ภาคผนวก ค

การพัฒนาระบบ

คุณสมบัติเฉพาะของการพัฒนาระบบ

๑. การวางแผน (Planning)

๑.๑. แผนการดำเนินการโครงการอย่างละเอียด

๑.๒. แผนการศึกษา วิเคราะห์ และเก็บข้อมูลเกี่ยวกับความต้องการในการใช้งานระบบงาน

๑.๓. แผนการออกแบบและพัฒนาระบบงานตามความต้องการใช้งานของระบบงาน

๑.๔. แผนการจัดทำคู่มือและการฝึกอบรมการใช้งานและปฏิบัติงานกับระบบงาน

๑.๕. แผนการนำระบบไปใช้จริง

๑.๖. แผนการใช้ทรัพยากรบุคคลในโครงการ

๒. การรวบรวมความต้องการ (Requirement Gathering)

๒.๑. ผู้รับจ้างต้องดำเนินการศึกษา เก็บรวบรวมความต้องการ (Requirement) ขั้นตอน และกระบวนการทำงานที่เกี่ยวข้องสำหรับการพัฒนาแพลตฟอร์มการตรวจเงินแผ่นดินจากสำนักงานการตรวจเงินแผ่นดินและแหล่งข้อมูลอื่นที่น่าเชื่อถือได้ เพื่อใช้ร่วมในการวิเคราะห์และออกแบบระบบได้อย่างครบถ้วนถูกต้องพร้อมทั้งสรุปเป็นเอกสารข้อกำหนดความต้องการระบบ

๒.๒. ผู้รับจ้างต้องจัดทำเอกสารข้อกำหนดความการระบบ (System Requirement Specification - SRS)

๒.๒.๑. จัดทำเป็นไฟล์ PDF และไฟล์แก้ไขได้ เช่น Word หรือเทียบเท่า

๒.๒.๒. ต้องมีสารบัญ (Table of Contents) และการจัดลำดับหัวข้อที่ชัดเจนตามมาตรฐาน IEEE 830 หรือมาตรฐานอื่นที่เทียบเท่า พร้อมแสดง Check list เพื่อตรวจสอบการปฏิบัติตามมาตรฐาน

๒.๓. ผู้รับจ้างต้องจัดทำเอกสารวิเคราะห์และออกแบบระบบ (System Design Specification - SDS) ตามรูปแบบที่สำนักงานการตรวจเงินแผนดินกำหนด และมีแผนภาพ (Diagram) อย่างน้อยดังนี้

๒.๓.๑. ผู้รับจ้างต้องจัดทำ Prototype ของระบบ โดยจัดทำรูปแบบไฟล์แก้ไขได้ เช่น Figma หรือเทียบเท่า

๒.๓.๒. ผู้รับจ้างต้องจัดทำ Architecture Diagram เพื่อแสดงภาพรวมของระบบ รวมถึงการเชื่อมต่อระหว่างองค์ประกอบต่าง ๆ และการออกแบบที่รองรับ High Availability และ Scalability

๒.๓.๓. ผู้รับจ้างต้องจัดทำ Use Case Diagram เพื่อแสดงการทำงานของระบบในมุมมองของผู้ใช้ รวมถึงระบุ Actor และ Use Case ทั้งหมด เพื่อใช้เป็นข้อมูลประกอบการวิเคราะห์และออกแบบระบบ

คณพศ หงสวรางกูร ปริญญา เพ็ญไพรัตน์กุล ธีรยุทธ วัฒนะธรรมศิริ ศิร์รัฐ ศรีสุนทรพินิต สาลิเดช สัมฤทธิ์
มธุรส ไพบูลย์วรชาติ วิมลมาศ วัฒนจรูญโรจน์ รัฐธรรม ชวนเชย พันธุ์นิชิต โปษยานนท์ นัฐพล สีสวย
วิชาญ จินดารัตน์ สิทธิศักดิ์ ไชยสุข อำนาจวิทย์ หมู่ศิลป์ จุฑามาศ วงศ์เหมอนันต์ ทักษิณ หลังสตา

๒.๓.๔. ผู้รับจ้างต้องจัดทำ Activity Diagram เพื่อแสดงลำดับการทำงานของแต่ละกระบวนการในระบบ ตั้งแต่การเริ่มต้นจนสิ้นสุด รวมถึงการตรวจสอบเงื่อนไขและการตัดสินใจ

๒.๓.๕. ผู้รับจ้างต้องจัดทำ State Diagram สำหรับ Object ที่มีการเปลี่ยนสถานะ เพื่อแสดงการเปลี่ยนผ่านของสถานะในกรณีต่าง ๆ เช่น การอนุมัติ การยกเลิก หรือการเปลี่ยนสถานะคำสั่งซื้อ

๒.๓.๖. ผู้รับจ้างต้องจัดทำ Sequence Diagram เพื่อแสดงลำดับการทำงานและการสื่อสารระหว่าง Object หรือ Component ภายในระบบ รวมถึงการเรียกใช้ API หรือ External Service

๒.๓.๗. ผู้รับจ้างต้องจัดทำ Class Diagram เพื่อแสดงโครงสร้างของ Class, Attribute, Method และความสัมพันธ์ระหว่าง Class ต่าง ๆ ในระบบที่ใช้แนวคิด Object-Oriented Programming

๒.๓.๘. ผู้รับจ้างต้องจัดทำ Component Diagram เพื่อแสดงการจัดวางและการเชื่อมต่อระหว่าง Component ภายในระบบ รวมถึงการสื่อสารระหว่าง Service, API Gateway และ Database

๒.๓.๙. ผู้รับจ้างต้องจัดทำ Deployment Diagram เพื่อแสดงการติดตั้งระบบในแต่ละ Environment (Dev, UAT, Production) รวมถึงการใช้ Load Balancer, Firewall, และ Cloud Services ที่เกี่ยวข้อง

๒.๓.๑๐. ผู้รับจ้างต้องจัดทำเอกสารแผนการ Deployment และ Workflows พร้อมตัวอย่างการทดสอบในแต่ละ Environment ด้วย

๒.๓.๑๑. ผู้รับจ้างต้องจัดทำ ER Diagram เพื่อแสดงโครงสร้างฐานข้อมูล รวมถึง Entity, Attribute และ Relationship ระหว่าง Table เพื่อให้มั่นใจว่าการจัดเก็บและเรียกใช้ข้อมูลมีประสิทธิภาพ

๒.๓.๑๒. ผู้รับจ้างต้องจัดทำเอกสาร DevSecOps Workflow Diagram หรือ Pipeline Flow ที่แสดงขั้นตอนทั้งหมด พร้อมอธิบายวัตถุประสงค์ของแต่ละขั้น และระบุว่าจะตรวจสอบความถูกต้องได้อย่างไร

๒.๓.๑๓. ต้องจัดทำเอกสารแสดงแผนผังการเชื่อมโยง (Integration Architecture) กับระบบ Monitoring โดยแสดงตัวอย่างการตั้งค่าระบบ Monitoring

๒.๓.๑๔. ผู้รับจ้างต้องจัดทำเอกสารตามข้อ ๒.๓.๒ ถึงข้อ ๒.๓.๑๓ ในรูปแบบดังต่อไปนี้

๑) ไฟล์ PDF และไฟล์แก้ไขได้ (เช่น Word) หรือใช้ Web-based Documentation/ Version Control System (เช่น Confluence, GitLab Pages, MkDocs) ในการจัดเก็บและควบคุมเวอร์ชั่นเอกสารต้นฉบับ (ไฟล์ต้นฉบับ Diagram เช่น draw.io, Mermaid, PlantUML) แทนการส่งไฟล์หลายฟอร์แมต

๒) Diagram ทั้งหมดจัดทำในรูปแบบไฟล์ที่แก้ไขได้ เช่น .drawio, .svg

๓) แนบไฟล์ภาพ เช่น PNG, JPEG เพื่อความสะดวกในการอ้างอิงและตรวจสอบ

๒.๔. ผู้รับจ้างต้องจัดทำเอกสารผลการทดสอบของระบบทั้งหมด รวมถึงแยกผลการทดสอบของซอฟต์แวร์สำเร็จรูป (Commercial Off-The-Shelf - COTS) ออกจากระบบที่พัฒนาขึ้นเอง โดยต้องระบุว่าใน

