดสอบประสิทธิภาพ (Performance Testing) เครื่องมืออัตโนมัติช่วยให้สามารถดำเนินการทดสอบซ้ำได้อย่างรวดเร็วและมีความแม่นยำสูง โดยลดภาระของผู้ทดสอบและเพิ่มประสิทธิภาพในการตรวจสอบคุณภาพซอฟต์แวร์ (Mahajan et al., 2016)

อย่างไรก็ตาม การทดสอบอัตโนมัติจำเป็นต้องใช้เวลาในการเตรียมการ เช่น การเขียนสคริปต์, การกำหนดเงื่อนไขการทดสอบ และการเตรียมข้อมูลทดสอบ ดังนั้นการทดสอบอัตโนมัติอาจไม่เหมาะสมสำหรับโครงการขนาดเล็กหรือโครงการที่มีระยะเวลาพัฒนาสั้น เนื่องจากเวลาที่ใช้ในการเตรียมการอาจไม่คุ้มค่ากับประโยชน์ที่ได้รับ

ข้อดีของการทดสอบอัตโนมัติ คือช่วยเร่งกระบวนการพัฒนาซอฟต์แวร์โดยลดระยะเวลาที่ใช้ในการทดสอบ สามารถดำเนินการซ้ำได้ มีความน่าเชื่อถือสูง สคริปต์ทดสอบสามารถนำกลับมาใช้ใหม่ได้ สามารถตั้งโปรแกรมเพื่อกำหนดเงื่อนไขการทดสอบได้ทำให้มีความยืดหยุ่นและปรับเปลี่ยนได้ง่าย ช่วยเพิ่มประสิทธิภาพในการทำงาน สามารถบันทึกผลการทดสอบได้อย่างละเอียด รองรับการทดสอบสคริปต์บนหลายแพลตฟอร์ม ช่วยลดต้นทุนในการทดสอบซอฟต์แวร์ (Mahajan et al., 2016)

ข้อเสียของการทดสอบอัตโนมัติ คือต้องใช้บุคลากรที่มีความสามารถในการเขียนสคริปต์และเข้าใจภาษาโปรแกรมมิ่งหลากหลายภาษาเพื่อสร้างสคริปต์สำหรับการทดสอบ นอกจากนี้ยังไม่ยืดหยุ่นเท่าการทดสอบด้วยมนุษย์

เนื่องจากไม่สามารถทดสอบอัตโนมัติได้ครบทุกกรณี บางกรณีจำเป็นต้องใช้การทดสอบด้วยมนุษย์เพิ่มเติม เพื่อให้แน่ใจว่าซอฟต์แวร์ทำงานถูกต้องในทุกเงื่อนไข

กรณีทดสอบ (Test Cases) สำหรับระบบบริหารจัดการการให้บริการในแผนกผู้ป่วยนอก

ที่มาของกรณีทดสอบ (Test Cases)

กรณีทดสอบ (Test Cases) สำหรับระบบบริหารจัดการการให้บริการในแผนกผู้ป่วยนอก ควรสร้างขึ้นจากข้อมูลหรือเงื่อนไขที่คาดว่าจะเกิดขึ้นจริงในการปฏิบัติงาน โดยอาจรวมถึงข้อมูลตั้งต้นและเหตุการณ์สมมติที่เกี่ยวข้องกับการทำงานในแผนกผู้ป่วยนอก ทั้งนี้ การออกแบบกรณีทดสอบควรคำนึงถึงการทำงานที่สลับไหลดังแต่ต้นจนจบ รวมถึงสถานการณ์ที่อาจเกิดปัญหาในการทำงาน เพื่อให้ผลการทดสอบมีความสมบูรณ์และมีประสิทธิภาพสูงสุด

การออกแบบกรณีทดสอบควรทำโดยผู้ใช้งานหรือผู้ที่เกี่ยวข้องกับการทำงานในแผนกผู้ป่วยนอก เนื่องจากผู้ใช้งานเข้าใจกระบวนการทำงานและสามารถมองเห็นภาพรวมได้อย่างครอบคลุม นอกจากนี้ การออกแบบกรณีทดสอบอาจเป็นการร่วมมือระหว่างผู้ใช้งานและผู้พัฒนาระบบ เพื่อให้มั่นใจได้ว่าการทดสอบครอบคลุมทุกสถานการณ์ที่อาจเกิดขึ้น

