1. หากต้องการ Design Test-case มี Technique อะไรบ้างที่สามารถช่วยให้การ Design Test-case เพื่อทดสอบระบบได้ครอบคลุม

1.1 Equivalence Partitioning : การแบ่งช่วงของค่าinputเป็นกลุ่มที่มีคุณสมบัติที่เทียบเท่ากัน และเลือกทดสอบแต่ละกลุ่มอย่างน้อยหนึ่งครั้ง เพื่อลดปริมาณของการทดสอบที่ต้องทำ แต่ยังครอบคลุมทุกช่วงค่าที่สำคัญ

1.2 Boundary Value Analysis การทดสอบด้วยค่าขอบหรือค่าส่วนย่อยที่อยู่ทางข้างของช่วงค่าที่กำหนด เพื่อตรวจสอบความถูกต้องของระบบที่แยกระหว่างข้อมูลที่ถูกต้องและไม่ถูกต้อง

1.3 Decision Table Testing : การสร้างตารางที่ระบุการกระทำของระบบเมื่ออินพุตที่แตกต่างกันถูกใส่เข้าไป และทดสอบทุกสถานการณ์ที่เป็นไปได้

1.4 State Transition Testing : การทดสอบโดยการตรวจสอบการเปลี่ยนสถานะของระบบเมื่อรับอินพุตหรือเหตุการณ์ที่ต่างกัน

1.5 Use Case Testing : การสร้าง Test cases โดยอิงจาก Use cases ที่อธิบายลำดับขั้นตอนการใช้งานของระบบ

1.6 Exploratory Testing: การทดสอบโดยการสำรวจระบบโดยไม่มีแผนการทดสอบที่กำหนดล่วงหน้า โดยใช้ประสบการณ์และความรู้ของผู้ทดสอบ

1.7 Pair Testing: การทดสอบโดยการให้คู่หรือกลุ่มของผู้ทดสอบทำงานร่วมกันเพื่อค้นพบข้อผิดพลาด

1.8 Risk-Based Testing: การกำหนดลำดับความสำคัญของการทดสอบตามความเสี่ยงที่สูงที่สุดและความเสี่ยงที่น้อยที่สุด

1.9 Ad Hoc Testing : การทดสอบโดยการทดลองและค้นพบข้อผิดพลาดอย่างไม่มีแผนที่กำหนดล่วงหน้า

1.10 User Interface Testing : การทดสอบการทำงานและความสมบูรณ์ของอินเตอร์เฟซผู้ใช้

1.11 Compatibility Testing): การทดสอบระบบบนแพลตฟอร์มหรืออุปกรณ์ที่แตกต่างกัน เช่น เว็บบราวเซอร์ที่แตกต่างกันหรือระบบปฏิบัติการที่แตกต่างกัน

1.12 Localization Testing : การทดสอบการแปลและการปรับให้เข้ากับภาษาและวัฒนธรรมของพื้นที่นั้น

1.13 Regression Testing : การทดสอบเพื่อตรวจสอบว่าการเปลี่ยนแปลงใด ๆ ที่มีการนำเข้าไปในระบบไม่ได้ส่งผลกระทบต่อความสามารถในการทำงานของส่วนที่เคยทำงานได้แล้ว

1.14 Usability Testing: การทดสอบการใช้งานของผู้ใช้จริงเพื่อวัดประสิทธิภาพและความพึงพอใจในการใช้งาน

1.15 Positive Testing :การทดสอบโดยให้ข้อมูลอินพุตที่คาดหวังว่าจะทำให้ระบบทำงานถูกต้อง หรือสำเร็จตามความต้องการของระบบ

1.16 Negative Testing :การทดสอบโดยให้ข้อมูลอินพุตที่ไม่คาดหวัง หรือที่ไม่ถูกต้องตามเงื่อนไขของระบบ

2.Design Test-case จากโจทย์ต่อไปนี้อย่างน้อย 5 Case พร้อมระบุ Technique ที่ใช้ข้อนั้น ๆ

[โจทย์] : ผู้ใช้ต้องการโอน Point จากบัญชีตัวเอง ไปยังบัญชีปลายทาง โดยเงื่อนไขคือ
ขั้นต่ำในการโอน Point คือ 100 / การทำรายการ
สูงสุดในการโอน Point คือ 3,000 / การทำรายการ
หากโอน < ขั้นต่ำ ระบบคิดค่า Fee 8 Point โดยบวกเพิ่มจากค่าที่กรอก
ต้องกรอก Passcode 4 หลัก ให้ถูกต้องจึงทำรายการสำเร็จ
บัญชีปลายทางต้องถูกต้อง จึงจะสามารถกรอก Passcode ได้

Test1: Input=101 Passcode Correct,Address Correct
Expected Output passed
Technique: Positive Testing
ใช้เพื่อทดสอบกรณีที่ Input ถูกต้องทั้ง Passcode และ Address