แต่ละกรณีสามารถทดสอบอะไรได้/ไม่ได้ พร้อมเหตุผลประกอบ และเอกสารต้องสามารถตรวจสอบย้อนหลังได้ (audit trail) โดยจัดทำในรูปแบบดังต่อไปนี้

๒.๔.๑. จัดทำเป็นไฟล์ PDF และไฟล์ Word ที่แก้ไขได้

๒.๔.๒. ไฟล์รายงานผลการทดสอบเชิงเทคนิค เช่น Log Files, CSV, JSON หรือ Excel ต้องแนบในรูปแบบไฟล์ดิบ (raw files) เพื่อให้ตรวจสอบย้อนหลัง (audit trail) ได้

๒.๔.๓. สำหรับผลการทดสอบซอฟต์แวร์สำเร็จรูป (Commercial Off-The-Shelf - COTS) ต้องแยกรายงานออกมาต่างหาก โดยใช้รูปแบบเดียวกัน

๒.๕. ในกรณีที่สำนักงานการตรวจเงินแผ่นดินมีการขอแก้ไขหรือปรับปรุงการออกแบบที่ได้จัดทำและส่งมอบแล้ว ผู้รับจ้างต้องดำเนินการแก้ไขและปรับปรุงเอกสารที่เกี่ยวข้องทั้งหมด (เช่น SRS, SDS, Wireframe, Diagram ต่าง ๆ) ให้สอดคล้องกับการแก้ไขนั้นโดยไม่คิดค่าใช้จ่ายเพิ่มเติม ทั้งนี้ การแก้ไขต้องอยู่ภายในขอบเขตงานเดิมที่ระบุไว้ใน TOR และไม่เกินกว่าขอบเขตงานเดิมหรือเงื่อนไขที่ผู้รับจ้างได้เสนอราคาไว้ หากการแก้ไขหรือปรับปรุงมีขอบเขตเกินกว่างานเดิมที่ตกลงกันไว้ หรือเป็นการเปลี่ยนแปลงสาระสำคัญของระบบ ผู้ว่าจ้างและผู้รับจ้างจะต้องหารือร่วมกันเพื่อกำหนดวิธีการดำเนินงานและการชดเชยที่เป็นธรรมต่อทั้งสองฝ่าย

๒.๖. จัดทำเอกสารอธิบายสิทธิ์การใช้งาน (Role Metric)

๒.๗. การจัดการสิทธิ์การใช้งานซอฟต์แวร์และไลบรารี (Software License Compliance) เพื่อให้มั่นใจว่าการใช้งานซอฟต์แวร์ในโครงการถูกต้องตามกฎหมาย และไม่มีความเสี่ยงด้านลิขสิทธิ์ ผู้รับจ้างต้องรับรองว่าซอฟต์แวร์ ไลบรารี หรือส่วนประกอบจากบุคคลที่สามที่นำมาใช้ มีการจัดการสิทธิ์การใช้งานอย่างเหมาะสม โดยมีข้อกำหนดอย่างน้อยดังนี้

๒.๗.๑. ซอฟต์แวร์และไลบรารีทั้งหมดที่นำมาใช้ต้องมีสิทธิ์การใช้งานที่ถูกต้องตามกฎหมาย และสอดคล้องกับรูปแบบการใช้งานในโครงการ หากเป็นซอฟต์แวร์โอเพนซอร์ส ต้องใช้ License ที่ได้รับการรับรองโดย Open Source Initiative (OSI) เช่น MIT, BSD, Apache ๒.๐ หรือที่เทียบเท่า

๒.๗.๒. ต้องจัดทำรายการซอฟต์แวร์ ไลบรารี และส่วนประกอบที่ใช้งานทั้งหมดในโครงการ โดยระบุชื่อ เวอร์ชั่น และประเภท License ที่เกี่ยวข้อง เพื่อใช้เป็นหลักฐานประกอบการตรวจสอบ

๒.๗.๓. ต้องปฏิบัติตามเงื่อนไขของ License อย่างเคร่งครัด เช่น การแสดงชื่อผู้พัฒนาเดิม การแนบสำเนา License และการระบุการเปลี่ยนแปลงที่เกิดขึ้นจากผู้รับจ้าง (ถ้ามี)

๒.๗.๔. อนุญาตให้ใช้ซอฟต์แวร์หรือไลบรารีภายใต้สัญญาอนุญาตแบบ GPL หรือ AGPL ได้ กรณีที่

๒.๗.๔.๑. การใช้งานเป็นการเรียกผ่าน API, Web Service หรือเป็นโมดูลแยกส่วน (ไม่ผนวกโค้ดเข้าในระบบหลักโดยตรง)

๒.๗.๔.๒. ไม่ส่งผลให้ซอร์สโค้ดของระบบหลักต้องถูกเปิดเผยหรือกระทบสิทธิ์การใช้งานเชิงพาณิชย์

๒.๗.๔.๓. ผู้รับจ้างต้องจัดทำ “รายการซอฟต์แวร์/ไลบรารี” พร้อมรายละเอียดวิธีการใช้งาน และส่งให้ผู้ว่าจ้างพิจารณาและขออนุมัติล่วงหน้า

๒.๗.๕. ผู้รับจ้างต้องใช้เครื่องมือหรือกระบวนการที่เหมาะสมในการตรวจสอบและติดตามการปฏิบัติตามสิทธิ์การใช้งานซอฟต์แวร์ (Software License Compliance) อย่างต่อเนื่อง เช่น การใช้ Software Composition Analysis (SCA) Tools

๓. การออกแบบ (Design)

๓.๑. ผู้รับจ้างต้องออกแบบระบบตามคุณสมบัติเฉพาะตามภาคผนวก ก รายละเอียดคุณลักษณะเฉพาะของระบบสารสนเทศของโครงการ สามารถปรับเปลี่ยนตามความต้องการของสำนักงานการตรวจเงินแผ่นดิน และต้องเป็นไปตามกฎหมาย ระเบียบ ประกาศ ข้อกำหนด และมาตรฐาน ดังนี้

๓.๑.๑. พระราชบัญญัติว่าด้วยการกระทำความผิดเกี่ยวกับคอมพิวเตอร์ พ.ศ. ๒๕๖๐

๓.๑.๒. พระราชบัญญัติคุ้มครองข้อมูลส่วนบุคคล พ.ศ. ๒๕๖๒

๓.๑.๓. พระราชบัญญัติการรักษาความมั่นคงปลอดภัยไซเบอร์ พ.ศ. ๒๕๖๒

๓.๑.๔. พระราชบัญญัติว่าด้วยธุรกรรมทางอิเล็กทรอนิกส์ พ.ศ. ๒๕๔๔

๓.๑.๕. มาตรฐานเว็บไซต์ภาครัฐ (Government Website Standard)

๓.๑.๖. มาตรฐานรัฐบาลดิจิทัล ว่าด้วยกรอบแนวทางการพัฒนามาตรฐานการเชื่อมโยงและแลกเปลี่ยนข้อมูลภาครัฐ (Thailand Government Information eXchange :TGIX)

๓.๑.๗. มาตรฐานรัฐบาลดิจิทัล ว่าด้วยแนวทางการเปิดเผยข้อมูลเปิดภาครัฐในรูปแบบดิจิทัลต่อสาธารณะ (Open Government Data Guideline)

๓.๑.๘. มาตรฐานและหลักเกณฑ์การจัดทำกระบวนการและการดำเนินงานทางดิจิทัลว่าด้วยเรื่องการใช้ดิจิทัลไอดีสำหรับบริการภาครัฐ สำหรับบุคคลธรรมดาที่มีสัญชาติไทย (Digital ID)

๓.๑.๙. กฎหมาย ระเบียบ ประกาศ ข้อกำหนด และมาตรฐานอื่น ๆ ที่เกี่ยวข้องกับการพัฒนาแพลตฟอร์มภาครัฐ

๓.๒. ผู้รับจ้างต้องนำแนวคิดของ Domain-Driven Design (DDD) มาประยุกต์ใช้ในการออกแบบและพัฒนาระบบ โดยครอบคลุมแนวทางหลัก อย่างน้อยดังต่อไปนี้

๓.๒.๑. ต้องเริ่มต้นจากการวิเคราะห์ปัญหาในมุมมองของธุรกิจ (Problem Space) และสร้างแบบจำลองของระบบจาก Business Domain จริง