ความละเอียดของข้อมูลในการทดสอบ

ข้อมูลที่ใช้ในการทดสอบควรมีความละเอียดและครอบคลุมทุกขั้นตอนการทำงาน ตั้งแต่การลงทะเบียนผู้ป่วย การจัดการนัดหมาย การเคลื่อนย้ายผู้ป่วย และการออกใบรับรองแพทย์ ข้อมูลที่ใช้ควรเป็นข้อมูลที่ใกล้เคียงกับสถานการณ์จริง เพื่อให้การทดสอบมีประสิทธิภาพและสามารถตรวจสอบความถูกต้องของระบบได้อย่างครบถ้วน

ตัวอย่างกรณีทดสอบ (Test Case) ลงทะเบียนผู้ป่วยที่มีนัดหมาย

เพื่อทดสอบความถูกต้องและประสิทธิภาพของฟังก์ชันการลงทะเบียนผู้ป่วยที่มีนัดหมายในระบบบริหารจัดการการให้บริการในแผนกผู้ป่วยนอก ผู้ป่วยจะต้องมีเลขประจำตัวผู้ป่วย (Hospital Number: HN) ที่ถูกต้องและลงทะเบียนในระบบเรียบร้อยแล้ว นอกจากนี้ระบบต้องสามารถแสดงข้อมูลส่วนตัวของผู้ป่วยได้อย่างถูกต้อง โดยขั้นตอนการทดสอบมีดังนี้

1. เข้าสู่ระบบและเลือกเมนู "ลงทะเบียนนัดหมาย": ผู้ทดสอบเข้าสู่ระบบและเลือกเมนูสำหรับการลงทะเบียนนัดหมาย
2. กรอกรหัสเลข HN ของผู้ป่วย: ผู้ทดสอบกรอกรหัสเลขประจำตัวผู้ป่วยของผู้ป่วยที่ต้องการลงทะเบียน
3. กดปุ่ม "ค้นหา" เพื่อดึงข้อมูลผู้ป่วย: ระบบทำการค้นหาข้อมูลผู้ป่วยตามหมายเลขประจำตัวผู้ป่วยที่กรอก
4. เลือกประเภทการเคลื่อนย้ายผู้ป่วย: ผู้ทดสอบเลือกประเภทการเคลื่อนย้ายผู้ป่วย เช่น เดินได้, รอนั่ง, เปลนอน
5. เลือกความต้องการใบรับรองแพทย์: ผู้ทดสอบระบุว่าผู้ป่วยต้องการใบรับรองแพทย์หรือไม่
6. กดปุ่ม "ลงทะเบียน" เพื่อบันทึกข้อมูล: ระบบทำการบันทึกข้อมูลการลงทะเบียนและแสดงผลลัพธ์

7. ผลลัพธ์ที่คาดหวังจากการทดสอบ (Expected Result)

ในกระบวนการลงทะเบียนนัดหมาย ระบบต้องดำเนินการให้สำเร็จในหลายขั้นตอน โดยเริ่มจากหน้าลงทะเบียนนัดหมายที่ต้องแสดงข้อมูลของผู้ป่วยที่ลงทะเบียนเข้ามาอย่างถูกต้อง นอกจากนี้ ระบบต้องสามารถป้อนข้อมูลเลขประจำตัวผู้ป่วย (Hospital Number: HN) ได้ และเมื่อมีการป้อนข้อมูลดังกล่าว ต้องพบว่าข้อมูลที่แสดงผลตรงกับเลขประจำตัวผู้ป่วยที่มาทำการรักษา อีกทั้งระบบต้องสามารถเลือกประเภทการเคลื่อนย้ายของผู้ป่วยได้ เช่น การเดินได้, การนั่งรถ และการใช้เปลนอน เพื่อให้เหมาะสมกับสภาพของผู้ป่วยในขณะนั้น นอกจากนี้ผู้ป่วยหรือผู้ดูแลต้องสามารถเลือกได้ว่าต้องการรับบริการรับรองแพทย์หรือไม่ ซึ่งเป็นข้อมูลสำคัญสำหรับกระบวนการดูแลรักษา เมื่อผู้ใช้กดปุ่ม "ลงทะเบียน" ระบบต้องสามารถบันทึกข้อมูลการลงทะเบียนอย่างถูกต้องและแจ้งผลลัพธ์ว่าการลงทะเบียนของผู้ป่วยนั้นสำเร็จหรือไม่ เพื่อให้ผู้ป่วยหรือผู้ดูแลมั่นใจว่าข้อมูลทั้งหมดได้รับการจัดเก็บและประมวลผลอย่างถูกต้อง