Test2: Input=101 Passcode inCorrect,Address Correct
Expected Output failed
Technique: Negative Testing (Passcode)
ใช้เพื่อทดสอบกรณีที่ Passcode ไม่ถูกต้องแต่ Address ถูกต้อง

Test3: Input=101 Passcode inCorrect,Address inCorrect
Expected Output failed
Technique: Negative Testing (Passcode, Address)
ใช้เพื่อทดสอบกรณีที่ Passcode และ Address ไม่ถูกต้อง

Test4: Input=101 Passcode Correct,Address Correct
Expected Output failed
Technique: Negative Testing (Address)
ใช้เพื่อทดสอบกรณีที่ Address ไม่ถูกต้องแต่ Passcode,Address ถูกต้อง

Test5: Input=90 Passcode Correct,Address Correct
Expected Output Passed but + fee 8 points
Technique: Boundary Value Analysis + Positive Testing
ใช้เพื่อทดสอบกรณีที่จำนวน Point เป็นขั้นต่ำแต่ Passcode,Adress ถูกต้อง

Test6: Input=90 Passcode inCorrect,Address Correct
Expected Output failed
Technique: Boundary Value Analysis + Negative Testing (Passcode)

ใช้เพื่อทดสอบกรณีที่จำนวน Point เป็นขั้นต่ำแต่ Passcode ไม่ถูกต้อง

Test7: Input=90 Passcode Correct,Address inCorrect
Expected Output failed
Technique: Boundary Value Analysis + Negative Testing (Address)
ใช้เพื่อทดสอบกรณีที่จำนวน Point เป็นขั้นต่ำแต่ Address ไม่ถูกต้อง

Test8: Input=90 Passcode inCorrect,Address inCorrect
Expected Output failed
Technique: Boundary Value Analysis + Negative Testing (Passcode, Address)
ใช้เพื่อทดสอบกรณีที่จำนวน Point เป็นขั้นต่ำและทั้ง Passcode และ Address ไม่ถูกต้อง

Test9: Input=3000 Passcode Correct,Address Correct
Expected Output passed
Technique: Boundary Value Analysis + Positive Testing
ใช้เพื่อทดสอบกรณีที่จำนวน Point เป็นขั้นสูงสุดและ Passcode และ Address ถูกต้อง

Test10: Input=3000 Passcode inCorrect,Address Correct
Expected Output failed
Technique: Boundary Value Analysis + Negative Testing (Passcode)
ใช้เพื่อทดสอบกรณีที่จำนวน Point เป็นขั้นสูงสุด แต่ Passcode ไม่ถูกต้อง และ Address ถูกต้อง

Test11: Input=3000 Passcode inCorrect,Address inCorrect
Expected Output failed
Technique: Boundary Value Analysis + Negative Testing (Passcode, Address)
ใช้เพื่อทดสอบกรณีที่จำนวน Point เป็นขั้นสูงสุด และทั้ง Passcode และ Address ไม่ถูกต้อง

Test12: Input=3000 Passcode Correct,Address inCorrect
Expected Output failed
Technique: Boundary Value Analysis + Negative Testing (Address)
ใช้เพื่อทดสอบกรณีที่จำนวน Point เป็นขั้นสูงสุด แต่ Address ไม่ถูกต้อง และ Passcode ถูกต้อง

Test13: Input=3001 Passcode Correct,Address Correct
Expected Output failed
Technique: Boundary Value Analysis + Negative Testing (Passcode, Address)
ใช้เพื่อทดสอบกรณีที่จำนวน Point เกินขั้นสูงสุด และทั้ง Passcode และ Address ถูกต้อง

Test14: Input=3001 Passcode inCorrect,Address Correct
Expected Output failed
Technique: Boundary Value Analysis + Negative Testing (Passcode)
ใช้เพื่อทดสอบกรณีที่จำนวน Point เกินขั้นสูงสุด แต่ Passcode ไม่ถูกต้อง และ Address ถูกต้อง

Test15: Input=3001 Passcode Correct,Address inCorrect
Expected Output failed
Technique: Boundary Value Analysis + Negative Testing (Address)
ใช้เพื่อทดสอบกรณีที่จำนวน Point เกินขั้นสูงสุด และ Address ไม่ถูกต้อง และ Passcode ถูกต้อง

Test16: Input=3001 Passcode inCorrect,Address inCorrect
Expected Output failed
Technique: Boundary Value Analysis + Negative Testing (Passcode, Address)
ใช้เพื่อทดสอบกรณีที่จำนวน Point เกินขั้นสูงสุด และทั้ง Passcode และ Address ไม่ถูกต้อง

3.
3.1 เรียกประชุมเพื่อวางแผน:
ทีมทุกคนที่เกี่ยวข้อง เช่น dev, tester, Pm เพื่อรับฟังความต้องการและการกำหนดคุณภาพของ Feature

3.2 ระบุขอบเขตและวัตถุประสงค์ของการทดสอบ:
กำหนดขอบเขตของ Feature ที่ต้องการทดสอบ
ระบุวัตถุประสงค์หลักของการทดสอบ เช่น การตรวจสอบความถูกต้องของการทำงาน, การตรวจสอบประสิทธิภาพ, การตรวจสอบความปลอดภัย เป็นต้น