๓.๒.๒. ใช้ Model แทน Business Entity อย่างถูกต้อง โดยต้องมีการใช้ Model แสดงความสัมพันธ์ระหว่าง Entity ต่าง ๆ เช่น User, Permission อย่างชัดเจน และสะท้อนการทำงานทางจริง

คณพศ หงสวรางกูร ปริญญา เพ็ญไพรัตน์กุล ธีรยุทธ วัฒนะธรรมศิริ ศิร์รัฐ ศรีสุนทรพินิต สาลิเดช สัมฤทธิ์
มธุรส ไพบูลย์วรชาติ วิมลมาศ วัฒนจรูญโรจน์ รัฐธรรม ชวนเชย พันธุ์นิชิต โปษยานนท์ นัฐพล สีสวย
วิชาญ จินดารัตน์ สิทธิศักดิ์ ไชยสุข อำนาจวิทย์ หมู่ศิลป์ จุฑามาศ วงศ์เหมอนันต์ ทักษิณ หลังสตา

๓.๒.๓. แยกขอบเขตบริบท (Bounded Context) โดยแยกโมดูลหรือบริการตาม Context ของแต่ละโดเมน เช่น User Context, Permission Context เพื่อป้องกันความซับซ้อนและลดการพึ่งพากันระหว่างโมดูล

๓.๒.๔. จัดกลุ่ม Aggregates เพื่อคง Consistency โดยใช้ Aggregate Root ควบคุมกลุ่มของ Object ที่ต้อง Consistent กัน เช่น User เป็น Root ของ UserDetail

๓.๒.๕. แยก Entity และ Value Object อย่างชัดเจน เช่น

๓.๒.๕.๑. Entity : มี ID และสามารถเปลี่ยนแปลงสถานะได้

๓.๒.๕.๒. Value Object : ไม่มี ID และไม่เปลี่ยนแปลงสถานะ (immutable) เช่น Address

๓.๒.๖. ใช้แนวคิด Layered Architecture โดยแบ่งระบบออกเป็น ๔ ชั้น ได้แก่ Presentation, Application, Domain, Infrastructure เพื่อให้โครงสร้างมีความชัดเจนและแยกความรับผิดชอบกัน

๓.๒.๗. ออกแบบ Model และ Use Case ตามเหตุการณ์ (Event Storming) ให้มีการระบุ Use Case และเหตุการณ์ (Event) ทางธุรกิจอย่างชัดเจน เช่น "Create User ", "Disable User"

๓.๒.๘. มี Layer สำหรับจัดการ Object เช่น Factory, Repository และ Service

๓.๒.๘.๑. Factory: สำหรับสร้าง Object

๓.๒.๘.๒. Service: สำหรับประมวลผล Logic

๓.๒.๘.๓. Repository: สำหรับจัดการ Persistence เช่น DB

๓.๓. ผู้รับจ้างระบบต้องออกแบบโดยใช้สถาปัตยกรรมแบบยืดหยุ่นและขยายขนาดได้ (Scalable Architecture) ในลักษณะแบบแยกส่วน (Modular Design) เพื่อให้รองรับจำนวนผู้ใช้พร้อมกันจำนวนมากได้อย่างมีประสิทธิภาพ และสนับสนุนการพัฒนาและบำรุงรักษาได้ง่าย โดยอาจใช้แนวทางการพัฒนาแบบ Microservices หรือวิธีอื่นที่ตอบโจทย์ด้านการขยายระบบและการรองรับผู้ใช้จำนวนมากในอนาคตโดยมีคุณสมบัติอย่างน้อยดังนี้

๓.๓.๑. มีคุณสมบัติ Single Responsibility โดยแต่ละ Service ย่อยต้องรับผิดชอบเพียงเรื่องเดียว (Single Concern) เพื่อให้ง่ายต่อการดูแล ปรับปรุง และขยายในอนาคต เช่น บริการ “User Service” จัดการเฉพาะเรื่องข้อมูลผู้ใช้งานเท่านั้น เช่น การดึงข้อมูล การเพิ่ม การแก้ไข

๓.๓.๒. ใช้ Lightweight Protocols

๓.๓.๒.๑. ใช้ REST, gRPC หรือ Message Queue (MQTT, Kafka) ตามความเหมาะสม

๓.๓.๒.๒. หลีกเลี่ยงการใช้ SOAP หรือโปรโตคอลหนักโดยไม่จำเป็น

๓.๓.๓. ต้องรองรับ Service Discovery

๓.๓.๓.๑. ต้องใช้ระบบ Service Registry เช่น Consul, Eureka หรือ Kubernetes Service

๓.๓.๓.๒. เพื่อให้สามารถค้นหาและเชื่อมต่อระหว่าง Services ได้แบบ Dynamic

๓.๓.๔. มีคุณสมบัติ Containers โดยระบบต้องออกแบบให้แต่ละ Service ย่อยถูกบรรจุและจัดการภายใน container เช่น Docker เพื่อให้สามารถนำไปติดตั้ง (Deploy) บนเครื่องเซิร์ฟเวอร์ใดก็ได้ ไม่ว่าจะเป็นสภาพแวดล้อม Dev, UAT หรือ Production โดยไม่ต้องปรับแต่งหรือแก้ไขการตั้งค่าของเครื่องนั้น ทำให้มั่นใจได้ว่าการทำงานของ service จะเหมือนกันทุกสภาพแวดล้อม ต้องมี Docker file สำหรับแต่ละ Service และรองรับการ Deploy บน Kubernetes หรือเทียบเท่า

๓.๓.๕. มีคุณสมบัติการจัดการ Container หลายตัวพร้อมกัน (Orchestration) โดยมีรายละเอียดอย่างน้อยดังนี้

๓.๓.๕.๑. ระบบต้องสามารถจัดการ container หลายตัวพร้อมกันได้ โดยใช้เครื่องมือ Orchestration เช่น Kubernetes

๓.๓.๕.๒. ระบบต้องสามารถเพิ่มหรือลดจำนวน container ของบริการย่อย เช่น user-service ได้อัตโนมัติตามจำนวนผู้ใช้งานที่เพิ่มขึ้นหรือลดลง

๓.๓.๕.๓. ระบบต้องมีการตรวจสอบและติดตามสถานะ active user session เพื่อใช้เป็นข้อมูลประกอบการเพิ่มหรือลดจำนวน container อย่างเหมาะสม

๓.๓.๕.๔. ระบบต้องรองรับการปรับขนาด (horizontal scaling) แบบ dynamic โดยสามารถเพิ่มหรือลดจำนวน instance ของ service ได้อัตโนมัติตามภาระงาน (load) ที่เปลี่ยนแปลง

๓.๓.๕.๕. หากไม่มีผู้ใช้งาน Service ใด Service หนึ่งติดต่อกันนานเกิน ๒ ชั่วโมง หรือระยะเวลาที่ผู้ว่าจ้างกำหนด ระบบต้องสามารถสั่งให้ Service ดังกล่าวเข้าสู่โหมด sleep ได้อัตโนมัติเพื่อคืนทรัพยากร

๓.๓.๕.๖. ระบบต้องรองรับการลดการใช้ทรัพยากรโดยอัตโนมัติ (Downscaling) ผ่านกลไกของ Kubernetes เช่น Horizontal Pod Autoscaler (HPA) หรือ Scale-to-Zero โดยการลดขนาดดังกล่าวต้องไม่กระทบต่อการทำงานของบริการอื่นในระบบ ทั้งนี้ต้องมีการกำหนด ResourceQuota และ PriorityClass อย่างเหมาะสม เพื่อป้องกันไม่ให้การลดขนาดของบริการหนึ่งส่งผลกระทบต่อ SLA ของบริการอื่น

๓.๓.๕.๗. เมื่อมีการร้องขอใช้งานใหม่ ระบบต้องสามารถปรับขนาด (Scale-up) ของบริการที่ถูกลดขนาดจนเหลือศูนย์ (Scale-to-Zero) หรือที่ถูก Downscale ให้อยู่ในระดับที่พร้อมให้บริการได้โดยอัตโนมัติ ภายในเวลาที่สอดคล้องกับ SLA ที่กำหนด ทั้งนี้ หากทรัพยากรของคลัสเตอร์ถูกใช้งานเต็มจนไม่สามารถ Scale-up ได้ ระบบต้องมีกลไกบริหารจัดการทรัพยากร เช่น การปรับลดจำนวน instance ของบริการที่มีความสำคัญรองลงมา โดยการปรับลดดังกล่าวต้องดำเนินการภายใต้เงื่อนไขของ PriorityClass และ Pod Disruption Budget (PDB) เพื่อให้มั่นใจว่าบริการที่สำคัญยังคงสามารถให้บริการได้ตาม SLA ที่กำหนด