สรุปผลตัวอย่างกรณีทดสอบสำหรับระบบบริหารจัดการการให้บริการในแผนกผู้ป่วยนอก

การออกแบบกรณีทดสอบสำหรับระบบบริหารจัดการการให้บริการในแผนกผู้ป่วยนอกควรคำนึงถึง ครอบคลุมของการทดสอบและความละเอียดของข้อมูลที่ใช้ รวมถึงการเชื่อมโยงกับระบบอื่น ๆ ภายในโรงพยาบาล การทดสอบนี้ไม่เพียงแต่ช่วยตรวจสอบความถูกต้องของระบบ แต่ยังช่วยเพิ่มความมั่นใจให้กับผู้ใช้งานและผู้ให้บริการในสถานพยาบาล เพื่อให้มั่นใจว่าระบบสามารถทำงานได้อย่างมีประสิทธิภาพและรองรับการใช้งานจริง (ภาพที่ 3)

No.	TEST CASE ID	Case Description	Test Step	Step Description	Expected Result	Test Result	Note/Comment /Question	Last Re-test
1	TC_001	ล็อกอินเข้าสู่ระบบ	Step 1	เปิดเบราว์เซอร์ ไปยัง Website ระบบ Patient Flow Optimization & Management System	- แสดงหน้า Login - มีช่องใส่กรณ Username / Password	Pass		01/02/2568
			Step 2	กรอก Username / Password	สามารถกรอกข้อมูลได้			
			Step 3	คลิกปุ่ม เข้าสู่ระบบ	- หากข้อมูลถูกต้อง ผู้ใช้จะถูกนำไปยังหน้าแดชบอร์ดหรือหน้าหลักของระบบ Patient Flow Optimization & Management System - หากข้อมูลไม่ถูกต้อง ระบบจะแสดงข้อความแสดงข้อผิดพลาดที่ชัดเจนเพื่อให้ผู้ใช้งานตรวจสอบแก้ไขข้อมูล			
2	TC_002	เลือกแผนก	Step 1	เลือกแผนก	- แสดงหน้าหลัก - แสดงชื่อตำแหน่งคอมพิวเตอร์และชื่อแผนกบนแถบ header	Pass		01/02/2568
3	TC_003	ตั้งค่าการใช้งาน	Step 1	เลือกตั้งค่าการใช้งาน	แสดงหน้าตั้งค่าการใช้งาน	Pass		01/02/2568
			Step 2	เลือก ตำแหน่งคอมพิวเตอร์	อัปเดตตำแหน่งคอมพิวเตอร์และชื่อแผนกบนแถบ header			
4	TC_004	ตั้งค่าห้องตรวจ	Step 1	เลือกการตั้งค่าห้องตรวจ	แสดงหน้าตั้งค่าห้องตรวจ	Pass		01/02/2568
			Step 2	เลือกโซน	สามารถเลือกโซนที่ต้องการตั้งค่าได้			
			Step 3	เลือกช่วงเวลา	สามารถเลือกช่วงเวลาที่ต้องการตั้งค่าได้			
			Step 4	คลิก dropdown ที่คอลัมน์ชื่อแพทย์	แสดงรายชื่อแพทย์			
			Step 5	กรอกชื่อแพทย์	กรอกข้อมูลแพทย์ที่มีชื่อตรงกับข้อมูลที่กรอก			
			Step 6	เลือกชื่อแพทย์ที่ต้องการ	สร้าง record ชื่อแพทย์ที่เลือก			
			Step 7	กรอกหมายเลขห้องตรวจ	สามารถกรอกหมายเลขห้องตรวจได้			
			Step 8	คลิกปุ่ม บันทึก	ระบบแสดงข้อความ "บันทึกข้อมูลสำเร็จ"			

