**บทที่ 3**

**ขั้นตอนการพัฒนาและออกแบบระบบ**

ในบทนี้จะกล่าวถึงขั้นตอนในการพัฒนาซอฟต์แวร์ทดสอบอัตโนมัติ โดยในการพัฒนานั้นได้ดำเนินการตามขั้นตอนที่ได้วางแผนไว้ดังต่อไปนี้

1. การศึกษาระบบ MicroData

2. การทำความเข้าใจจุดประสงค์และขอบเขตในการพัฒนา

3. การออกแบบซอฟต์แวร์

4. การพัฒนาซอฟต์แวร์

**3.1 การศึกษาระบบ MicroData**

เนื่องจากซอฟต์แวร์ทดสอบอัตโนมัติที่ทำการสร้างขึ้นนั้นมีจุดประสงค์เพื่อที่จะทดสอบความถูกต้องในการทำงานของระบบ MicroData โดยเฉพาะ ดังนั้นจึงมีความจำเป็นที่ผู้เขียนจะต้องทำความเข้าใจถึงการทำงานของระบบ MicroData ทั้งในมุมมองด้านธุรกิจ และมุมมองทางด้านเทคนิคในเชิงลึก เพื่อที่จะสามารถสร้างซอฟต์แวร์ทดสอบระบบที่ครอบคลุมกรณีปัญหาต่างๆที่เกิดขึ้นได้จริงในการทำงานอย่างครบถ้วน

โดยระบบ MicroData นั้นเป็นระบบที่ถูกสร้างขึ้นเพื่อช่วยในการดำเนินการแปลงเอกสาร แบบฟอร์มต่างๆที่อยู่ในรูปแบบของกระดาษ (Paper-based) ให้เปลี่ยนแปลงไปอยู่ในรูปของข้อมูลอิเล็กทรอนิก เพื่อที่สามารถนำไป จัดเก็บ ประมวลผล หรือดำเนินการอื่นๆ ด้วยระบบคอมพิวเตอร์ต่อไปได้ ซึ่งขั้นตอนการทำงานในการแปลงนี้ ถ้าให้มนุษย์เป็นผู้ดำเนินการ อาจทำให้เกิดข้อผิดพลาดในการทำได้ ถ้าแบบฟอร์มนั้นมีความซับซ้อนสูง หรือมีจำนวนมาก อีกทั้งยังเป็นการสิ้นเปลืองทรัพยากรมนุษย์ในการทำขั้นตอนต่างๆเหล่านี้ เนื่องจากแบบฟอร์มที่มีความซับซ้อน เช่นแบบฟอร์มทางด้านการเงิน จำเป็นต้องใช้คนทำงานใช้ความรู้ (Knowledge worker) ดังนั้นการใช้ระบบ MicroData มาทำงานในส่วนนี้แทน จึงมีประโยชน์อย่างมากต่อองค์กรที่ต้องทำงานกับเอกสารจำนวนมาก ช่วยลดต้นทุน ลดจำนวนทรัพยากรที่ต้องเสียไปในการทำการแปลงเอกสารเหล่านี้

ซึ่งวิธีการทำงานของระบบ MicroData จะเป็นไปดังรูปที่ 3.1

โดยขั้นตอนการทำงานนั้นเริ่มจากการที่ลูกค้าที่ใช้ระบบทำการแสกนเอกสารหรือแบบฟอร์มของตนเองที่ต้องการนำมาแปลงข้อมูล เมื่อแสกนได้ไฟล์มาเป็นรูปภาพซึ่ง นามสกุลไฟล์นั้นขึ้นอยู่กับว่าได้ทำการตกลงไฟล์ฟอร์แมทใดบ้างเอาไว้ ทำการอัพโหลดไฟล์ภาพขึ้นมาบนระบบของ MicroData ผ่านทางโพรโตคอล SFTP เนื่องจาก ระบบของ MicroData นั้นเป็นระบบที่มีการรักษาความปลอดภัยเพื่อรักษาข้อมูลของลูกค้า ไฟล์ที่ถูกอัพโหลดขึ้นมานั้นจะถูกนำเข้าสู่ Data Entry process

ซึ่งมีหน้าที่ในการตรวจสอบฟอร์แมทของไฟล์ภาพ และทำงานต่างๆในการบันทึกไฟล์นี้ลงในระบบของ MicroData หลังจากข้อมูลถูกบันทึกแล้วก็จะมี third party server ตัวนึงมาทำการดึงไฟล์ภาพไปทำการแย่งภาพออกเป็นส่วนย่อยๆหรือเรียกว่าการทำ data dissection เมื่อเสร็จแล้วก็จะทำการส่งภาพเหล่านั้นไปทำการแปลเอกสารเป็นข้อมูลอิเล็กทรอนิก หรือเรียกว่าการทำ data transcription

จากนั้นข้อมูลที่ได้จะถูกนำกลับมารวมกันเป็นข้อมูลของไฟล์ภาพเต็มนั้น ซึ่งประโยชน์ของการกระจาย

ไฟล์ภาพเป็นส่วนๆ ก็เพื่อช่วยให้การแปลงภาพเป็นข้อมูลนั้นทำได้ง่าย โดยใช้หลักการ “key what you see” ทำให้ไม่จำเป็นต้องใช้ knowledge worker มาทำงานในส่วนนี้ และข้อดีอีกข้อนึงก็คือ ในขั้นตอนการแปลงข้อมูลนั้น จะไม่มีใครได้เห็นเอกสารทั้งฉบับ แต่จะเห็นเพียงแค่ส่วนหนึ่ง ดังนั้นจึงเป็นการรักษาความปลอดภัยให้กับลูกค้าด้วยเช่นกัน หลังจากเอกสารของเราถูกแปลงเป็นข้อมูลอิเล็ก-ทรอนิกเรียบร้อย third party ก็จะทำการเรียกใช้ web Service ที่ระบบ MicroData เตรียมเอาไว้ โดยเป็น Web Service ประเภท RESTFul (Representational state transfer) เพื่อทำการส่งข้อมูลหลังการทำ data transcription กลับมายังระบบ MicroData และเข้าสู่โพรเซส Data Validation เพื่อทำการตรวจสอบความถูกต้องของข้อมูลที่ผ่านการแปลง ถ้าข้อมูลมีความผิดพลาด ก็จะมี third party อีกตัวหนึ่ง ซึ่งมีหน้าที่ในการซ่อมข้อมูลที่มีข้อผิดพลาดที่ส่งมาให้ถูกต้อง โดยที่ระบบ third party ตัวนี้มีส่วนติดต่อกับโปรแกรมเป็น web Service ประเภท SOAP (Simple Object Access protocol) ซึ่งเมื่อ third party ตัวนี้ทำการซ่อมข้อมูลเสร็จเรียบร้อยแล้วก็จะทำการส่งข้อมูลที่ถูกซ่อมกลับมาให้ผ่าน SOAP web Service ที่ระบบ MicroData เตรียมไว้เพื่อทำการติดต่อกับ third party ตัวนี้เพื่อนำข้อมูลไปเข้าสู่กระบวนการ data validation อีกรอบ เมื่อข้อมูลผ่านการ validation อย่างถูกต้องแล้ว ข้อมูลก็จะถูกสร้างออกมาเป็น ไฟล์อิเล็กทรอนิกตามรูปแบบที่ลูกค้าต้องการเพื่อนำไปใช้ประโยชน์ต่อไป

**3.2 การทำความเข้าใจจุดประสงค์และขอบเขตในการพัฒนา**

จากหัวข้อที่ 3.1 การศึกษาโปรเซสการทำงานของระบบ MicroData ตั้งแต่ลูกค้า ทำการ อัพโหลดไฟล์ขึ้นมาจนถึงขั้นตอนที่ระบบ MicroData ทำการสร้างไฟล์ผลลัพธ์ออกมานั้น ระหว่างทางนั้นมีโปรเซสต่างๆ ทั้งการตรวจสอบความถูกต้อง การบันทึกข้อมูล การสร้างไฟล์ ย้ายไฟล์ การติดต่อกับ third party ด้วย web service และโปรเซสอื่นๆอีกมากมาย ซึ่งจะเห็นได้ว่าระบบ MicroData นั้นมีกระบวนการทำงานที่ค่อนข้างจะซับซ้อน ซึ่งในปัจจุบันการทดสอบความถูกต้องของระบบนั้นใช้การทดสอบแบบ manual testing ซึ่งใช้เวลามาก ในการตรวจสอบแต่ละครั้ง เนื่องจากการทดสอบจะต้องทำการ manual ในจุดต่างๆ เช่น การรับส่งข้อมูลจาก third party การสร้างไฟล์ การอัพโหลดไฟล์ และในการตรวจสอบนั้นก็จะต้องไปไล่ดูข้อมูลในจุดต่างๆเอง ซึ่งอาจเกิดการตรวจสอบได้ไม่ทั่วถึง และอีกข้อสำคัญก็คือ หากมีการเปลี่ยนแปลงแก้ไขซอร์สโค้ดในทุกครั้ง จะต้องมีการตรวจสอบ ความถูกต้องของระบบ ซึ่งถ้ามีการเปลี่ยนแปลงซอร์สโค้ดบ่อยๆ หรือมีกรอบเวลาในการพัฒนาที่สั้น การทำ manual testing นั้นอาจจะทำได้ไม่ทัน และสิ้นเปลืองทรัพยากรมากในการทำ