๓.๓.๕.๘. ระยะเวลาในการปลุก service และให้พร้อมรับคำสั่งใช้งานได้ต้องไม่เกิน ๑๐ วินาที หรือระยะเวลาที่ผู้ว่าจ้างยอมรับ โดยต้องทำได้โดยอัตโนมัติ

๓.๓.๕.๙. ระบบต้องได้รับการออกแบบให้สามารถทำงานต่อได้แม้เกิดความล้มเหลวของบริการบางส่วน (Partial Failure)

๓.๓.๕.๑๐. ในกรณีที่บริการสำคัญ เช่น user-service ล้มเหลวหรือไม่สามารถตอบสนองได้ ระบบที่เกี่ยวข้องต้องสามารถทำงานต่อไปได้โดยไม่ล้มตาม โดยทำการบันทึกข้อมูลที่จำเป็น เช่น ข้อมูลผู้ใช้งาน ไว้ในระบบ queue ชั่วคราว

๓.๓.๕.๑๑. ระบบ queue ที่ใช้งานต้องมีความน่าเชื่อถือสูง (High Reliability) เช่น RabbitMQ, Apache Kafka หรือเทคโนโลยีอื่นที่มีความสามารถดีกว่า เพื่อรองรับการเก็บข้อมูลและส่งข้อมูลซ้ำ (Retry) ได้อย่างปลอดภัย

๓.๓.๕.๑๒. ระบบต้องมีการออกแบบกระบวนการ Retry และ Dead-letter queue (DLQ) เพื่อให้สามารถจัดการข้อมูลที่ประมวลผลล้มเหลวได้อย่างเหมาะส

๓.๓.๕.๑๓. ระบบต้องมีการ monitor และแจ้งเตือนอัตโนมัติ (Auto Alert) หากเกิดความผิดปกติในกระบวนการ queue หรือเมื่อ Service สำคัญไม่สามารถให้บริการได้ รวมทั้งต้องสามารถแสดงข้อมูลสถานะของแต่ละ Service ได้ เช่น

๑) แสดงให้เห็นว่า service ใดอยู่ในโหมด sleep

๒) แสดงปริมาณการใช้ทรัพยากร (Resource Usage) ของแต่ละ service เช่น CPU, Memory, Inode, Disk I/O และ Network I/O แบบ real-time หรือ near real-time

๓) การขยายขนาด (Scalability) แต่ละ service ย่อยต้องสามารถขยายตัว (Scale) ได้ง่าย เช่น เพิ่ม instance ของ login-service จาก ๒ เป็น ๑๐ ตัวในช่วงที่มีการเข้าสู่ระบบเป็นจำนวนมาก

๔) ความสามารถในการสังเกต (Observability) ระบบต้องสามารถตรวจสอบการทำงานได้ เช่น ติดตั้ง Grafana และ Prometheus เพื่อดูการทำงานของ auth-service แบบ Real-time

๕) ความปลอดภัย (Security) ระบบต้องได้รับการออกแบบโดยคำนึงถึงความปลอดภัยเป็นหลัก (Secure by Design) โดยมีข้อกำหนดอย่างน้อยดังนี้

๕.๑) ข้อมูลผู้ใช้ที่จัดเก็บในระบบ เช่น ข้อมูลส่วนบุคคลและข้อมูลรับรอง ต้องได้รับการเข้ารหัสด้วยมาตรฐาน AES-๒๕๖ หรือที่ดีกว่า

๕.๒) การจัดการคีย์เข้ารหัสต้องดำเนินการผ่านระบบบริหารจัดการคีย์ (Key Management System - KMS) ที่ปลอดภัย เช่น AWS KMS หรือ HashiCorp Vault หรือ KMS แบบ Open Source หรือแบบอื่น ๆ ที่เหมาะสมและได้รับความเห็นชอบจากสำนักงานการตรวจเงินแผ่นดิน

๖) API Gateway เป็นตัวกลางในการควบคุมการรับส่งข้อมูล เช่น Client เรียก /api/users ผ่าน Gateway โดย Gateway จะส่งคำขอไปที่ user-service จริงๆ ผ่าน API Gateway โดยแต่ละ API ต้องสื่อสารผ่าน API Gateway และมีคุณสมบัติอย่างน้อยดังต่อไปนี้

๖.๑) ใช้ HTTPS เพื่อเข้ารหัสข้อมูลระหว่างไคลเอนต์และเซิร์ฟเวอร์ ป้องกันการดักฟัง (eavesdropping) และการโจมตีแบบการแอบดักฟังข้อมูลระหว่างผู้ส่งสารและผู้รับสาร (Man-in-the-Middle - MITM)

๖.๒) ใช้ OAuth2 หรือ JWT เพื่อมอบสิทธิ์ (authorization) ให้แอปพลิเคชันเข้าถึงข้อมูลผู้ใช้โดยไม่ต้องเปิดเผยรหัสผ่าน

คณพศ หงสวรางกูร ปริญญา เพ็ญไพรัตน์กุล ธีรยุทธ วัฒนะธรรมศิริ ศิร์รัฐ ศรีสุนทรพินิต สาลิเดช สัมฤทธิ์
มธุรส ไพบูลย์วรชาติ วิมลมาศ วัฒนจรูญโรจน์ รัฐธรรม ชวนเชย พันธุ์นิชิต โปษยานนท์ นัฐพล สีสวย
วิชาญ จินดารัตน์ สิทธิศักดิ์ ไชยสุข อำนาจวิทย์ หมู่ศิลป์ จุฑามาศ วงศ์เหมอนันต์ ทักษิณ หลังสตา

๖.๓) ใช้ API Key หลายระดับเพื่อจำกัดสิทธิ์ของผู้ใช้ เช่น ระบบหน้าบ้าน (Frontend) ใช้ API Key ที่สามารถอ่านข้อมูล (read-only) ได้เท่านั้น ขณะที่ระบบหลังบ้าน Admin ใช้ Key ที่สามารถสร้างหรือแก้ไขข้อมูล (read-write)

๖.๔) ใช้การกำหนดสิทธิ์ (Authorization) เพื่อควบคุมว่าใครสามารถทำอะไรได้บ้าง เช่น ผู้อำนวยการสามารถดูข้อมูลผู้ใต้บังคับบัญชาได้แต่ไม่สามารถแก้ไขข้อมูลได้

๖.๕) ใช้ Rate Limiting เพื่อจำกัดจำนวนคำขอที่ client สามารถส่งเข้ามาได้ในช่วงเวลาหนึ่ง เช่น จำกัด API สำหรับการค้นหาไม่เกิน ๑๐๐ ครั้ง/นาที/ไอพี เพื่อป้องกันการโจมตีแบบ DDoS

๖.๖) ใช้ API Versioning เพื่อสามารถเปลี่ยนแปลง API ได้โดยไม่กระทบ client เดิม เช่น การเรียก GET /api/v1/users/123 ใช้ข้อมูลแบบโครงสร้างเก่า ในขณะที่ GET /api/v2/users/123 มีการเพิ่มข้อมูลใหม่ เช่น วันเกิดผู้ใช้

๖.๗) ใช้ Allowlist เพื่อจำกัดว่าเฉพาะ IP หรือบัญชีผู้ใช้ที่กำหนดเท่านั้นที่สามารถเข้าถึง API ได้ เช่น อนุญาตเฉพาะ IP ของสำนักงานใหญ่เข้าถึง API สำหรับการแก้ไขฐานข้อมูลพนักงาน

๖.๘) การจัดการข้อผิดพลาด (Error Handling) ระบบต้องออกแบบการจัดการข้อผิดพลาดให้ปลอดภัยและสอดคล้องกับมาตรฐาน HTTP โดยมีข้อกำหนดอย่างน้อยดังนี้