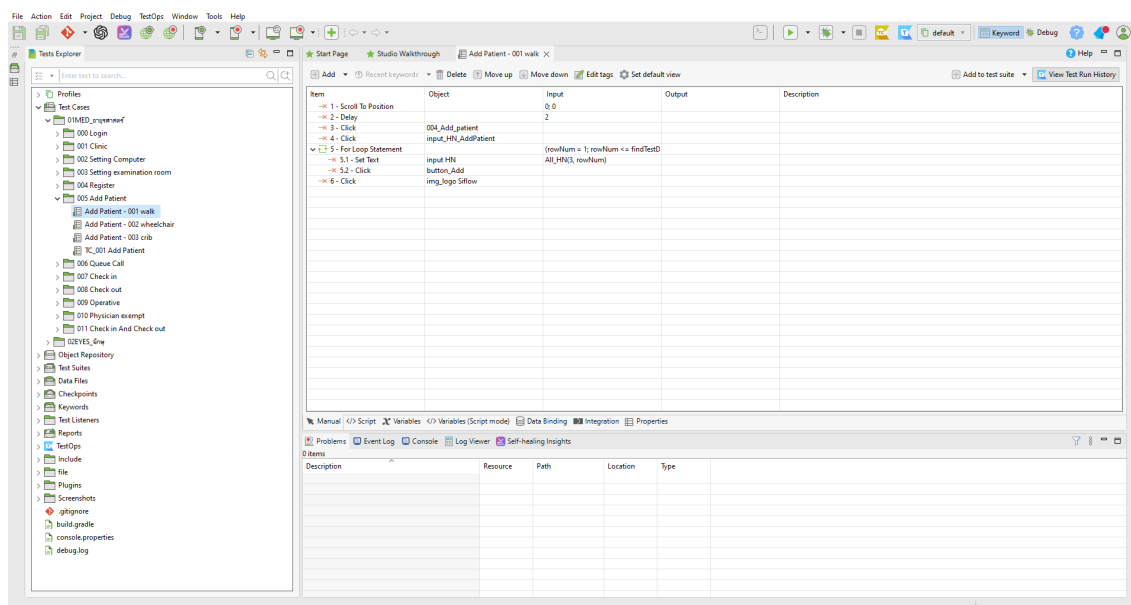
ภาพที่ 3 ตัวอย่าง Test Case ระบบบริหารจัดการการให้บริการในแผนกผู้ป่วยนอก (จากกรณีทดสอบลงทะเบียนผู้ป่วยที่มีนัดหมาย)

การประยุกต์ใช้การทดสอบอัตโนมัติในระบบบริหารจัดการการให้บริการในแผนกผู้ป่วยนอก

ปัจจุบันการทดสอบอัตโนมัติ (Automated Testing) ได้รับความนิยมอย่างแพร่หลายในอุตสาหกรรมซอฟต์แวร์ เครื่องมือที่ใช้สำหรับการทดสอบอัตโนมัติมีหลากหลาย เช่น Selenium, Robot Framework, Cypress และ Katalon Studio โดยแต่ละเครื่องมือมีคุณสมบัติที่แตกต่างกันและเหมาะสมกับลักษณะการทดสอบที่หลากหลาย

ในบทความนี้จะยกตัวอย่างการใช้เครื่องมือ Katalon Studio เป็นเครื่องมือพัฒนาสำหรับการทดสอบอัตโนมัติที่สร้างขึ้นบนพื้นฐานของ Selenium Framework มีหน้าต่างโปรแกรมที่เป็นมิตรเพื่อให้ผู้ใช้สร้าง ดำเนินการ และบำรุงรักษาการทดสอบได้อย่างรวดเร็ว อีกทั้งยังมีฟังก์ชันทั้งหมดพร้อมสำหรับการทำอัตโนมัติบนเว็บไซต์ , API และมือถือ (Android, iOS) พร้อมการทำงานร่วมกันเป็นทีมและการจัดการการทดสอบ โดย Katalon Studio มีสามวิธีหลักในการสร้างการทดสอบ ได้แก่ โหมดบันทึกและเล่นซ้ำ, โหมดแมนวล และโหมดสคริปต์ (Ereiz, 2019)