3.3 ระบุ Test Cases ที่เกี่ยวข้อง:
สร้างหรือระบุ Test Cases ที่ต้องการทดสอบ Feature นั้น ๆ โดยละเอียด
ระบุเงื่อนไขการทดสอบ ข้อมูลอินพุต, ผลลัพธ์ที่คาดหวัง, และเหตุผลที่จำเป็น

3.4 กำหนดรายละเอียดของการทดสอบ:
ระบุเครื่องมือหรือเทคโนโลยีที่ใช้ในการทดสอบ เช่น Automated Testing Tools, Manual Testing Process
ระบุขั้นตอนการทดสอบเพื่อให้ทุกคนในทีมเข้าใจว่าควรทำอย่างไร

3.5กำหนดเงื่อนไขการผ่านการทดสอบ:
ระบุวิธีที่ใช้ในการตรวจสอบผลลัพธ์ของการทดสอบว่ามีการผ่านหรือไม่
ระบุเงื่อนไขสำหรับการตรวจสอบผลลัพธ์ที่คาดหวังในแต่ละ Test Case

3.6กำหนดแผนการทดสอบ:
ระบุเป้าหมายของการทดสอบ รวมถึงกำหนดเวลา, ทรัพยากร, และบุคลากรที่เกี่ยวข้อง
สร้างแผนการทดสอบที่เป็นไปได้ รวมถึงกำหนดเวลาและวันที่การทดสอบ

3.7ตรวจสอบและอนุมัติ:
ตรวจสอบและปรับปรุง Test Plan ตามความเห็นของทุกคนที่เข้าร่วมประชุม
อนุมัติ Test Plan เพื่อเริ่มการทดสอบ

3.8ดำเนินการทดสอบ:
ทำตามขั้นตอนและ Test Cases ที่ระบุใน Test Plan
บันทึกผลการทดสอบและรายงานประจำวัน

3.9วิเคราะห์ผลการทดสอบ:
ตรวจสอบผลการทดสอบเพื่อหาข้อผิดพลาดและปัญหาที่เกิดขึ้น
ปรับปรุงและดำเนินการทดสอบในครั้งถัดไปตามความจำเป็น

3.10สรุปและรายงาน:
สรุปผลการทดสอบและสร้างรายงานสำหรับผู้บริหารและทีมพัฒนา
ให้ข้อเสนอแนะเพื่อปรับปรุง Feature หรือการทดสอบในอนาคต

4.
4.1 ความน่าเชื่อถือ: การทดสอบช่วยให้เรามั่นใจได้ว่าซอฟต์แวร์ทำงานตามคาดหวังและมีประสิทธิภาพตามที่ต้องการ ทำให้ผู้ใช้งานมั่นใจและเชื่อมั่นในการใช้งานซอฟต์แวร์นั้นๆ

4.2 ประหยัดเวลาและค่าใช้จ่าย: การตรวจสอบและแก้ไขข้อผิดพลาดในขั้นตอนแรกของการพัฒนาจะช่วยประหยัดเวลาและค่าใช้จ่ายในการแก้ไขในภายหลัง เนื่องจากการแก้ไขข้อผิดพลาดในขั้นตอนหลังการนำซอฟต์แวร์ไปใช้งานอาจทำให้ต้องใช้เวลาและทรัพยากรมากขึ้น

4.3 ปรับปรุงคุณภาพของซอฟต์แวร์: การทดสอบช่วยให้รู้ว่าซอฟต์แวร์มีคุณภาพที่ดีและปลอดภัยตามมาตรฐานที่กำหนด ทำให้ผู้ใช้งานได้รับประสิทธิภาพและประสบการณ์การใช้งานที่ดีกับซอฟต์แวร์นั้นๆ

4.4 เพิ่มความเชื่อมั่นในซอฟต์แวร์: การทดสอบที่ดีช่วยเพิ่มความเชื่อมั่นในซอฟต์แวร์ ทำให้ผู้ใช้งานมีความเชื่อมั่นในการที่ซอฟต์แวร์จะทำงานได้อย่างเสถียรและปลอดภัย

4.5 ประหยัดเวลาและทรัพยากร: การทดสอบช่วยลดการตรวจสอบและแก้ไขข้อผิดพลาดในซอฟต์แวร์ที่เกิดขึ้นในขั้นตอนหลังการนำซอฟต์แวร์ไปใช้งาน ซึ่งจะทำให้ประหยัดเวลาและทรัพยากรในระยะยาว

4.6 เพิ่มความเชื่อถือในทีมพัฒนา: การทดสอบช่วยเพิ่มความมั่นใจในทีมพัฒนา ทำให้สมาชิกในทีมมั่นใจว่าซอฟต์แวร์ที่พัฒนาขึ้นมีคุณภาพและปลอดภัยตามมาตรฐานที่กำหนด