- ใช้รหัสสถานะ HTTP มาตรฐานเพื่อระบุประเภทของข้อผิดพลาด เช่น
  - 400 Bad Request – สำหรับคำขอที่มีข้อมูลไม่ถูกต้องหรือรูปแบบไม่ถูกต้อง (เช่น อีเมลไม่ถูกต้อง)
  - 401 Unauthorized – เมื่อผู้ใช้ไม่ได้รับการยืนยันตัวตน
  - 403 Forbidden – เมื่อผู้ใช้ไม่มีสิทธิ์เข้าถึงทรัพยากร
  - 404 Not Found – เมื่อไม่พบทรัพยากรที่ร้องขอ
  - 500 Internal Server Error – สำหรับข้อผิดพลาดภายในของเซิร์ฟเวอร์
- หลีกเลี่ยงการใช้รหัสสถานะที่ไม่เป็นมาตรฐานหรือกำหนดเอง เพื่อความเข้ากันได้กับเครื่องมือและไคลเอนต์ต่าง ๆ
- ห้ามเปิดเผยรายละเอียดภายในของระบบ เช่น Stack Trace หรือข้อมูลโครงสร้างภายใน เพื่อป้องกันการเปิดเผยข้อมูลที่อาจนำไปสู่ช่องโหว่ด้านความปลอดภัย
- สำหรับข้อผิดพลาดที่ไม่คาดคิดหรือข้อผิดพลาดภายในของเซิร์ฟเวอร์ ควรตอบกลับด้วยรหัสสถานะ ๕๐๐ Internal Server Error และข้อความที่เป็นมิตรต่อผู้ใช้ เช่น "An unexpected error occurred. Please try again later."

คณพศ หงสวรางกูร ปริญญา เพ็ญไพรัตน์กุล ธีรยุทธ วัฒนะธรรมศิริ ศิร์รัฐ ศรีสุนทรพินิต สาลิเดช สัมฤทธิ์
มธุรส ไพบูลย์วรชาติ วิมลมาศ วัฒนจรูญโรจน์ รัฐธรรม ชวนเชย พันธุ์นิชิต โปษยานนท์ นัฐพล สีสวย
วิชาญ จินดารัตน์ สิทธิศักดิ์ ไชยสุข อำนาจวิทย์ หมู่ศิลป์ จุฑามาศ วงศ์เหมอนันต์ ทักษิณ หลังสตา

- บันทึกข้อผิดพลาดภายในระบบ (Log) สำหรับการวิเคราะห์และแก้ไขปัญหาในภายหลัง โดยไม่เปิดเผยข้อมูลดังกล่าวต่อผู้ใช้
- ระบบควรมีการตรวจสอบและแจ้งเตือนเมื่อเกิดข้อผิดพลาดที่สำคัญ เช่น การเชื่อมต่อฐานข้อมูลล้มเหลว หรือบริการภายนอกไม่สามารถเข้าถึงได้ เพื่อให้สามารถดำเนินการแก้ไขได้อย่างรวดเร็ว

๖.๙) ทำ Input Validation เพื่อป้องกันข้อมูลไม่ถูกต้องหรือการโจมตี เช่น เมื่อรับหมายเลขโทรศัพท์ผ่าน API ต้องตรวจสอบว่าเป็นตัวเลขเท่านั้นและมีความยาวไม่เกิน ๑๐ หลัก

๖.๑๐)การแบ่งหน้า (Pagination) เพื่อเพิ่มประสิทธิภาพในการจัดการชุดข้อมูลขนาดใหญ่ โดยการส่งข้อมูลกลับไปยังผู้ใช้แบบไหลต่อเนื่อง (streaming) เพื่อเพิ่มความรวดเร็วในการให้บริการ

๖.๑๑)การบันทึกล็อกแบบอะซิงโครนัส (Asynchronous Logging) เป็นการส่งข้อมูลล็อกไปยังบัฟเฟอร์ที่ไม่ต้องล็อก (lock-free buffer) และคืนการทำงานทันที แทนที่จะเขียนลงดิสก์ทุกครั้ง จากนั้นระบบจะทำการเขียนข้อมูลล็อกลงดิสก์เป็นระยะ ช่วยลดภาระการทำงาน I/O อย่างมีนัยสำคัญ

๖.๑๒)การแคชข้อมูล (Data Caching) ข้อมูลที่มีการเรียกใช้งานบ่อย ๆ จะถูกเก็บไว้ในแคช เพื่อให้สามารถดึงข้อมูลได้รวดเร็วยิ่งขึ้น ลูกค้าจะทำการตรวจสอบแคชก่อนที่จะทำการดึงข้อมูลจากฐานข้อมูล โดยระบบเก็บข้อมูลในหน่วยความจำ (เช่น Redis) ทำให้การเข้าถึงข้อมูลเร็วขึ้นมาก

๖.๑๓)การบีบอัดข้อมูล (Payload Compression) เพื่อให้การรับส่งข้อมูลระหว่างผู้ใช้งานและเซิร์ฟเวอร์เร็วขึ้น จึงมีการบีบอัดข้อมูลทั้งฝั่งการร้องขอและการตอบกลับ เช่น การใช้ gzip ลดเวลาที่ใช้ในการอัปโหลดและดาวน์โหลด

๖.๑๔)การใช้พูลการเชื่อมต่อ (Connection Pooling) เป็นการใช้กลุ่มการเชื่อมต่อฐานข้อมูลที่เปิดค้างไว้ เพื่อหลีกเลี่ยงการเสียเวลาสร้างและปิดการเชื่อมต่อใหม่ทุกครั้งที่ต้องโหลดข้อมูล ระบบจะบริหารจัดการการเชื่อมต่ออย่างมีประสิทธิภาพ

๓.๓.๕.๑๔. ระบบไม่ต้องเก็บข้อมูลสถานะของผู้ใช้ (Stateless Authentication) เช่น session บนฝั่งเซิร์ฟเวอร์ แต่ใช้ JSON Web Token (JWT) เพื่อจัดการการยืนยันตัวตนและการอนุญาต (authentication & authorization) แทนและสามารถกำหนดเวลาหมดอายุของ Token ได้

๓.๓.๕.๑๕. ต้องแยกความเป็นเจ้าของข้อมูล (Data Ownership) โดยมีคุณสมบัติอย่างน้อยดังนี้

๑) แต่ละ Service ย่อยต้องเป็นเจ้าของ Database ของตัวเอง
๒) ห้ามแชร์ Database โดยตรงระหว่าง Services

๓.๓.๕.๑๖. ระบบต้องออกแบบตามแนวคิด Event-Driven Architecture (EDA) โดยให้บริการต่าง ๆ ภายในระบบสามารถสื่อสารกันผ่านการส่งเหตุการณ์ (Event) เช่น เมื่อมีการเปลี่ยนแปลงข้อมูลสำคัญในบริการหนึ่ง ระบบต้องสามารถสร้างและส่งเหตุการณ์ (Event) ไปยังระบบกลาง (Message Broker) เช่น RabbitMQ, Apache Kafka หรือเทคโนโลยีอื่นที่มีความสามารถดีกว่า ที่ทำหน้าที่จัดการการส่ง

คณพศ หงสวรางกูร ปริญญา เพ็ญไพรัตน์กุล ธีรยุทธ วัฒนะธรรมศิริ ศิร์รัฐ ศรีสุนทรพินิต สาลิเดช สัมฤทธิ์
มธุรส ไพบูลย์วรชาติ วิมลมาศ วัฒนจรูญโรจน์ รัฐธรรม ชวนเชย พันธุ์นิชิต โปษยานนท์ นัฐพล สีสวย
วิชาญ จินดารัตน์ สิทธิศักดิ์ ไชยสุข อำนาจวิทย์ หมู่ศิลป์ จุฑามาศ วงศ์เหมอนันต์ ทักษิณ หลังสตา

ต่อเหตุการณ์นี้ไปยังบริการอื่น ๆ ที่เกี่ยวข้องได้ เพื่อให้บริการปลายทางสามารถนำเหตุการณ์ดังกล่าวไปประมวลผลต่อได้ เช่น การแจ้งเตือนผู้ใช้ การอัปเดตข้อมูลเชิงสถิติ หรือกระบวนการอื่น ๆ ที่เกี่ยวข้อง ทั้งนี้ ผู้รับจ้างสามารถเลือกเทคโนโลยีหรือเครื่องมือที่เหมาะสม โดยต้องมั่นใจว่าโครงสร้างงานรองรับการขยายระบบ (Scalability) และการสื่อสารแบบแยกส่วน (Decoupling) ได้อย่างมีประสิทธิภาพ