การนำ Katalon Studio มาใช้ในการทดสอบระบบบริหารจัดการการให้บริการในแผนกผู้ป่วยนอกสามารถช่วยให้มั่นใจได้ว่าระบบทำงานได้อย่างถูกต้องในทุกขั้นตอน ตั้งแต่ การลงทะเบียนนัดหมายผู้ป่วย, การนัดหมายแพทย์, การเรียกผู้ป่วยเข้าห้องตรวจ, การจัดการข้อมูลเวชระเบียน, ไปจนถึงการออกใบเสร็จ และการตรวจสอบสิทธิ์ประกันสุขภาพ โดยการใช้ การทดสอบอัตโนมัติ สามารถช่วย ลดระยะเวลาการทดสอบ, เพิ่มความแม่นยำ และช่วยให้กระบวนการพัฒนาระบบเป็นไปอย่างมีประสิทธิภาพมากขึ้น



ภาพที่ 4 ตัวอย่าง Test Cases ในโปรแกรม Katalon Studio

เมื่อผู้ทดสอบเตรียมสคริปต์ทดสอบ ดังภาพที่ 4 หลังจากนั้นดำเนินการทดสอบโดยใช้โปรแกรมอัตโนมัติ ระบบจะทำตามสคริปต์ทดสอบที่ได้ทำการสร้างไว้ ผู้ทดสอบสามารถเริ่มการทำงานได้โดย กดปุ่มสีเขียว (สัญลักษณ์สามเหลี่ยมชี้ทางขวา) บนแถบเครื่องมือของ Katalon Studio จากนั้นโปรแกรมจะประมวลผลตามคำสั่งที่ละบรรทัด ในตัวอย่างนี้ ระบบจะดำเนินการลงทะเบียนผู้ป่วยที่มีนัดหมาย ผ่านการป้อนข้อมูลเลขประจำตัวผู้ป่วย จากนั้นกดปุ่มลงทะเบียน และทำงานซ้ำโดยอัตโนมัติ ตามข้อมูลที่กำหนดค่าไว้ใน Excel เมื่อการทดสอบเสร็จสิ้น โปรแกรมจะสร้างรายงานผลการทดสอบโดยอัตโนมัติ ดังภาพที่ 5

Execution Environment

Host name: Suppatuch cha - Suppatuch-cha.sihmis.si
Local OS: Windows 10 64-bit
Katalon version: 3.3.1.212
Browser: Chrome 123.0.0.0

Test Execution Log

TEST SUITE: 2_Add Patient Full Name: 2_Add Patient Start / End / Elapsed: 2024-04-03 09:24:08.418 / 2024-04-03 09:24:46.933 / 00:00:38.515 Status: 4 test total, 4 passed, 0 failed, 0 error, 0 incomplete, 0 skipped	Expand All
TEST CASE: Test Cases/01MED_ธพจตวสว000 Login/TC_001 Positive Case	Expand All
TEST CASE: Test Cases/01MED_ธพจตวสว001 Clinic/TC_001 Select clinic	Expand All
TEST CASE: Test Cases/01MED_ธพจตวสว002 Setting Computer/TC_001 setting computer	Expand All
TEST CASE: Test Cases/01MED_ธพจตวสว005 Add Patient/Add Patient - 001 walk Full Name: 2_Add Patient/Test Cases/01MED_ธพจตวสว005 Add Patient/Add Patient - 001 walk Start / End / Elapsed: 2024-04-03 09:24:35.935 / 2024-04-03 09:24:46.933 / 00:00:10.998 Status: PASSED TEST STEP: scrollToPosition(0, 0) TEST STEP: delay(2) TEST STEP: click(findTestObject("Menu/004_Add_patient")) TEST STEP: click(findTestObject("MED/AddPatient/Menu202Input_HV_AddPatient")) TEST STEP: for (rowNum = 1, rowNum <= findTestData("HV").getRowCount(), (rowNum++)) Start / End / Elapsed: 2024-04-03 09:24:39.260 / 2024-04-03 09:24:46.423 / 00:00:07.163 TEST STEP: set(findTestObject("MED/AddPatient/Page_ - FlowInput_HV"), findTestData("HV").getValue(3, rowNum)) TEST STEP: click(findTestObject("MED/AddPatient/Page_ - Flowbutton_Add")) TEST STEP: set(findTestObject("MED/AddPatient/Page_ - FlowInput_HV"), findTestData("HV").getValue(3, rowNum)) TEST STEP: click(findTestObject("MED/AddPatient/Page_ - Flowbutton_HV")) TEST STEP: set(findTestObject("MED/AddPatient/Page_ - FlowInput_HV"), findTestData("HV").getValue(3, rowNum)) TEST STEP: click(findTestObject("MED/AddPatient/Page_ - Flowbutton_HV")) TEST STEP: set(findTestObject("MED/AddPatient/Page_ - FlowInput_HV"), findTestData("HV").getValue(3, rowNum)) TEST STEP: click(findTestObject("MED/AddPatient/Page_ - Flowbutton_HV")) TEST STEP: set(findTestObject("MED/AddPatient/Page_ - FlowInput_HV"), findTestData("HV").getValue(3, rowNum)) TEST STEP: click(findTestObject("MED/AddPatient/Page_ - Flowbutton_HV")) TEST STEP: set(findTestObject("MED/AddPatient/Page_ - FlowInput_HV"), findTestData("HV").getValue(3, rowNum)) TEST STEP: click(findTestObject("MED/AddPatient/Page_ - Flowbutton_HV")) TEST STEP: click(findTestObject("Page_login/Page_login - Flowimg_logo_Sflow")) Start / End / Elapsed: 2024-04-03 09:24:46.423 / 2024-04-03 09:24:46.932 / 00:00:00.509 msg:24-46.932 PASSED Object: Object Repository/Page_login/Page_login - Flowimg_logo_Sflow is clicked on	Expand All