ดังนั้น โครงการพัฒนาซอฟต์แวร์ทดสอบอัตโนมัติสำหรับระบบ MicroData ด้วยการพัฒนาแบบขับเคลื่อนจากพฤติกรรมผู้ใช้ จึงถูกสร้างขึ้นมาเพื่อช่วยทดสอบความถูกต้องในการทำงานของระบบ MicroData ด้วยการทดสอบแบบอัตโนมัติ (automated test) ซึ่งจะช่วยให้การทดสอบระบบ

สามารถทำได้ง่าย รวดเร็ว เพื่อตอบสนองต่อกรอบเวลาในการพัฒนาที่สั้น และมีความต้องการในการทำการทดสอบถี่ โดยซอฟต์แวร์นี้ จะช่วยให้สามารถตรวจสอบการทำงานของระบบ ได้ครบถ้วน ครอบคลุม กรณีที่เกิดขึ้นเป็นประจำ ในทุกๆครั้งที่ทำการทดสอบระบบ เนื่องจากมีการเขียน กรณี ทดสอบที่จำเป็นต้องทดสอบทุกครั้งเอาไว้ ดังนั้นการทดสอบแบบอัตโนมัตินั้นจะทำการตรวจสอบทุกกรณีที่เขียนไว้ได้อย่างครบถ้วน ในทุกๆครั้งที่ทำการรัน โดยการทดสอบอัตโนมัติจะทำการตรวจสอบความถูกต้องในการทำงานของระบบตั้งแต่เริ่มการทำงานในส่วนของการรับไฟล์ที่ลูกค้าทำการอัพโหลด ตรวจสอบในทุกส่วนของระบบ ไปจนกระทั่งไฟล์ผลลัพธ์ถูกสร้างออกมา ซึ่งขั้นตอนต่างๆ เช่น การอัพโหลดไฟล์ขึ้นไปบนระบบ การส่งข้อมูลที่ถูกแปลงแล้ว หรือการซ่อมเอกสารต่างๆจะเป็นไปแบบอัตโนมัติทั้งหมด ทำให้ ไม่จำเป็นที่จะต้องทำการอัพโหลดไฟล์ หรือทำการเรียกใช้ web Service แทน third party ด้วยมือทั้งหมด ซึ่งในส่วนของวิธีการทดสอบนั้น จะแบ่งการตรวจสอบหลักๆ ออกเป็น 3 ส่วนคือ 1.การตรวจสอบจากบันทึกการทำงานของระบบ (Processing log) 2.ข้อมูลภายในฐานข้อมูลที่ถูกบันทึกระหว่างการทำงาน และ 3.การตรวจสอบไฟล์ผลลัพธ์ที่ถูกสร้างออกมาให้กับลูกค้า โดยการพัฒนาซอฟต์แวร์ทดสอบอัตโนมัติตัวนี้ จะใช้รูปแบบการพัฒนาแบบขับเคลื่อนจากพฤติกรรมผู้ใช้ (Behavior driven development) ซึ่งเป็นวิธีที่ช่วยให้การพัฒนาการทดสอบแบบอัตโนมัติทำได้สะดวกและง่ายดายขึ้น และผู้เขียนยังได้พัฒนาในส่วนของเว็บแอพพลิเคชั่นซึ่งเป็นเครื่องมือที่สร้างขึ้นมาเพื่อช่วยให้การตรวจสอบ และค้นหาข้อมูลในฐานข้อมูลทำได้สะดวกมากขึ้น เนื่องจากในปัจจุบันการจะตรวสอบข้อมูลในฐานข้อมูลนั้นจะต้องทำการพิมพ์คำสั่งภาษา SQL ผ่านโปรแกรมจัดการฐานข้อมูลตรงๆ เพื่อเรียกดูข้อมูล ดังนั้นการมีเว็บแอพพลิเคชั่นตัวนึงซึ่งช่วยให้การเข้าถึงข้อมูลในฐานข้อมูลด้วยส่วนติดต่อผู้ใช้ จะช่วยให้การตรวจสอบข้อมูลทำได้รวดเร็วขึ้น และลดการพิมพ์คำสั่ง SQL ที่ซับซ้อน ทำให้ผู้ใช้ใช้งานได้สะดวกมากยิ่งขึ้น

**3.3 การออกแบบซอฟต์แวร์**

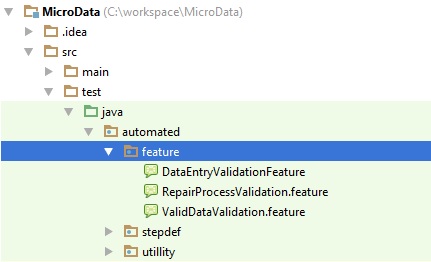
ในการออกแบบการพัฒนาของซอฟต์แวร์ทดสอบอัตโนมัตินั้นเราจะพูดถึงการออกแบบในส่วนของ test Script ซึ่งเป็นตัวที่จะถูกใช้ในการรันการทดสอบแบบอัตโนมัติ โดย test Script ของเรานั้นจะถูกเขียนขึ้นด้วย framework ที่ชื่อว่า Cucumber โดยเราจะเรียก script ที่ถูกเขียน ขึ้นด้วย Cucumber framework ว่า “Feature file” ซึ่ง feature file นั้นจะรับผิดชอบในการทดสอบอัตโนมัติสำหรับแต่ละฟังก์ชั่นในระบบที่ต้องการทดสอบ ดังนั้นในส่วนการออกแบบของเราจะมุ่งเน้นไปที่การออกแบบตัว feature file ว่าแต่ละ feature file ของเรานั้นถูกสร้างขึ้นเพื่อทดสอบ อะไร และทดสอบอย่างไร ซึ่งหลักๆแล้วในการทดสอบระบบของเราจะมีวิธีการพื้นฐานเดียวกันก็คือ เราจะทำการจำลองการทำงานเช่น การสร้างไฟล์ การอัพโหลดไฟล์สู่ระบบ และการจำลองการ รับ-ส่ง ข้อมูลกับ third party เพื่อให้เกิดการทำงานของโปรเซสต่างๆภายในระบบ ทำให้มีการสร้างprocessing Log และข้อมูลจริงในระบบเกิดขึ้น หลังจากนั้นจึงทำการตรวจสอบค่าที่เกิดขึ้น ว่าระบบทำงานได้ถูกต้องหรือไม่ ซึ่งทุกโปรเซสที่กล่าวมานั้นจะเป็นการทำงานแบบอัตโนมัติทั้งหมด โดยทุก feature file หรือ test script ที่เขียนขึ้นนั้นก็จะทำการดำเนินงานตามขั้นตอนข้างตั้นทั้งหมด ดังนั้นสิ่งที่จะเป็นข้อแตกต่างในแต่ละ feature file ที่เราจะทำการเขียนขึ้นนั้นก็คือ แต่ละ feature file จะทำหน้าที่เป็นตัวทดสอบสำหรับแต่ละกรณีที่เกิดขึ้นได้ในการทำงานจริง เช่น จะมี feature ที่ใช้ทดสอบว่าถ้าเราส่งอินพุตแบบต่างๆเข้ามาในระบบจะเกิดอะไรขึ้น หรือ feature ที่ใช้ในการตรวจ สอบการทำงานของการซ่อมข้อมูลที่เสียหาย เป็นต้น

ดังนั้นผู้เขียนจึงได้ออกแบบโดยการแบ่ง Feature file ออกเป็น 3 ไฟล์หลักๆ คือ

1. DataEntryValidation Feature

2. ValidDataValidation Feature

3. RepairProcessValidation Feature

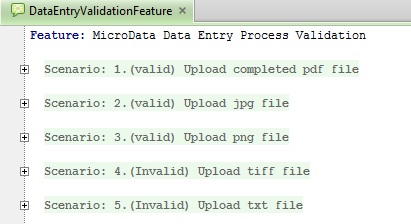


**รูปที่ 3.xx** MicroData feature file

โดยรายละเอียดขั้นตอนการทำงานของแต่ละ Feature จะอธิบายดังต่อไปนี้

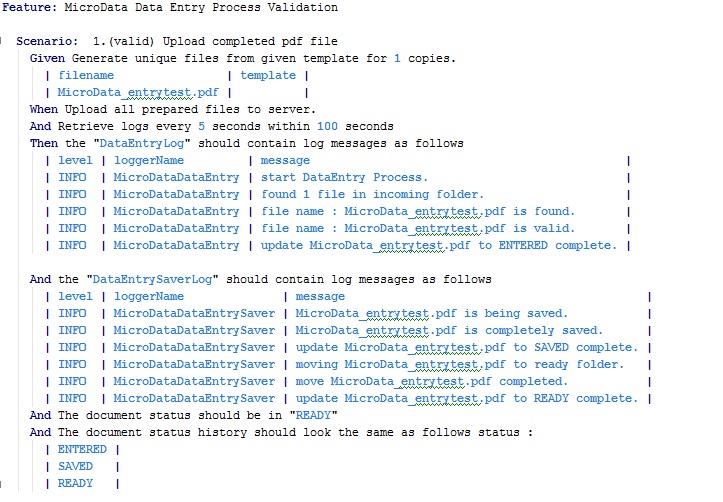
1. DataEntryValidation Feature เป็น feature file ที่ใช้ในการตรวจสอบความถูกต้องใน