๓.๓.๕.๑๗. รักษาความสม่ำเสมอของเวอร์ชันและคุณภาพโค้ด (Keep Code at a Similar Level of Maturity) ในแต่ละ Service ย่อย เช่น user-service, auth-service ควรใช้เวอร์ชันของ Framework และไลบรารีที่สอดคล้องกัน โดยมีข้อกำหนดดังนี้

๑) ให้ใช้เวอร์ชันล่าสุดที่เสถียรและอยู่ในระยะ Long Term Support (LTS)

๒) ให้ใช้เฉพาะ ไลบรารีหรือแพ็กเกจที่เป็น Official หรือได้รับการรับรองจากผู้พัฒนา Framework เท่านั้น เช่น Spring Official, Node.js Foundation, Python PyPI Verified

๓) หลีกเลี่ยงการใช้ไลบรารีที่ไม่ได้รับการดูแล หรือมีอัตราการอัปเดตต่ำ (Stale Repository)

๔) ต้องมีระบบตรวจสอบช่องโหว่ของไลบรารี (Dependency Scanning) เช่น OWASP Dependency-Check, Snyk หรือ GitHub Dependabot

๓.๓.๕.๑๘. แยกกระบวนการ Build และ Deploy สำหรับแต่ละ service ย่อย และต้องมี Pipeline สำหรับการ Build และ Deploy ของตนเอง เช่น user-service, notification-service, และ reporting-service ควรมีขั้นตอนการพัฒนาและนำขึ้นระบบที่แยกจากกันอย่างชัดเจน สำหรับ environment dev

๓.๓.๕.๑๙. รองรับการทำ Logging แบบรวมศูนย์ (Centralized Logging) โดยมีคุณสมบัติอย่างน้อยดังนี้

๑) ทุก Service ย่อยต้องส่ง Log ไปยังระบบกลาง เช่น EFK (Elasticsearch, Fluentd, Kibana) หรือที่ดีกว่า

๒) ต้องสามารถติดตาม Request และ Debug ได้แบบ End-to-End

๓) ต้องมีการจัดเก็บ Log แบบเป็นระบบ (Log Archival) พร้อมสามารถเรียกดูย้อนหลังได้ตามระยะเวลาที่กำหนด เช่น ๙๐ วัน

๔) ต้องมีการกำหนดนโยบายการหมุนเวียน Log (Log Rotation) อัตโนมัติ เพื่อควบคุมขนาดพื้นที่จัดเก็บ เช่น รายวัน หรือเมื่อมีขนาดเกินกำหนด

๕) ใช้แนวทางการพัฒนาแอปพลิเคชันที่มีความทนทานต่อความล้มเหลว (Resiliency Patterns) โดยออกแบบและพัฒนาแอปพลิเคชันให้สามารถทำงานต่อได้ แม้จะเกิดปัญหาบางส่วน เช่น ระบบบางส่วนล่มหรือใช้งานไม่ได้ เพื่อให้ผู้ใช้งานยังสามารถใช้บริการได้ต่อเนื่อง ไม่สะดุด และมีคุณสมบัติอย่างน้อยดังนี้ เช่น รองรับ Circuit Breaker, Retry, Fallback ตามลักษณะของแต่ละ Service ผู้รับจ้างต้องออกแบบกระบวนการทำงานแบบ DevSecOps เพื่อให้การพัฒนา ทดสอบ และนำส่งระบบสามารถดำเนินการได้แบบต่อเนื่อง (Continuous) และควบคู่กับการตรวจสอบความปลอดภัย

๓.๔. ผู้รับจ้างต้องออกแบบกระบวนการทำงานแบบ DevSecOps เพื่อให้การพัฒนาทดสอบ และนำส่งระบบสามารถดำเนินการได้แบบต่อเนื่อง (Continuous) และควบคู่กับการตรวจสอบความปลอดภัยโดยต้องนำแนวคิด Shift Left มาใช้ตั้งแต่ระยะเริ่มต้นของวงจรชีวิตการพัฒนา (SDLC) โดยมีคุณสมบัติอย่างน้อยดังนี้

๓.๔.๑. ต้องมีการออกแบบ Workflow หรือ Pipeline ที่ประกอบด้วยขั้นตอนอย่างน้อยดังนี้

๓.๔.๑.๑. ระบบต้องรองรับการจัดการเวอร์ชันของซอร์สโค้ด (Version Control) และการตรวจสอบการเปลี่ยนแปลง โดยมีกระบวนการอย่างน้อย เช่น

๑) ผู้รับจ้างทำการ commit โค้ดไปยังระบบ version control เช่น Git หรือเทียบเท่า โดยแต่ละงานแยกเป็น branch ของตัวเอง จากนั้นให้ผู้ที่มีประสบการณ์ เช่น Senior Developer เป็นผู้ตรวจสอบโค้ด (Code Review) ก่อนทำการ merge เข้าสู่ branch หลัก เช่น master หรือ main เพื่อช่วยควบคุมคุณภาพและลดความผิดพลาด ก่อนส่งงานเข้าสู่ขั้นตอนถัดไปใน pipeline

๒) ผู้รับจ้างต้องจัดการการพัฒนาระบบโดยใช้แนวทาง Git Flow ในการควบคุมเวอร์ชันและการจัดการ branch โดยมีการกำหนดชื่อ branch ตามมาตรฐานเพื่อแยกการทำงานแต่ละประเภทอย่างชัดเจน เพื่อให้สามารถบริหารจัดการโค้ดได้อย่างเป็นระบบ และลดความเสี่ยงจากความผิดพลาด

๓.๔.๑.๒. การวิเคราะห์ความปลอดภัยและคุณภาพของซอร์สโค้ด (Static Code Analysis) ผู้รับจ้างต้องดำเนินการวิเคราะห์ซอร์สโค้ดต้นฉบับ (Source Code) โดยใช้เครื่องมือวิเคราะห์โค้ดแบบสถิต (Static Analysis Tool) ที่สามารถตรวจสอบช่องโหว่ด้านความปลอดภัย ข้อผิดพลาดเชิงตรรกะ และปัญหาคุณภาพของโค้ดได้อย่างครอบคลุม โดยมีข้อกำหนดอย่างน้อยดังนี้

๑) คุณสมบัติของเครื่องมือวิเคราะห์ที่ใช้ต้องมีความสามารถอย่างน้อยดังต่อไปนี้

๑.๑) ตรวจจับช่องโหว่ความปลอดภัย (Security Vulnerabilities) จุดอ่อนของโค้ด (Code Smells) และข้อผิดพลาด (Bugs)

๑.๒) วิเคราะห์ความซ้ำซ้อนของโค้ด (Code Duplication) ความซับซ้อนเชิงตรรกะ (Cognitive Complexity) และ สัดส่วนโค้ดที่ถูกทดสอบโดย Unit Test (Unit Test Coverage)

๑.๓) สร้างรายงานผลการวิเคราะห์ และสามารถตรวจสอบย้อนหลังได้ (Audit Trail)

๑.๔) รองรับภาษาโปรแกรมที่ใช้ในระบบ เช่น Java, JavaScript, Python, C#

๑.๕) สามารถกำหนดเกณฑ์การผ่าน/ไม่ผ่าน (Quality Gate) ได้อย่างชัดเจน

๓.๔.๑.๓. ผู้รับจ้างต้องตรวจสอบไลบรารีและไฟล์ dependencies ที่นำมาใช้ในระบบอย่างสม่ำเสมอ เพื่อตรวจหาช่องโหว่หรือความเสี่ยงด้านความปลอดภัยที่อาจมีอยู่ เช่น การใช้ OWASP Dependency-Check หรือเครื่องมือเทียบเท่า โดยมีเป้าหมายเพื่อป้องกันไม่ให้โค้ดที่มีความเสี่ยงถูกนำไปใช้งานในระบบ ซึ่งอาจส่งผลต่อความปลอดภัยหรือเสถียรภาพของระบบในอนาคต

๓.๔.๑.๔. ผู้รับจ้างต้องทำการ compile และสร้างแอปพลิเคชัน (build) ตามขั้นตอนมาตรฐานที่กำหนด โดยในกระบวนการนี้ต้องมีการนำแนวทางการเขียนโค้ดอย่างปลอดภัย (secure coding practices) มาใช้ เช่น การตรวจสอบความถูกต้องของข้อมูลที่ป้อนเข้ามา (input validation) การเข้ารหัสข้อมูลที่สำคัญ (data encryption) และการจัดการข้อผิดพลาด (error handling) อย่างรัดกุม เพื่อช่วยลดความเสี่ยงด้านความปลอดภัย และป้องกันไม่ให้เกิดปัญหาซ้ำซ้อนหรือกระทบระบบในอนาคต