ภาพที่ 5 ตัวอย่าง รายงานผลการทดสอบจากโปรแกรม Katalon Studio (จากการรันสคริปต์ทดสอบภาพที่ 4)

เปรียบเทียบความแตกต่างระหว่างผลการทดสอบด้วยมนุษย์กับการทดสอบแบบอัตโนมัติ

จากผลการทดสอบระบบในกรณีศึกษาการลงทะเบียนผู้ป่วยที่มีนัดหมาย ซึ่งครอบคลุมกระบวนการตั้งแต่การเข้าสู่ระบบ, การเลือกคลินิกที่มีการนัดหมาย และการดำเนินการลงทะเบียนผู้ป่วยจำนวน 5 ราย ทำซ้ำทั้งหมด 10 รอบ สามารถเปรียบเทียบความแตกต่างของการทดสอบทั้งสองรูปแบบได้ (ตารางที่ 1) ดังนี้

ตารางที่ 1 เปรียบเทียบระหว่างการทดสอบด้วยมนุษย์และการทดสอบอัตโนมัติในด้านต่าง ๆ โดยใช้ค่าทางสถิติ

	การทดสอบด้วยมนุษย์ (Manual Testing)	การทดสอบอัตโนมัติ (Automated Testing)
1. ความถูกต้อง (Correctness Testing)	อาจมีข้อผิดพลาดเกิดขึ้นจาก ความไม่รอบคอบหรือความเหนื่อยล้าของผู้ทดสอบ	มีความแม่นยำสูง เนื่องจากใช้ สคริปต์ที่กำหนดไว้อย่างชัดเจน จึงช่วยลดข้อผิดพลาดที่เกิดจากมนุษย์
2. ความน่าเชื่อถือ (Reliability Testing)	มีแนวโน้มให้ผลลัพธ์ที่ไม่สม่ำเสมอ เนื่องจากขึ้นอยู่กับ ปัจจัยภายนอก เช่น อารมณ์หรือประสิทธิภาพของผู้ทดสอบ	สามารถให้ผลลัพธ์ที่ มีความสม่ำเสมอ และทำซ้ำได้ โดยไม่มีความแปรปรวน
3. ความปลอดภัย (Security Testing)	ต้องอาศัยความเชี่ยวชาญเฉพาะทางของผู้ทดสอบ ในการระบุช่องโหว่ด้านความปลอดภัย	สามารถวิเคราะห์ช่องโหว่ผ่านสคริปต์ และสั่งโปรแกรมให้ทดสอบซ้ำในหลายเงื่อนไข แต่อาจไม่สามารถตรวจจับปัญหาที่ซับซ้อนบางกรณี ได้เหมือนการทดสอบด้วยมนุษย์