การทำงานของ Data Entry Process ในระบบ MicroData ซึ่งทำงานเกี่ยวกับการรับไฟล์ภาพที่ถูกอัพโหลดขึ้นมาบนเซิฟเวอร์ และบันทึกข้อมูลเบื้องต้น ดังนั้นจุดประสงค์ของ DataEntryValidation Feature คือเป็น Feature ที่รวบรวมสถานการณ์ทดสอบ (Scenario) ที่เกี่ยวกับการสร้างและอัพโหลดไฟล์ประเภทต่างๆขึ้นมาบนระบบเช่น อัพโหลด file นามสกุล pdf, jpg ,png เป็นต้น เพื่อตรวจสอบถึงผลลัพธ์ว่า ถ้าอัพโหลดไฟล์ที่ระบบซัพพอร์ท ไฟล์ Processing Log จะต้องบันทึกข้อความที่ควรจะเกิดขึ้นถ้าระบบทำงานถูกต้อง และบันทึกข้อมูลลงฐานข้อมูลในเบื้องต้นอย่างถูกต้อง และถ้าไฟล์ประเภทนั้นระบบไม่ซัพพอร์ท ไฟล์บันทึกการทำงานของระบบจะต้องมีการบันทึกข้อความที่แสดงถึงข้อความที่ปฎิเสธไฟล์เหล่านั้น รวมถึงข้อมูลในฐานข้อมูล จะต้องไม่มีไฟล์เหล่านั้นด้วยเช่นกัน โดย scenario ต่างๆใน Data Entry Validation Feature จะมีดังรูปที่ 3.xx



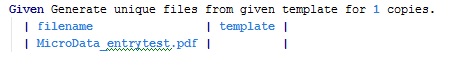
**รูปที่ 3.xx** DataEntryValidation Feature

จากรูป 3.xx จะเห็นได้ว่าแต่ละ Scenario ใน Feature นั้นจะทำการอัพโหลดไฟล์ที่มีนามสกุลต่างๆกันขึ้นไปบนระบบเพื่อทำการตรวจสอบถึงผลที่ออกมา สำหรับรายละเอียดการทำงานของแต่ละ Scenario จะอธิบายจากรูปที่ 3.xx



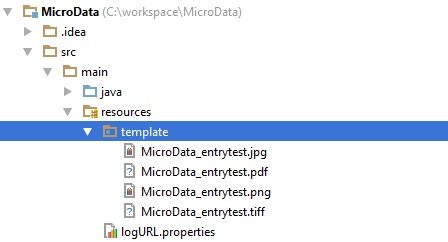
**รูปที่ 3.xx** รายละเอียดของ Data Entry Scenario

โดย Scenario ที่นำมายกตัวอย่างนี้ คือกรณีที่ทำการสร้างไฟล์นามสกุล pdf แล้วทำการ อัพโหลด ไฟล์นี้ขึ้นสู่ระบบ จากที่เห็นในรูป บรรทัดภายใน Scenario ที่มี keyword นำหน้าเป็นสีส้ม ที่เขียนว่า Given, When, Then นั้นจะถูกเรียกว่า Step ซึ่งแต่ละ Step นั้นจะมีการเชื่อมโยงกับการทำงานด้านหลัง ซึ่งเขียนด้วยภาษา Java เพื่อทำงานตามประโยคที่เขียน แต่ภายในหัวข้อนี้ จะอธิบาย เพียงแค่หน้าที่ของแต่ละ Step ส่วนการ Implementation หรือส่วนของ Java จะทำการอธิบาย ในหัวข้อต่อไป เริ่มต้นที่ Step แรก ดังรูปที่ 3.xx



**รูปที่ 3.xx** Step การสร้างไฟล์จาก template

โดย Step Given หรืออินพุตของระบบ นี้มีหน้าที่ในการสร้างไฟล์ที่มีชื่อและนามสกุลของไฟล์ตามที่ระบุเอาไว้ในช่อง file name ซึ่งในที่นี้คือ MicroData\_entrytest ซึ่งมีนามสกุลเป็น pdf ส่วนในช่องที่ 2 ด้านข้างนั้นคือช่อง template ซึ่งเอาไว้ใส่ path ของไฟล์ที่เราต้องการนำมาใช้เป็น template โดยโปรแกรมจะทำการคัดลอก template ไฟล์มาใส่ให้ไฟล์ที่ถูกสร้างขึ้น ไฟล์ที่ถูกสร้างขึ้นจะมีชื่อไฟล์ตามที่ระบุในช่อง filename และมีเนื้อหาของไฟล์เป็นเนื้อหาของไฟล์ template ส่วนในกรณีที่ไม่ได้ระบุ template เอาไว้หรือปล่อยช่อง template ว่างไว้ ซอฟต์แวร์จะถือว่าเป็นการใช้ default template และจะไปหาไฟล์ที่มีชื่อเดียวกันใน folder template และคัดลอกไฟล์มาให้กับไฟล์ของเรา โดย default template จะอยู่ในโปรเจค ดังรูป 3.xx



**รูปที่ 3.xx** template โฟลเดอร์

โดยตัวเลขในประโยค Given ซึ่งในที่นี้คือ 1 copies หมายถึง ต้องการสร้างไฟล์จำนวนกี่ไฟล์ซึ่งเราสามารถใช้ประโยชน์จากการสร้างไฟล์หลายๆไฟล์ได้ ในกรณีที่เราต้องการจำลองการส่งไฟล์ขึ้นสู่ระบบหลายๆไฟล์ในเวลาเดียวกัน เพื่อทำ Load test หลังจากเราได้ทำการสร้างไฟล์ขึ้นมา จาก template file ด้วย step Given ในขั้นตอนก่อนหน้าสำเร็จ ขั้นตอนต่อไป คือขั้นตอนในการอัพโหลดไฟล์ทั้งหมดที่สร้างขึ้นไปบนเซิฟเวอร์ ด้วย step When ดังรูปที่ 3.xx

deWhen.jpg

**รูปที่ 3.xx** Step การอัพโหลดไฟล์สู่เซิฟเวอร์

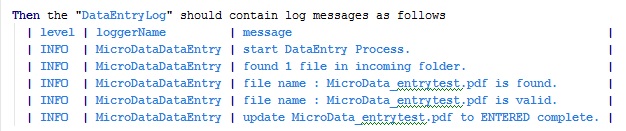
การทำงานของ Step นี้ จะเริ่มจากการทำการเชื่อมต่อกับ Secure shell server โดยเขียนโปรแกรมด้วย Library Jsch ของภาษา Java เชื่อมต่อกับ server ผ่าน SFTP protocol แล้วทำการนำเอาไฟล์ทั้งหมดที่ถูกสร้างไปวางไว้ที่ folder ของระบบ MicroData โดย ระบบ MicroData จะมี กระบวนการตรวจสอบที่จะนำไฟล์เหล่านี้เข้าสู่โปรเซสต่อๆไปโดยอัตโนมัติ ดังนั้นหน้าที่ของ Step นี้ก็คือการนำไฟล์ที่ถูกสร้างไปวางไว้ใน incoming folder ของระบบเพื่อทำให้โปรเซสต่างๆ เริ่ม ทำงาน ในส่วนของการ Implementation จะอธิบายในหัวข้อต่อไป

เมื่อทำการอัพโหลดไฟล์เข้าสู่ระบบเสร็จเรียบร้อย การทำงานของโปรเซสต่างๆในระบบ MicroData จะเริ่มต้นทำงานโดยอัตโนมัติ ขั้นตอนต่อไปในการ Feature ของเราก็คือการตรวจสอบ. โดยการตรวจสอบขั้นแรกของเราคือ การตรวจสอบ ไฟล์บันทึกการทำงานระบบ (Processing log) แต่เนื่องจากการทำงานในแต่ละโปรเซสอาจจะมีการใช้เวลาเล็กน้อยก่อนที่จะทำการบันทึกข้อความการดำเนินงานลงในไฟล์ ดังนั้นหากเราทำการดึงไฟล์ขึ้นมาตรวจสอบข้อความเลยทันที อาจพบว่า การทดสอบนั้นแสดงผลผิดพลาด ไม่พบกับข้อความที่เราต้องการ เนื่องจาก Log ยังไม่ถูก save ลงในไฟล์ ดังนั้นเราจึงมี Step ในการตั้งค่า เพื่อให้การทดสอบนั้นยืดหยุ่น โดยอนุญาติให้มีการดึงไฟล์ขึ้นมาใหม่เพื่อทดสอบซ้ำได้ในเวลาที่กำหนด ตาม step ในรูปที่ 3.xx

deAnd.jpg

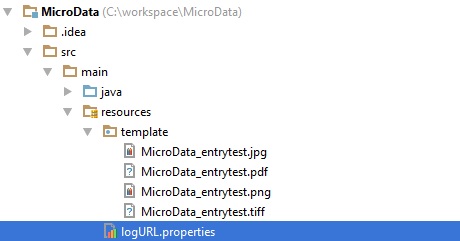
**รูปที่ 3.xx** Step ในการหน่วงเวลาการทดสอบ