๓.๔.๑.๕. ผู้รับจ้างต้องจัดให้มีระบบสร้างแอปพลิเคชันแบบอัตโนมัติ (auto build) ซึ่งเชื่อมโยงกับกระบวนการพัฒนา เช่น เมื่อมีการ commit หรือ merge โค้ด ระบบจะทำการ build แอปพลิเคชันโดยอัตโนมัติ ทั้งนี้ pipeline ที่ออกแบบต้องครอบคลุมขั้นตอนตามความต้องการที่ได้เก็บรวบรวมจากผู้ว่าจ้าง (ในขั้นตอน Requirement Gathering)

๓.๔.๑.๖. ผู้รับจ้างต้องจัดให้มีการทดสอบระบบแบบอัตโนมัติ (automated testing) เพื่อยืนยันว่าฟังก์ชันแต่ละส่วนทำงานได้ถูกต้อง และการแก้ไขหรืออัปเดตโค้ดใหม่ไม่ก่อให้เกิดช่องโหว่หรือข้อผิดพลาดใหม่ โดยต้องมีการทดสอบในสองระดับ คือ ตรวจสอบการทำงานของแต่ละฟังก์ชันหรือโมดูลย่อยอย่างละเอียด(Unit Test) ตรวจสอบการทำงานร่วมกันระหว่างโมดูลหรือระบบย่อย(Integration Test) เพื่อให้มั่นใจว่าการเชื่อมต่อระหว่างส่วนต่าง ๆ ของระบบเป็นไปอย่างถูกต้อง

๓.๔.๑.๗. ผู้รับจ้างต้องดำเนินการสแกน Container Image เพื่อหาช่องโหว่ (Vulnerability Scan) ด้วยเครื่องมือที่เหมาะสม เช่น Clair, Trivy หรือเทียบเท่า ก่อนนำไป Deploy

๓.๔.๑.๘. ไม่อนุญาตให้นำ Container Image ที่พบช่องโหว่ระดับรุนแรง (Critical Severity) หรือสูง (High Severity) ไปใช้งานในสภาพแวดล้อมจริงโดยไม่ได้รับการแก้ไขหรืออนุมัติจากผู้ว่าจ้าง

๓.๔.๑.๙. ผู้รับจ้างต้องดำเนินการทดสอบความปลอดภัยของแอปพลิเคชันในระหว่างที่ระบบทำงานจริง (DAST) เพื่อค้นหาช่องโหว่ที่อาจเกิดขึ้นในขั้นตอน Runtime

๓.๔.๑.๑๐. รับจ้างต้องดำเนินการตรวจสอบไฟล์ IaC เช่น Terraform, Ansible, Kubernetes YAML หรือเทียบเท่า เพื่อค้นหาความผิดพลาดในการตั้งค่าที่อาจเป็นจุดเสี่ยงด้านความปลอดภัย (Misconfigurations)

๓.๔.๑.๑๑. ผู้รับจ้างต้องออกแบบให้ Pipeline สำหรับการพัฒนาและนำส่งโค้ด มีขั้นตอนการตรวจสอบความปลอดภัยครบถ้วนในแต่ละขั้นตอนที่เกี่ยวข้อง

๓.๔.๑.๑๒. ต้องกำหนด Gate (จุดตรวจสอบบังคับ) ให้โค้ดและ Artifact ทั้งหมดต้องผ่านการตรวจสอบความปลอดภัยตามเกณฑ์ที่กำหนดก่อนเท่านั้น จึงจะสามารถ Deploy ไปยัง Environment ถัดไปได้

๓.๔.๑.๑๓. ผู้รับจ้างต้องดำเนินการนำโค้ดหรือแอปพลิเคชันขึ้นสู่ Production Environment เฉพาะโค้ดที่ผ่านการตรวจสอบคุณภาพและความปลอดภัยในทุกขั้นตอน และได้รับอนุมัติจากผู้ว่าจ้างแล้วเท่านั้น

๓.๔.๑.๑๔. มีพื้นที่จัดเก็บไฟล์ build (Artifact Repository) ที่มีคุณสมบัติรองรับการรักษาความมั่นคงปลอดภัยของข้อมูล (Security) การควบคุมการเข้าถึง การตรวจสอบย้อนหลัง และสามารถพิสูจน์หรือนำเสนอได้ว่าเป็นระบบที่น่าเชื่อถือ

๓.๔.๑.๑๕. กระบวนการ Deploy ต้องดำเนินการผ่าน Pipeline อัตโนมัติขึ้นสู่ environment ต่างๆ ที่ควบคุมกระบวนการ Deploy อย่างปลอดภัย (Secure Deployment Process) ผู้รับจ้างต้องมีการนำข้อกำหนดการปรับใช้ระบบ (Deployment Strategies) เพื่อให้การนำซอฟต์แวร์ขึ้นระบบ Production เป็นไปอย่างมีประสิทธิภาพ ปลอดภัย และสามารถควบคุมความเสี่ยงได้อย่างเหมาะสม ผู้รับจ้างต้องเสนอและดำเนินการ Deployment Strategy อย่างน้อยหนึ่งรูปแบบต่อไปนี้ ตามลักษณะงานที่เหมาะสม

๑) Blue/Green Deployment ใช้เพื่อเปลี่ยนระบบไปยังเวอร์ชันใหม่ โดยไม่กระทบผู้ใช้งาน และสามารถย้อนกลับไปเวอร์ชันเดิมได้ทันทีหากพบปัญหา

๑.๑) เป็นการเตรียมระบบขึ้น ๒ ชุด คือ ชุดปัจจุบัน (Blue) และชุดใหม่ (Green) โดยไม่กระทบกับผู้ใช้งานระหว่างเตรียมระบบใหม่

- เมื่อระบบใหม่ทดสอบเรียบร้อยแล้ว จะทำการ สลับผู้ใช้งานมายัง Green ทันที
- หากพบปัญหา ต้องสามารถ ย้อนกลับไปใช้ Blue ได้อย่างรวดเร็ว (Rollback)
- Canary Deployment ใช้เพื่อทยอยปล่อยระบบใหม่ให้ผู้ใช้งานทีละกลุ่ม เพื่อตรวจสอบความเสถียรก่อนเปิดให้ผู้ใช้ทั้งหมด
- ปล่อยระบบใหม่ให้ กลุ่มผู้ใช้งานจำนวนน้อย เช่น ๑๐% - ๒๕% ใช้งานก่อน
- ตรวจสอบว่าระบบใหม่ไม่มีปัญหา จึงค่อย ๆ ขยายการใช้งานไปยังผู้ใช้ทั้งหมด
- ต้องสามารถ หยุดและถอยกลับ (Rollback) ได้ทันทีหากเกิดข้อผิดพลาด

๑.๒) A/B Testing ใช้เพื่อทดสอบและเปรียบเทียบประสิทธิภาพของฟีเจอร์หรือการออกแบบระหว่างเวอร์ชัน A และ B กับผู้ใช้งานจริง

- ใช้สำหรับการทดลองฟีเจอร์ใหม่ หรือเปรียบเทียบการออกแบบ (UI/UX) โดย แบ่งกลุ่มผู้ใช้งานเป็นกลุ่ม A และ B
- ต้องสามารถเก็บข้อมูลพฤติกรรมผู้ใช้งานแต่ละกลุ่มเพื่อนำมาวิเคราะห์ก่อนตัดสินใจใช้งานจริง

๑.๓) Feature Flag Deployment ใช้เพื่อควบคุมการเปิด-ปิดฟีเจอร์ใหม่ได้แบบยืดหยุ่น โดยไม่ต้อง deploy โค้ดใหม่ทุกครั้ง

- ฟีเจอร์ใหม่สามารถ เปิด-ปิดได้แบบ Dynamic โดยไม่ต้อง Deploy Code ใหม่
- สามารถเลือกเปิดใช้ฟีเจอร์ใหม่ให้เฉพาะบางกลุ่มผู้ใช้งาน (เช่น ๑๐%-๒๐%) ได้
- ต้องมีระบบบริหารจัดการ Feature Flag ที่ปลอดภัย และต้องลบ Feature Flag ที่ไม่ใช้งานแล้วเสมอ