	การทดสอบด้วยมนุษย์ (Manual Testing)	การทดสอบอัตโนมัติ (Automated Testing)
4. ทดสอบประสิทธิภาพ (Performance Testing)	ใช้เวลามาก และต้องอาศัย เครื่องมือ เพิ่มเติม ในการวัดผล	สามารถจำลองภาระงานที่แตกต่างกัน ได้อย่างแม่นยำ และช่วยให้สามารถ ประเมินขีดความสามารถของระบบได้ อย่างเป็นระบบและสม่ำเสมอ
5. เวลา (Time using)	ใช้เวลาเฉลี่ย 120.6 วินาที (SD = 4.33) โดยมีค่าสูงสุด 129 วินาที และ ค่าต่ำสุด 115 วินาที โดยมีความ แปรปรวนของเวลาที่มากกว่าการ ทดสอบอัตโนมัติ ซึ่งสะท้อนถึงความไม่ สม่ำเสมอของมนุษย์ในกระบวนการ ทดสอบ	ใช้เวลาเฉลี่ย 37.24 วินาที (SD = 0.93) โดยมีค่าสูงสุด 38.82 วินาที และค่าต่ำสุด 35.98 วินาที ซึ่งต่ำกว่า การทดสอบด้วยมนุษย์อย่างมีนัยสำคัญ และมีค่าความแปรปรวนต่ำกว่า แสดง ถึงความสม่ำเสมอของการทำงาน อัตโนมัติ

สรุป

การทดสอบแบบอัตโนมัติ เป็นเทคนิคการทดสอบซอฟต์แวร์ที่ช่วยเพิ่มความเร็ว, ความแม่นยำ และความครอบคลุมของการทดสอบ โดยช่วยลดภาระของผู้ทดสอบจากการดำเนินการทดสอบซ้ำ ๆ และสามารถรองรับการทดสอบจำนวนมากในเวลาอันสั้น อีกทั้งยังช่วยให้สามารถตรวจสอบระบบได้ในหลากหลายแพลตฟอร์มและสภาพแวดล้อมที่แตกต่างกัน ซึ่งเป็นข้อได้เปรียบเหนือการทดสอบแบบดั้งเดิมที่ใช้แรงงานมนุษย์ อย่างไรก็ตาม การเลือกใช้ระหว่างการทดสอบอัตโนมัติและการทดสอบด้วยมนุษย์จะขึ้นอยู่กับลักษณะของซอฟต์แวร์ วัตถุประสงค์ของการทดสอบและทรัพยากรที่มีอยู่เพื่อให้กระบวนการทดสอบมีประสิทธิภาพสูงสุด (Thant & Tin, 2023)

ในการศึกษานี้พบว่า ระบบบริหารจัดการการให้บริการในแผนกผู้ป่วยนอกสามารถยกระดับกระบวนการทดสอบซอฟต์แวร์ได้อย่างมีประสิทธิภาพด้วยเครื่องมือทดสอบอัตโนมัติ โดยเฉพาะในส่วนที่ต้องดำเนินการซ้ำ ๆ และต้องการความถูกต้องแม่นยำ เช่น การตรวจสอบข้อมูลผู้ป่วย การบันทึกการนัดหมายและการจัดการข้อมูลในระบบคิว ซึ่งการใช้เครื่องมืออัตโนมัติช่วยลดภาระของผู้ทดสอบ ลดโอกาสเกิดข้อผิดพลาดจากมนุษย์และเพิ่มความสม่ำเสมอของผลลัพธ์ อย่างไรก็ตาม การทดสอบด้วยมนุษย์ยังมีความจำเป็น โดยเฉพาะในกรณีที่เกี่ยวข้องกับประสบการณ์ผู้ใช้และการใช้งานจริงของเจ้าหน้าที่โรงพยาบาล เช่น ระบบคิวที่ต้องรองรับผู้ป่วยจำนวนมากหรือการใช้งานซอฟต์แวร์ของแพทย์และเจ้าหน้าที่ ซึ่งอาจมีปัจจัยแวดล้อมที่ส่งผลต่อการใช้งานและไม่สามารถจำลองได้ด้วยการทดสอบอัตโนมัติ

ดังนั้น การผสมผสานระหว่างการทดสอบอัตโนมัติและการทดสอบด้วยมนุษย์ จะช่วยให้การทดสอบซอฟต์แวร์มีความครอบคลุมมากขึ้น ลดข้อผิดพลาด เพิ่มประสิทธิภาพและช่วยให้ระบบบริหารจัดการในแผนกผู้ป่วยนอกทำงานได้อย่างมีประสิทธิภาพ ซึ่งส่งผลโดยตรงต่อคุณภาพของการให้บริการและประสบการณ์ของผู้ป่วยนอก