โดยรูปที่ 3.xx นั้นมีวิธีการทำงานคือ เป็น Step ที่ทำการตั้งค่าให้ซอฟต์แวร์ว่าจะ ทำการอ่านไฟล์ จากระบบมาทำการตรวจสอบภายในทุกกี่วินาที และถ้าครบเวลาภายในกี่วินาทีให้ทำการยกเลิกการตรวจสอบ และแสดงข้อความว่าการตรวจสอบผิดพลาด เนื่องจากเกินเวลาที่กำหนด เนื่องจากระบบ MicroData ยังไม่ทำการสร้าง Processing Log หรือสร้าง Log ที่ไม่ถูกต้องลงในไฟล์ แต่ถ้ากระบวนการตรวจสอบข้อความที่ถูกบันทึกตรงตามที่คาดไว้สำเร็จ ระบบจะแสดงข้อความว่าตรวจสอบถูกต้อง และจะทำการข้ามไปทำการตรวจสอบขั้นตอนต่อไปทันทีโดยไม่ต้องรอให้ ครบเวลาที่กำหนด ซึ่ง Step ในการตรวจสอบไฟล์บันทึกการทำงานของระบบนั้น จะเป็นดังรูปที่ 3.xx



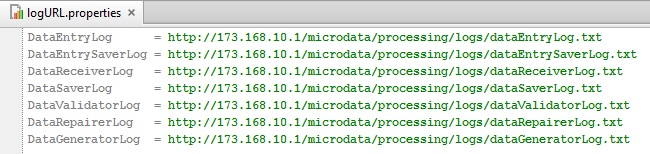
**รูปที่ 3.xx** Step ในการตรวจสอบ DataEntry Log

โดย Step นี้จะทำหน้าที่ในการตรวจสอบความถูกต้องในการทำงานของระบบ ด้วยการเปรียบเทียบระหว่าง processing log ที่ถูกสร้างขึ้นจริงระหว่างการทำงานในแต่ละโปรเซสของระบบ MicroData กับเซตของ log ที่เตรียมเอาไว้ภายในตารางที่อยู่ใน feature file โดยเซตของ Log นี้จะเป็นข้อความที่ผู้ใช้ใส่เข้าไป ซึ่งข้อความนี้จะเป็นข้อความที่ควรจะเกิดขึ้น ถ้าหากระบบทำงานได้ถูกต้องตามปกติ โดยได้มาจากการลองทำ manual testing แล้วนำเอาข้อความที่เกิดขึ้นมา หรือทำการศึกษาเอกสาร process workflow ของระบบ การทำงานของ step นี้ก็คือจะนำเอาชื่อของ log ที่เขียนไว้ใน feature ซึ่งในทีนี้คือ “DataEntryLog” ซึ่งคือ log ของโปรเซส DataEntry ไปทำการหา url ที่เก็บไว้ในไฟล์ logURL.properties ดังภาพที่ 3.xx



**รูปที่ 3.xx** ที่เก็บไฟล์ log url

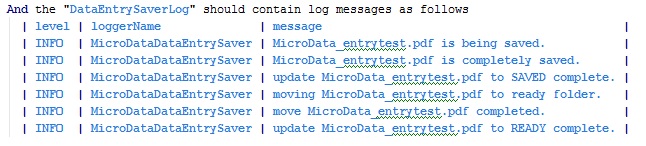
ซึ่งในไฟล์ logURL.properties นั้นจะทำการเก็บ url ที่ใช้ในการเข้าถึงไฟล์ processing log จริงในระบบ ในรูปแบบของ key value ดังรูปที่ 3.xx



**รูปที่ 3.xx** ไฟล์ logURL.properties

เมื่อ step ได้ url ที่ต้องการมาแล้วก็ทำการเข้าไปอ่านไฟล์ log จาก url ที่ได้ และทำการ เปรียบเทียบกับเซตของ log ใน feature ซึ่งถ้าเซตของ log ที่เราเตรียมไว้ตรวจสอบสำเร็จทั้งหมดภายในเวลาที่กำหนด ก็จะมีการแสดงข้อความว่าได้ตรวจสอบ log นี้แล้ว และเริ่มทำ step ต่อไป

จากรูป 3.xx Step นี้จะทำการตรวจสอบว่า process DataEntry ได้พบไฟล์ที่เราทำการ upload ขึ้นสู่ระบบ และเป็นไฟล์ที่ระบบรองรับจริง และไฟล์นั้นได้ทำการเข้าสู่สถานะ ENTERED เรียบร้อยแล้ว จากนั้น step ต่อไปในรูปที่ 3.xx จะไปตรวจสอบ log ของ process DataEntrySaver ซึ่งทำหน้าที่ในการบันทึกไฟล์ของเราลงในระบบ ว่าได้ทำการบันทึกไฟล์เรียบร้อย และทำการย้ายไฟล์ไปยัง ready folder ที่เตรียมไว้สำหรับรอ third party มาดึงไฟล์ไปทำการ data dissection สำเร็จ



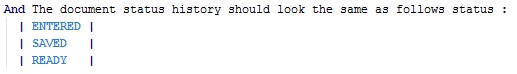
**รูปที่ 3.xx** Step ในการตรวจสอบ DataEntrySaver Log

ส่วนต่อมาคือ 2 step สุดท้ายของ scenario นี้ ซึ่งเกี่ยวข้องกับการตรวจสอบข้อมูลจริงที่ถูกบันทึกลงไปในฐานข้อมูลของระบบ MicroData โดย step แรกคือ step ในรูปที่ 3.xx

deAndDB1.jpg

**รูปที่ 3.xx** step ในการตรวจสอบสถานะ READY ของไฟล์ใน database

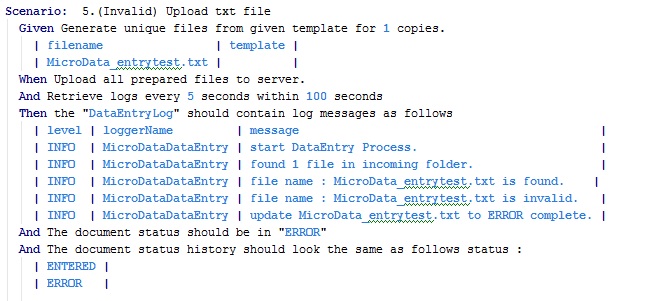
Step นี้จะทำหน้าที่ในการดึงข้อมูลของไฟล์ภาพ ที่ถูกบันทึกลงไปในฐานข้อมูล ในตาราง ไฟล์ภาพ (Image Table) ขึ้นมา แล้วทำการเอาค่าในฟิลด์ “STATUS” ซึ่งเป็นฟิลด์หนึ่งในตาราง โดยฟิลด์นี้ มีไว้สำหรับเก็บ สถานะปัจจุบันของไฟล์ว่า ขณะนี้ไฟล์ของเรากำลังอยู่ในสถานะใด เพื่อทำให้ทราบว่า ตอนนี้ไฟล์ของเราอยู่ที่ไหน กำลังอยู่ในโปรเซสใด เพื่อง่ายต่อการติดตาม เมื่อได้ค่าของฟิลด์ STATUS มาแล้วก็จะนำมาเปรียบเทียบว่า ค่าของฟิลด์ STATUS ในฐานข้อมูลที่ดึงขึ้นมานั้นว่ามีค่าเดียวกันกับค่าที่เรากำหนดไว้ใน feature file หรือไม่ ซึ่งในที่นี้ค่า STATUS ที่เราทำการ กำหนด ไว้ คือ “READY” ซึ่งหมายถึง ไฟล์พร้อมที่จะถูกนำไปทำการ dissection แล้ว เพราะฉะนั้น step นี้จึงเป็นการตรวจสอบว่า ไฟล์ของเราเมื่อผ่าน process data entry และ data entry saver แล้วสถานะของไฟล์นั้น จะต้องมีค่าเป็น READY ถ้าหากมีค่าเป็นอย่างอื่น test ก็จะ fail และเราจะทำให้เรารู้ได้ว่า ระบบมีข้อผิดพลาดภายในโปรเซสเหล่านี้ ส่วน step สุดท้ายในรูปที่ 3.xx



**รูปที่ 3.xx** step ในการตรวจสอบประวัติสถานะของไฟล์ใน database

Step นี้จะทำงานคล้ายกับ Step ก่อนหน้า โดยวิธีการทำงานของขั้นตอนนี้ก็คือ จะไปทำการดึงข้อมูลขึ้นมา เพื่อเปรียบเทียบกับข้อมูลที่เตรียมเอาไว้เช่นกัน แต่ step นี้จะทำการดึงข้อมูลจากตารางชื่อ ประวัติสถานะของไฟล์ภาพ (Image status history table) ซึ่งเป็นตารางที่ทำหน้าที่ในการบันทึกว่า ไฟล์ของเราตั้งแต่เข้ามาในระบบ จนถึงเวลาปัจจุบัน เคยผ่านการ เปลี่ยนแปลงสถานะ เป็นอะไรมาบ้าง เพื่อเป็นการเก็บ work flow ของไฟล์นั้น โดย step จะทำการดึงข้อมูลส่วนนี้ขึ้นมา และทำการเปรียบเทียบ ซึ่งมีจุดสำคัญคือ ลำดับของสถานะของไฟล์จะต้องเรียงกัน ตามสถานะที่เตรียมเอาไว้ ใน feature file ทุกประการ เพื่อเป็นการ check ว่าระบบทำงานตาม work flow ได้ถูกต้อง หากเรียงกันผิดพลาด test จะ fail และจะทำให้เรารู้ว่าระบบทำงานผิด work flow ที่ใด.

สำหรับ scenario ข้างต้นนั้นเป็นกรณีที่ทำการอัพโหลดไฟล์นามสกุล pdf ที่ระบบรองรับและใน scenario ที่ 2และ 3 จะทำงานเช่นเดียวกันกับ scenario แรก คืออัพโหลดไฟล์นามสกุล jpg และ png ซึ่งระบบรองรับทั้งคู่ การตรวจสอบจึงมีความคล้ายคลึงกัน แตกต่างกันที่ข้อความในบางจุด ทางผู้เขียนจึงขอข้ามการอธิบายในรายละเอียดของทั้ง 2 scenario ส่วนในกรณีที่ อัพโหลดไฟล์นามสกุลที่ระบบไม่รองรับ เช่น tiff และ txt จะถูกเขียนไว้ใน Scenario ที่ 4 และ 5 ตามลำดับ. สำหรับการทำงานของการตรวจสอบในกรณีที่ระบบไม่รองรับไฟล์นั้น จะยกตัวอย่างด้วย scenario ที่ 5 ดังรูปที่ 3.xx



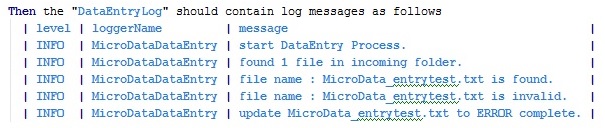
**รูปที่ 3.xx** ตัวอย่าง feature กรณีไฟล์ที่ระบบไม่รองรับ

ใน scenario นี้จะเริ่มโดยการทำการสร้างไฟล์นามสกุล .txt ซึ่งระบบไม่รองรับด้วย step Given ตามรูปที่ 3.xx

given5.jpg

รูปที่ 3.xx Step ในการสร้างไฟล์นามสกุล txt

หลักจากนั้นจะทำการอัพโหลดไฟล์ที่สร้างขึ้นไปบนเซิฟเวอร์ตามปกติ โดยในส่วนของการตรวจสอบ processing log จะมีความแตกต่างจากกรณีไฟล์ที่ระบบรองรับ คือ ข้อความที่ตรวจสอบ จะเป็นข้อความที่แสดงว่า ไฟล์มีข้อผิดพลาด และจะตรวจสอบเพียงแค่ log ของโปรเซส Data Entry เท่านั้น เนื่องจากโปรเซส Data Entry Saver ไม่มีการทำงานเกิดขึ้น เพราะว่าไฟล์ถูกเปลี่ยนเป็นไฟล์ที่มีข้อผิดพลาด และไม่ส่งต่อไปดำเนินการอื่นๆต่อจาก Data Entry. โดยข้อความที่เกิดขึ้นจะแสดงดังรูปที่ 3.xx



**รูปที่ 3.xx** Step ตรวจสอบ processing log สำหรับไฟล์ที่ระบบไม่รองรับ

หลังจากทำการตรวจสอบ processing log เสร็จเรียบร้อย step ต่อไป จะทำงานเหมือนกับ การตรวจสอบไฟล์ที่ระบบรองรับคือ การตรวจสอบสถานะของไฟล์ และ ประวัติสถานะของไฟล์ โดยใน scenario นี้จะทำการตรวจสอบสถานะของไฟล์ว่าเป็น “ERROR” หรือไม่จาก step ดังรูปที่ 3.xx

And15.jpg

**รูปที่ 3.xx** step ในการตรวจสอบสถานะ error

และหลังจากนั้นจะทำการตรวจสอบประวัติของสถานะ โดยเริ่มต้น เมื่อไฟล์เข้าสู่โปรเซส DataEntry จะมีสถานะเป็น ENTERED และหลังจากตรวจสอบว่าไฟล์ไม่รองรับ สถานะจะเปลี่ยนเป็น ERROR และจบการทำงาน ดังนั้นใน scenario นี้ ประวัติสถานะของไฟล์จะต้องเริ่มต้นด้วย ENTERED และจบลงด้วย ERROR เพียง 2 สถานะเรียงตามนี้เท่านั้น โดย step ในการตรวจสอบจะเป็นดังรูปที่ 3.xx

and25.jpg

**รูปที่ 3.xx** step ในการตรวจสอบประวัติสถานะกรณีไฟล์ไม่รองรับ

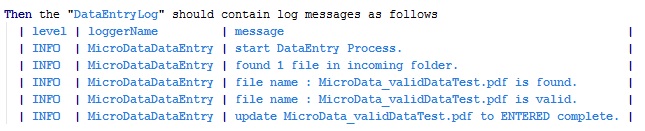
2. ValidDataValidation Feature เป็น feature file ที่ใช้ตรวจสอบการทำงานโดยรวม ของระบบ MicroData ซึ่งต่างจาก feature DataEntryValidation ตรงที่ จุดประสงค์ของ feature DataEntry นั้นมีไว้เพื่อตรวจสอบเฉพาะโปรเซส DataEntry เท่านั้น แต่ ValidaDataValidation feature จะทำการตรวจสอบตั้งแต่โปรเซส DataEntry ผ่านโปรเซส Validation จนถึงการตรวจสอบไฟล์ output ที่ถูกสร้างในภายหลัง. จึงสามารถเรียกได้ว่า feature file นี้เป็นการตรวจสอบตลอดทั้งโปรเซส หรือ end to end testing. โดยวิธีการทำงานแต่ละขั้นของ feature นี้ จะอธิบายดังต่อไปนี้.

ขั้นตอนแรกของ ValidDataValidation feature คือการสร้างไฟล์เพื่อใช้ในการส่งขึ้นไปบนเซิฟเวอร์ เหมือนกับการทดสอบใน feature ก่อนหน้า เพียงแต่ใน feature นี้เราจะสร้างเฉพาะไฟล์ นามสกุลที่ระบบรองรับเท่านั้น เพื่อให้ระบบสามารถทำงานในโปรเซสต่อๆไปได้ ในที่นี้จะใช้ ไฟล์ pdf

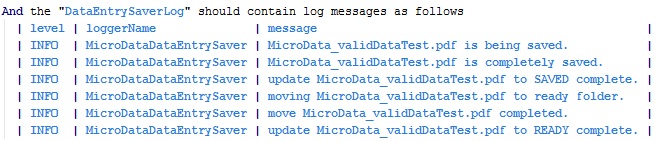
given.jpg

**รูปที่ 3.xx** Step ในการสร้างไฟล์ pdf สำหรับ Valid data feature

จากรูปที่ 3.xx เป็นการสร้างไฟล์ชื่อ MicroData\_validDataTest.pdf ด้วย default template สำหรับใช้ในการทดสอบใน scenario นี้ หลังจากสร้างไฟล์แล้วก็จะทำการอัพโหลดไฟล์สู่ระบบ ด้วย step ในการอัพโหลดไฟล์สู่ระบบ step เดียวกับรูปที่ 3.xx หลังจากนั้นก็ทำการตรวจสอบ processing log ของทั้ง 2 โปรเซสเดิมคือ Data Entry และ Data Entry Saver ว่ามี processing log ของไฟล์ MicroData\_validDataTest.pdf ที่แสดงข้อความว่าไฟล์บันทึกถูกต้อง ดังรูปที่ 3.xx และ 3.xx



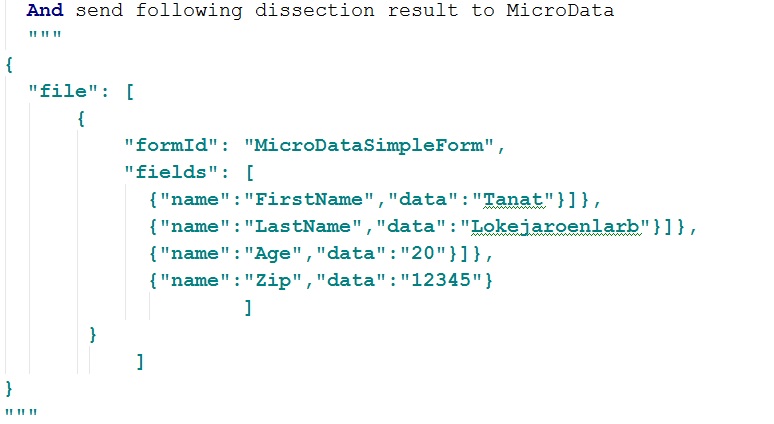
**รูปที่ 3.xx** Step ตรวจสอบ DataEntryLog ของ ValidData feature