ข้อเสนอแนะ

การทดสอบซอฟต์แวร์แบบอัตโนมัติเป็นแนวทางที่ช่วยเพิ่มประสิทธิภาพในการตรวจสอบคุณภาพซอฟต์แวร์ โดยสามารถลดระยะเวลาการทดสอบและข้อผิดพลาดที่เกิดจากมนุษย์ในการทดสอบซ้ำ ๆ อีกทั้งยังรองรับการทดสอบในสภาพแวดล้อมที่หลากหลายได้อย่างรวดเร็วและแม่นยำ อย่างไรก็ตาม การทดสอบด้วยมนุษย์ยังคงมีบทบาทสำคัญในบางกรณีที่ต้องการการวิเคราะห์เชิงลึก ความยืดหยุ่น หรือการตรวจสอบในแง่มุมที่เครื่องมืออัตโนมัติอาจไม่สามารถประเมินได้อย่างครบถ้วน

ดังนั้น การเลือกใช้การทดสอบแบบอัตโนมัติหรือการทดสอบด้วยมนุษย์ควรพิจารณาจากลักษณะของโครงการ ข้อกำหนดด้านคุณภาพและทรัพยากรที่มีอยู่ การผสมผสานการทดสอบทั้งสองรูปแบบอย่างเหมาะสมจะช่วยให้กระบวนการทดสอบมีประสิทธิภาพและครอบคลุมมากขึ้น ส่งผลให้ซอฟต์แวร์มีความน่าเชื่อถือและพร้อมใช้งานในสภาพแวดล้อมที่แท้จริงได้อย่างมั่นใจ

กิตติกรรมประกาศ

บทความทางวิชาการฉบับนี้สำเร็จลุล่วงไปได้ด้วยดี โดยได้จากการสนับสนุนจากฝ่ายสารสนเทศ คณะแพทยศาสตร์ศิริราชพยาบาล มหาวิทยาลัยมหิดลซึ่งได้เล็งเห็นความสำคัญของการพัฒนาความรู้และทักษะของพนักงานในองค์กร โดยให้โอกาสในการจัดทำบทความนี้ขึ้นมาเพื่อเสริมสร้างความรู้และเพิ่มประสิทธิภาพในการปฏิบัติหน้าที่ นอกจากนี้ฝ่ายสารสนเทศยังได้ให้คำแนะนำที่มีคุณค่าในการทดสอบซอฟต์แวร์โดยเน้นถึงความจำเป็นในการใช้งานจริง ซึ่งมีส่วนช่วยให้ผู้เขียนสามารถดำเนินการเขียนบทความนี้ได้อย่างมีประสิทธิภาพ ผู้เขียนขอกราบขอบพระคุณเป็นอย่างสูงไว้ ณ โอกาสนี้

เอกสารอ้างอิง

- Ereiz, Z. (2019). *Automating web application testing using Katalon Studio*.
<https://www.researchgate.net/publication/343162368>
- lamkhajornchai, C. (2023). เราต้องตรวจสอบ และ ทดสอบไรงี้บ้างใน Software หนึ่งตัว. BorntoDev.
<https://www.borntodev.com/2023/09/19/เราต้องตรวจสอบ-software-และทดสอบ/>
- Jinda, N., & Amornmontien, W. (2023). The importance of software testing Medical before actual use. *Mahidol R2R e-Journal*, 10(2), 1-12.
- Jobsdb. (2023). *Test case คืออะไรทำไม Tester จำเป็นต้องทำ?* <https://th.jobsdb.com/th/career-advice/article/test-case>
- Kumar, D., & Mishra, K. K. (2016). The impacts of test automation on software's cost, quality and time to market. *Procedia Computer Science*, 79, 8-15.
- Mahajan, P., Shedge, H., & Patkar, U. (2016). *Automation testing in software organization*. *International Journal of Computer Applications Technology and Research*, 5(4), 198-201.

Senna Labs. (2023). *Software testing มีกี่ประเภท และวิธีการเลือกใช้ให้เหมาะสม*.

<https://sennalabs.com/blog/type-of-software-testing>

Skooldio. (2024). *ความแตกต่างของ Software Testing และ Automated Testing*.

<https://blog.skooldio.com/software-testing-vs-automated-testing/>

Thant, K. S., & Tin, H. H. K. (2023). The impact of manual and automatic testing on software testing efficiency and effectiveness. *Indian Journal of Science and Research*, 14(2), 45-56.