**รูปที่ 3.xx** Step ตรวจสอบ DataEntrySaverLog ของ ValidData feature

หลังจากตรวจสอบ processing log ของ DataEntry process เสร็จเรียบร้อยจะทำให้เรารู้ได้ว่า ขณะนี้ไฟล์ของเราอยู่สถานะ READY ซึ่งคือสถานะที่พร้อมในการรอให้ third party มาดึงไฟล์ไปทำโปรเซส data dissection ซึ่งในการทำงานจริงจะต้องรอให้มีคนทำโปรเซส data dissection และโปรเซส data transcription แล้วทำการส่งข้อมูลหลังจากการแปลงไฟล์ภาพเรียบร้อยกลับมาให้ ระบบ MicroData ผ่าน RESTful web service แต่ว่า เนื่องจากซอฟต์แวร์ของเราเป็นซอฟต์แวร์ทดสอบอัตโนมัติ ทำให้เราไม่สามารถที่จะรอโปรเซสเหล่านี้ถูกทำด้วยมือได้ เราจึงจะทำการ จำลอง ให้ตัวซอฟต์แวร์ของเราเองทำการส่งข้อมูลที่ได้จากการแปลงไฟล์ (transcribed data) กลับมาที่ระบบ เนื่องจากไฟล์ที่ใช้ในการทดสอบเป็นไฟล์ของเราเองดังนั้น เราจึงสามารถที่จะรู้ได้ว่า ข้อมูลที่ควรจะได้รับกลับมาจากโปรเซส data transcription จะมีหน้าตาอย่างไร มีค่าเป็นอย่างไรบ้าง ซึ่งจะช่วยประหยัดเวลาในการทดสอบ ทำให้ไม่ต้องรอการทำ data dissection และ transcription ซึ่งใช้เวลาในการทำงาน. โดยข้อมูลที่จะทำการส่งกลับมานั้น ในปกติแล้วจะส่งกลับมาอยู่ในรูปของ JSON (JavaScript Object Notation). ดังนั้น step ที่เราจะใช้ในการจำลองการส่งข้อมูลนั้นจะเป็น

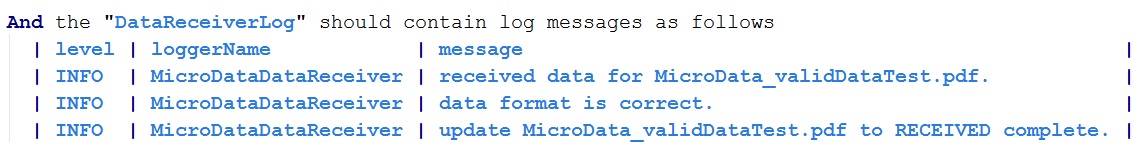
ดังรูปที่ 3.xx



**รูปที่ 3.xx** Step ในการจำลองการส่งข้อมูล

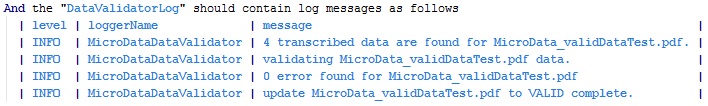
สำหรับการทำงานของ step นี้ มีหน้าที่ในการนำเอาข้อมูลในรูปแบบ JSON ที่ใส่เอาไว้ที่ด้านล่างของ step แล้วทำการส่งข้อมูลนี้ผ่าน RESTful web service ของระบบ MicroData ไปที่โปรเซส Data receiver. เมื่อทำการส่งข้อมูลของเราไปที่ MicroData แล้วขั้นตอนต่อไปก็คือขั้นตอน

การตรวจสอบ processing log ของโปรเซสต่อไป ซึ่งก็คือ Data receiver. โดยเราจะตรวจสอบว่า DataReceiver ได้รับข้อมูลที่ส่งกลับมาหรือไม่ โดยข้อความที่ใช้ในการตรวจสอบ เป็นไปดังรูปที่ 3.xx



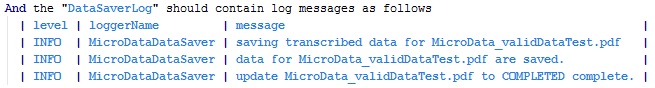
**รูปที่ 3.xx** Step ตรวจสอบ DataReceiverLog ของ ValidData feature

โดยหลักๆแล้วการตรวจสอบจะดูว่า ข้อมูลถูกส่งเข้ามาที่ระบบหรือไม่ และรูปแบบของข้อมูล JSON นั้นถูกฟอร์แมทหรือไม่ และสุดท้ายคือการตรวจสอบว่าสถานะของไฟล์เปลี่ยนเป็น RECEIVED หรือไม่. เมื่อผ่านการตรวจสอบ processing log ของโปรเซส DataReceiver แล้วแสดงว่าข้อมูลของเราได้ผ่านการตรวจสอบเบื้องต้นแล้ว ข้อมูลก็จะถูกส่งต่อไปที่โปรเซส DataValidation เพื่อทำการตรวจสอบความถูกต้องของข้อมูล โดย step ในการตรวจสอบ processing log ของ โปรเซส เป็นไปดังรูปที่ 3.xx



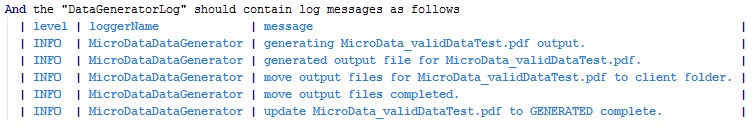
**3.xx** Step การตรวจสอบ DataValidationLog ของ ValidData feature

โดยการตรวจสอบการทำงานของโปรเซส Data validation นั้นเราจะทำการตรวจสอบข้อความว่า Data validation ได้ข้อมูลไปเท่ากับจำนวนข้อมูลที่เราส่งมาด้วย JSON หรือไม่ หลังจากนั้นจะทำการตรวจสอบว่า มีข้อผิดพลาดเกิดขึ้นหรือไม่. ซึ่งข้อผิดพลาดนี้จะเกิดขึ้นได้จากการที่ JSON ไฟล์ที่เราส่งกลับมานั้น ขัดกับข้อจำกัดที่มีการตั้งค่าไว้ โดยข้อกำหนดเหล่านี้ จะต้องทำการตั้งค่าด้วยมือเองไว้ก่อนเช่น การตั้งข้อกำหนดไว้ว่า ข้อมูลที่จะได้รับกลับมานั้นจะต้องมีฟิลด์กี่ฟิลด์ และชื่อฟิลด์มีอะไรบ้าง ซึ่งถ้าข้อมูลทีได้รับกลับมาของไฟล์ที่เราได้ตั้งข้อกำหนดนั้นไม่เป็นไปตามที่ตั้งไว้ ในขั้นตอนการตรวจสอบจะทำการมองว่าข้อมูลไม่ถูกต้อง และต้องทำการส่งไปซ่อมข้อมูลในอีกโปรเซสหนึ่ง ซึ่งในที่นี้ผู้เขียนได้ทำการตั้งค่าข้อกำหนดเบื้องต้นไว้ว่า JSON ที่ส่งมาจะต้องมี 4 ฟิลด์ และแต่ละ ฟิลด์ชื่ออะไรบ้าง เพราะฉะนั้น ในการสร้าง JSON ที่เราได้ทำการส่งมาใน step ก่อนหน้านั้น เนื่องจากเรารู้ข้อจำกัดอยู่แล้ว ดังนั้น JSON นั้นก็จะถูกสร้างโดยไม่ทำให้เกิดข้อผิดพลาดใดๆ ซึ่งข้อความที่ควรจะเกิดขึ้นใน processing log เมื่อได้รับข้อมูล JSON นั้นก็คือ “0 error found” สำหรับไฟล์ของเรา และในขั้นตอนสุดท้ายคือการตรวจสอบว่า มีข้อความที่แสดงว่า ไฟล์ของเราเปลี่ยนสถานะเป็น VALID เรียบร้อย. เมื่อข้อมูลของเราผ่านโปรเซสการตรวจสอบเรียบร้อยแล้ว โปรเซสต่อไปก็คือการบันทึกข้อมูลเหล่านั้นสำหรับไฟล์ของเรา ลงไปในฐานข้อมูล ด้วยโปรเซส DataSaver โดย step ในการตรวจสอบโปรเซส DataSaver มีลักษณะดังรูปที่ 3.xx



**รูปที่ 3.xx** Step ตรวจสอบ DataSaverLog ของ ValidData feature

การตรวจสอบข้อความของ DataSaverLog จะตรวจสอบว่ามีข้อความที่แสดงถึงการบันทึกข้อมูล และบันทึกสำเร็จหรือไม่ และหลังจากนั้นก็จะตรวจสอบในขั้นสุดท้าย ว่าสถานะของไฟล์เราได้เปลี่ยนไปเป็น COMPLETED ซึ่งแสดงว่าไฟล์ของเราได้รับข้อมูลที่ผ่านการ transcription เรียบร้อยแล้วหรือไม่. เมื่อทำการบันทึกข้อมูลเสร็จเรียบร้อยแล้ว ไฟล์ของเราจะถูกนำเข้าสู่โปรเซสสุดท้ายของ ระบบ MicroData คือโปรเซส DataGenerator ซึ่งเป็นโปรเซสที่ทำหน้าที่ในการสร้างไฟล์ผลลัพธ์ ขึ้นมาจากข้อมูลที่ได้รับ ออกมาให้กับลูกค้า. Step ที่ใช้ในการตรวจสอบโปรเซสนี้มีข้อความต่างๆดังรูปที่ 3.xx



**รูปที่ 3.xx** Step ตรวจสอบ DataGeneratorLog ของ ValidData feature

โดยข้อความหลักๆของโปรเซส Data generator ที่จะทำการตรวจสอบ จะเริ่มจาก ข้อความที่แสดงว่า ไฟล์ของเรากำลังถูกสร้างไฟล์ผลลัพธ์ และข้อความที่แสดงว่าการสร้างไฟล์สำเร็จ หลังจากนั้นก็จะตรวจสอบว่า ไฟล์ของเราได้ถูกย้ายไปยังโฟลเดอร์ของลูกค้าสำเร็จแล้วหรือไม่ ส่วนในขั้นตอนสุดท้ายก็คือ การตรวจสอบว่ามีการเปลี่ยนสถานะของไฟล์เป็น GENERATED ซึ่งเป็นสถานะสุดท้าย ซึ่งแสดงถึงว่า ไฟล์ของเราได้ถูกสร้างและส่งให้ลูกค้าสำเร็จ.

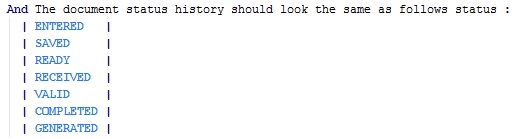
หลังจากตรวจสอบ processing log ทั้งหมดของทุกโปรเซสเป็นที่เรียบร้อยแล้ว เพื่อเป็นการ เพิ่มความแม่นยำในการตรวจสอบ ขั้นตอนต่อไปของ scenario นี้คือการตรวจสอบข้อมูลจริงในฐาน-ข้อมูล ซึ่งจะมีลักษณะการทำงานเหมือนกันกับ feature ก่อนหน้าทุกประการ ส่วนที่แตกต่างกันคือ ค่าของข้อมูลที่จะต้องสอดคล้องกับ scenario นี้ โดยขั้นตอนการตรวจสอบฐานข้อมูลขั้นตอนแรก คือ

การตรวจสอบว่า สถานะของไฟล์ ซึ่งผ่านโปรเซสทุกอย่างมาแล้วจนเสร็จสิ้น จะต้องมีสถานะเป็น GENERATED โดย step จะมีลักษณะดังรูปที่ 3.xx

statusGen.jpg

**รูปที่ 3.xx** step ในการตรวจสอบสถานะ GENERATED ของไฟล์ใน database

และขั้นตอนต่อไปในการตรวจสอบเกี่ยวกับฐานข้อมูลคือ การตรวจสอบประวัติสถานะที่เคยเกิดขึ้นกับไฟล์ โดยสถานะทั้งหมดที่ควรจะต้องเกิดขึ้นกับไฟล์ใน scenario นี้ ควรจะต้องเกิดขึ้น ครบถ้วนและ เรียงตามลำดับ ดังรูปที่ 3.xx

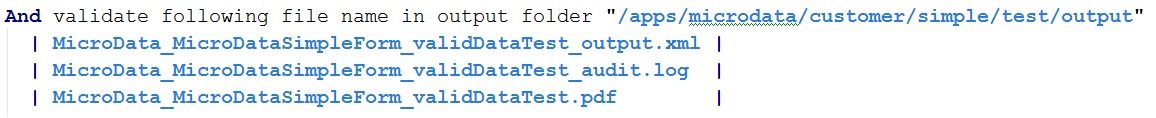


**รูปที่ 3.xx** step ในการตรวจสอบประวัติสถานะของ ValidData feature

จากรูปคือสถานะทั้งหมดที่ควรจะเกิดขึ้นถ้าการทำงานของระบบ MicroData ใน scenario นี้ทำงานได้ถูกต้อง ทุกโปรเซสที่ควรจะต้องมีการทำงาน ทำงานเรียงกันอย่างถูกต้อง

และสำหรับขั้นตอนสุดท้ายในการทดสอบระบบใน ValidDataValidation feature

คือขั้นตอนการตรวจสอบถึงไฟล์จริงๆในโฟลเดอร์ ที่เอาไว้สำหรับเก็บไฟล์ผลลัพธ์ของลูกค้า ว่าภายในโฟลเดอร์มีไฟล์ผลลัพธ์ ที่เกิดขึ้นจากข้อมูลทดสอบของเราหรือไม่ โดย step ที่ใช้ในการตรวจสอบนั้น จะมีลักษณะดังรูปที่ 3.xx

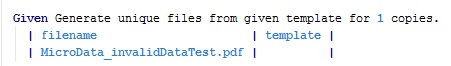


**รูปที่ 3.xx** step ในการตรวจไฟล์ผลลัพธ์

โดยการทำงานหลักๆของ step นี้ก็คือ จะเริ่มจากเชื่อมต่อกับเซิฟเวอร์ผ่าน SFTP protocol โดยการเชื่อมต่อเมื่อสำเร็จแล้วจะทำการเปลี่ยนตำแหน่งไปที่ path ที่กำหนดไว้ใน step ของเรา ซึ่งในที่นี้คือ “/apps/microdata/customer/simple/test/output” ซึ่งเป็นตำแหน่งที่เอาไว้เก็บไฟล์ผลลัพธ์ของไฟล์ของเรา การที่ต้องมาใส่ path ใน step เนื่องจากไฟล์แต่ละไฟล์ถูกกำหนดไว้สำหรับลูกค้าต่างคนกัน ดังนั้น ไฟล์ผลลัพธ์ที่ถูกสร้างขึ้นจะไปอยู่ในตำแหน่งของโฟลเดอร์เฉพาะ ซึ่งเราจะกำหนดตำแหน่งนั้นผ่านทางตัวแปรใน step นี้. เมื่ออยู่ที่ตำแหน่งโฟลเดอร์ที่เก็บไฟล์ผลลัพธ์แล้ว ก็จะทำการอ่านไฟล์ทั้งหมดในโฟลเดอร์ เพื่อนำมาเปรียบเทียบกับไฟล์ที่เรากำหนดเอาไว้ที่ด้านล่างของ step ของเรา ว่ามีไฟล์ที่เราคาดไว้อยู่ในโฟลเดอร์หรือไม่ ถ้าไม่มีแปลว่า ขั้นตอนการสร้างไฟล์ผลลัพธ์มีข้อผิดพลาด โดยชื่อไฟล์ที่ถูกสร้างจะเกิดขึ้นจากการนำเอาชื่อแบบฟอร์มรวมกับชื่อไฟล์ที่อัพโหลดขึ้นไป. เมื่อผ่านการตรวจสอบทั้งหมดทุกขั้นตอนข้างต้น จะช่วยยืนยันได้ว่า ขั้นตอนการทำงานโดย รวมของระบบ MicroData นั้นทำงานได้ถูกต้อง ในกรณีที่ ข้อมูลที่อัพโหลดรองรับโดยระบบ และข้อมูลที่ส่งกลับมาถูกต้อง ไม่ขัดกับข้อกำหนดที่ตั้งไว้.

3. RepairProcessValidation feature เป็น feature สุดท้ายของการทดสอบ โดย feature นี้เป็น feature ที่สร้างขึ้นมาเพื่อใช้ในการทำงานของระบบ MicroData ในกรณีที่มีการทำงานกับการซ่อมข้อมูลที่ไม่ถูกต้อง เนื่องจากเกิดการผิดข้อกำหนดในขั้นตอนการตรวจสอบข้อมูลใน โปรเซส Data validation โดย feature นี้จะมีหน้าที่ในการตรวจสอบว่า เมื่อมีการส่งข้อมูลที่ผิดกับ ข้อกำหนดมา โปรเซสในการตรวจสอบจะต้องตรวจพบได้ว่า ข้อมูลที่ส่งมามีข้อผิดพลาด และจะต้อง ส่งข้อมูลที่ผิดพลาดเหล่านั้นไปยังตัว data repairer ซึ่งเป็น third party ตัวหนึ่งที่ทำหน้าที่ ในการแก้ไขข้อมูล เหล่านั้นให้ถูกต้อง และหลังจากนั้นก็จะทำการตรวจสอบว่า เมื่อเสร็จสิ้นการซ่อมข้อมูลเรียบร้อยแล้ว ข้อมูลที่กลับมาจากการซ่อมจะต้องผ่านการตรวจสอบอย่างถูกต้อง และระบบ MicroData สามารถ สร้างไฟล์ผลลัพธ์ออกมาได้ในขั้นตอนสุดท้าย. โดยใน feature นี้จะแบ่งออกเป็น 2 scenario แต่ละ scenario จะมีข้อแตกต่างกันเล็กน้อยในส่วนของ JSON โดย scenario แรกจะอธิบายดังต่อไปนี้

สำหรับ scenario แรกนั้น เราจะทำการส่งข้อมูล JSON ที่มีส่วนที่ผิดพลาดคือ เราจะสร้าง ให้ข้อมูลนั้นมีฟิลด์ 2 ฟิลด์ที่หายไป จากข้อกำหนดที่เราตั้งไว้คือ 4 ฟิลด์หลังจากนั้นเราจะทำการ ซ่อมข้อมูลนั้น โดยการจำลองการทำงานเป็น third party ที่ทำการซ่อมข้อมูล หลังจากนั้นก็จะ ตรวจสอบความถูกต้องการทำงานในส่วนต่างๆ ตามที่เคยได้ทำมา. โดย step แรกของ scenario นี้ก็ จะเริ่มต้นเหมือนกับ scenario อื่นๆนั่นคือ การสร้างไฟล์เพื่อใช้ในการทดสอบ โดย step ดังรูปที่ 3.xx

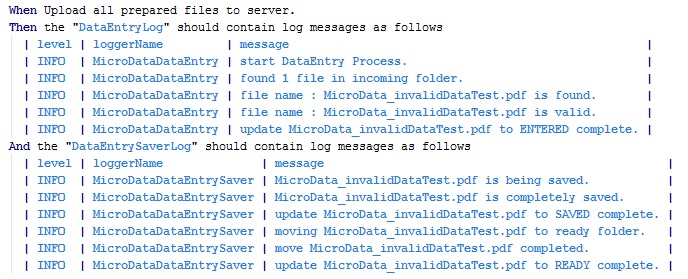


**รูปที่ 3.xx** Step ในการสร้างไฟล์ pdf สำหรับ repair process feature

โดยไฟล์ที่สร้างจะสร้างจาก default template และมีชื่อไฟล์ว่า MicroData\_invalidData

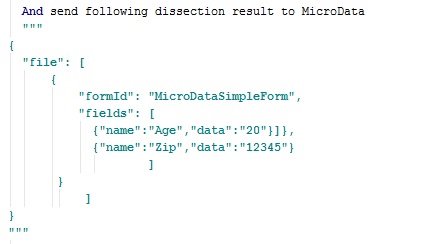
Test.pdf. หลังจากสร้างไฟล์ที่ต้องการเสร็จเรียบร้อยแล้ว step ต่อไปจากนี้คือการอัพโหลดไฟล์และ

ตรวจสอบ processing log ของโปรเซส data entry และโปรเซส data entry saver แบบเดียวกันกับ step ใน feature valid data เนื่องจากว่าไฟล์เป็นไฟล์ที่ระบบรองรับ โดยจะแสดง step ทั้งหมดในรูปที่ 3.xx



**รูปที่ 3.xx** step อัพโหลดและตรวจสอบขั้นต้นของ repair process feature

เมื่อจบขั้นตอนข้างต้นแล้ว ขั้นตอนต่อไปของ scenario คือการจำลองการทำงานของ third party ในการส่งข้อมูล transcribed data กลับมา ซึ่งข้อมูลในรูป JSON ที่เราจะส่งกลับมา เราจะจงใจสร้างให้ข้อมูลมีฟิลด์หายไป 2 ฟิลด์ จากข้อกำหนดที่เราตั้งไว้ว่า ข้อมูลที่กลับมาจะต้องมีฟิลด์ 4 ฟิลด์ โดยในกรณีนี้ฟิลด์ที่เราจะไม่ส่งกลับมาให้นั้นคือ ฟิลด์ FirstName และฟิลด์ LastName โดย step ในการส่งข้อมูลกลับมา และหน้าตาของ ข้อมูลที่อยู่ในรูป JSON ที่เราจะส่งกลับมานั้น จะแสดง ดังรูปที่ 3.xx



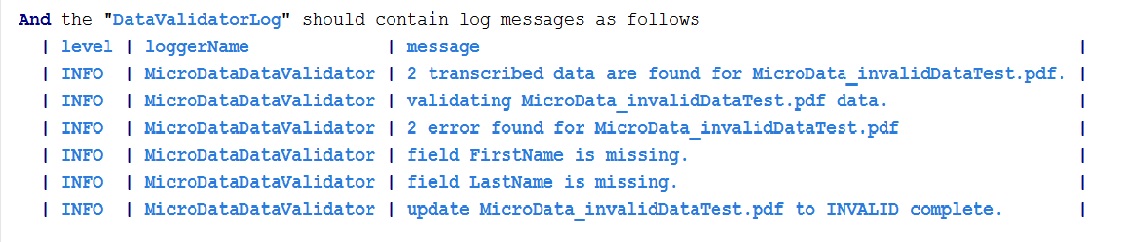
**รูปที่ 3.xx** ข้อมูล JSON ที่ฟิลด์หายไป 2 ฟิลด์

หลังจากส่งข้อมูลที่มีข้อผิดพลาดกลับมาแล้ว step ต่อไปคือการตรวจสอบ processing log ของโปรเซส data receiver ซึ่งในกรณีนี้ถึงข้อมูลที่เราส่งกลับมาจะเป็นข้อมูลที่จงใจให้ผิดพลาด แต่ในโปรเซส data receiver นั้นจะมีหน้าที่ในการตรวจสอบถึงรูปแบบของข้อมูล JSON เท่านั้น ไม่ใช่การตรวจสอบถึงการผิดข้อกำหนด เพราะฉะนั้นโปรเซส data receiver จะทำงานได้ตามปกติ ไม่มีข้อผิดพลาด ดังนั้น processing log ของโปรเซส data receiver จะมีข้อความที่แสดงว่าดังรูปที่ 3.xx



**รูปที่ 3.xx** Step ตรวจสอบ DataReceiverLog ของ repair process feature

ต่อมาในส่วนของโปรเซส data validator นั้น ใน scenario นี้ หากการทำงานของระบบถูกต้องตามปกติ การทำงานที่ควรจะเกิดขึ้นคือ โปรเซส data validator จะต้องตรวจพบว่าข้อมูลที่ ได้รับมานั้น มีข้อผิดพลาดโดย มีฟิลด์ 2 ฟิลด์หายไปจากที่ควรจะมี 4 ฟิลด์ และทำการบอกว่าฟิลด์ที่หายไปนั้นคือฟิลด์ FirstName และ LastName ตามที่เราตั้งใจ จากนั้นต้องทำการอัพเดทสถานะของไฟล์เป็น INVALID โดยข้อความที่เกิดขึ้นนั้นควรจะเป็นไปดังรูปที่ 3.xx



**3.xx** Step การตรวจสอบ DataValidationLog ของ repair process feature

หลังจากไฟล์ของเราอยู่ในสถานะ INVALID แล้วขั้นตอนต่อไปในการทำงานตามปกติก็คือ ระบบจะทำการเรียกใช้บริการของ third party ตัวนึง ที่มีส่วนติดต่อแอพพลิเคชั่นแบบ SOAP web service เพื่อส่งข้อมูลที่มีข้อผิดพลาดไปทำการซ่อม เมื่อซ่อมเสร็จแล้ว third party จะทำการส่งข้อมูลที่ซ่อมเสร็จแล้วกลับมาให้กับระบบ MicroData ด้วยการเรียกใช้บริการของ MicroData ที่เป็น SOAP web service เช่นเดียวกัน ดังนั้นในการทำงานของซอฟต์แวร์ทดสอบอัตโนมัติของเรานั้น เราจะทำการจำลองการทำงาน 2 ขั้นตอนคือ ขั้นแรกจะทำการสร้าง request ที่ใช้ในการขอซ่อมไฟล์ ในรูปแบบของ SOAP message แล้วทำการส่ง message นั้นไปหา third party และหลังจากนั้น ปกติแล้วเราจะต้องรอ third party ทำการซ่อมไฟล์ และส่งกลับมาให้เรา แต่ในที่นี้เราจะทำการจำลองการทำงานของ third party ด้วยการสร้าง SOAP message ที่มีข้อมูลที่สมบูรณ์เหมือนผ่านการซ่อมแล้วอย่างถูกต้อง พร้อมที่จะส่งกลับมาให้กับ MicroData แล้วนำ message นั้นไปวางไว้ที่ด้านล่างของ step ใน feature file ของเรา เพื่อให้ สามารถส่งข้อมูลที่ซ่อมแล้วกลับมาแบบอัตโนมัติ.

โดย step ที่ใช้ในการทำขั้นตอนข้างต้น และหน้าตาของ SOAP message ที่ใช้ในการซ่อมข้อมูล จะแสดงดังรูปที่ 3.xx



**รูปที่ 3.xx** step ในการซ่อมข้อมูล scenario ที่ฟิลด์หายไป 2 ฟิลด์

หลังจากทำการส่งข้อมูลที่ได้รับการซ่อม กลับมาที่ระบบ MicroData แล้ว โปรเซสต่อไปที่เราจะทำการตรวจสอบคือ โปรเซส data repairer

**3.4 การพัฒนาซอฟต์แวร์**