๑.๔) Rolling Deployment ใช้เพื่ออัปเดตระบบทีละส่วน (Node/Instance) อย่างต่อเนื่อง โดยไม่ทำให้ระบบหยุดให้บริการ

คณพศ หงสวรางกูร ปริญญา เพ็ญไพรัตน์กุล ธีรยุทธ วัฒนะธรรมศิริ ศิร์รัฐ ศรีสุนทรพินิต สาลิเดช สัมฤทธิ์
มธุรส ไพบูลย์วรชาติ วิมลมาศ วัฒนจรูญโรจน์ รัฐธรรม ชวนเชย พันธุ์นิชิต โปษยานนท์ นัฐพล สีสวย
วิชาญ จินดารัตน์ สิทธิศักดิ์ ไชยสุข อำนาจวิทย์ หมู่ศิลป์ จุฑามาศ วงศ์เหมอนันต์ ทักษิณ หลังสตา

- ค่อย ๆ อัปเดตระบบใหม่ ทีละ Node/Instance แบบต่อเนื่อง โดยไม่หยุดให้บริการทั้งหมด
- ต้องสามารถ ตรวจสอบสถานะและ Rollback ได้ทันที หากพบปัญหา

๑.๕) Multi-Service Deployment ใช้เพื่อปรับใช้หลาย Service พร้อมกันอย่างเป็นระบบ โดยคำนึงถึง Dependency ระหว่าง Service ที่เกี่ยวข้อง

- ใช้สำหรับระบบที่ประกอบด้วยหลาย Service ต้อง Deploy พร้อมกันเป็นชุดเดียว
- ต้องจัดทำแผนลำดับการ Deploy อย่างชัดเจน เนื่องจากบาง Service อาจขึ้นอยู่กับ Service อื่น (Dependency)
- ต้องมีแผนสำรองหากบาง Service ล้มเหลว โดยไม่ให้กระทบระบบทั้งหมด

๓.๔.๑.๑๖. ต้องสามารถตรวจสอบประวัติการ Deploy ได้ครบถ้วน (Audit Trail) ระบุอย่างน้อยได้ว่า ใคร Deploy หรือ Deploy อะไร หรือ เวลาใด

๓.๔.๑.๑๗. ผู้รับจ้างต้องดำเนินการติดตั้งและกำหนดค่าระบบ Monitoring และ Logging สำหรับเฝ้าระวังสถานะระบบและเหตุการณ์ความผิดปกติ รวมถึงพฤติกรรมที่อาจเป็นภัยคุกคามทางไซเบอร์ โดยระบบที่นำมาใช้ต้องมีคุณสมบัติอย่างน้อยดังนี้

๑) สามารถรวบรวมและแสดงผลข้อมูล Log และข้อมูล Monitoring จากทุกองค์ประกอบของระบบ (Application, Database, Server, Network)

๒) สามารถกำหนดเกณฑ์แจ้งเตือน (Alert) กรณีพบเหตุการณ์ผิดปกติ เช่น การใช้งานระบบผิดปกติ พฤติกรรมเสี่ยง การพยายามบุกรุกระบบ

๓) สามารถจัดเก็บ Log อย่างปลอดภัยและตรวจสอบย้อนหลังได้ตามนโยบายของผู้ว่าจ้าง

๓.๔.๑.๑๘. ผู้รับจ้างต้องจัดให้มีระบบตอบสนองเหตุการณ์อัตโนมัติในโครงการ โดยต้องมีคุณสมบัติอย่างน้อยดังนี้

๑) สามารถแจ้งเตือน (Alert) ไปยังเจ้าหน้าที่ผู้ดูแลระบบเมื่อเกิดเหตุการณ์ผิดปกติหรือภัยคุกคามที่ตรวจพบ เช่น ผ่านอีเมล, Line, SMS, หรือระบบแจ้งเตือนที่ใช้งานอยู่

๒) สามารถดำเนินการกักกัน (Quarantine) อุปกรณ์หรือบริการที่ตรวจพบความเสี่ยง เพื่อลดความเสียหายที่อาจเกิดขึ้น

๓) สามารถดำเนินการ Rollback ระบบหรือข้อมูลกลับไปยังสถานะที่ปลอดภัยก่อนหน้าอัตโนมัติ หากกำหนดไว้ในแผนงานหรือนโยบาย

๔) ระบบตอบสนองอัตโนมัติต้องกำหนดนโยบายและขอบเขตที่ชัดเจน และผู้รับจ้างต้องทดสอบการทำงานและจัดทำคู่มือการปฏิบัติ (Incident Response Playbook) เพื่อประกอบการตรวจสอบและยืนยันคุณภาพการตอบสนอง

๓.๔.๒. การออกแบบระบบต้องมีกลไกการตรวจสอบความปลอดภัยทั้งก่อนและระหว่างการพัฒนา ตัวอย่างเช่น การตรวจสอบโค้ดอัตโนมัติเมื่อมีการ Commit (โดยข้อความ Commit ต้องเป็นไปตามมาตรฐาน CI/CD) หรือ Merge การตรวจสอบช่องโหว่ของไลบรารีหรือ Dependency ที่ใช้งาน และการตรวจสอบไฟล์ Configuration เช่น YAML/JSON เพื่อป้องกันการตั้งค่าที่ไม่ถูกต้อง

๓.๔.๓. Workflow ที่ออกแบบต้องสามารถทำงานร่วมกับระบบควบคุมเวอร์ชันของ Source Code เช่น Git หรือเทียบเท่า และสามารถเชื่อมโยงกับระบบทดสอบแบบอัตโนมัติ เพื่อให้เกิดการ Deploy ระบบอย่างต่อเนื่องโดยไม่ต้องทำด้วยมือ

๓.๔.๔. ต้องสามารถตั้งค่าตัวกรอง (Gate) เพื่อไม่ให้โค้ดที่มีความเสี่ยงร้ายแรง (Critical Vulnerability) ถูกนำไป Deploy โดยไม่ได้รับการตรวจสอบ

๓.๔.๕. ต้องสามารถตรวจสอบและบันทึกผลการทำงานของ Pipeline แต่ละรอบได้ เช่น บันทึกว่าใคร Deploy อะไร เมื่อใด และ ผลการทดสอบเป็นอย่างไร

๓.๕. ผู้รับจ้างต้องวางแผนและดำเนินการด้านความปลอดภัยสำหรับการใช้งาน Container Technology เช่น Docker หรือเทียบเท่า ให้เป็นส่วนหนึ่งของกระบวนการ DevSecOps โดยต้องครอบคลุมประเด็นด้านความปลอดภัย ดังต่อไปนี้

๓.๕.๑. ต้องมีขั้นตอนการตรวจสอบความปลอดภัยของ Container Image ที่ใช้ทั้งในขั้นตอนการพัฒนาและการนำขึ้นใช้งานจริง

๓.๕.๒. ต้องใช้ Container Image ที่มาจากแหล่งที่เชื่อถือได้ (Trusted Registry) เท่านั้น

๓.๕.๓. ต้องมีการสแกน Container Image เพื่อหาช่องโหว่ (Vulnerability Scan) ก่อนนำไปใช้งาน เช่น Common Vulnerabilities and Exposures (CVE), Misconfiguration

๓.๕.๔. ผู้รับจ้างต้องจัดทำ Dockerfile หรือ docker-compose.yml ให้สอดคล้องกับแนวทางด้านความปลอดภัยที่เหมาะสม โดยมีรายละเอียดอย่างน้อยดังต่อไปนี้

๓.๕.๔.๑. รัน Container ด้วยสิทธิ์ที่น้อยที่สุด โดย Container ต้องไม่รันด้วย root โดยตรง ให้สร้างและใช้ ผู้ใช้ (User) ใหม่ใน container เพื่อลดความเสี่ยงด้านความปลอดภัย

๓.๕.๔.๒. ใช้ COPY แทน ADD เมื่อไม่จำเป็น

๓.๕.๔.๓. ลดจำนวน Layer โดยรวมคำสั่งหลาย ๆ คำสั่งเข้าด้วยกัน

๓.๕.๔.๔. จัดเรียงคำสั่งใน Dockerfile ให้เหมาะสมกับการ Cache เช่น คำสั่งที่เปลี่ยนบ่อยให้อยู่ล่างสุด เพื่อให้ Docker สามารถใช้ layer cache ได้อย่างมีประสิทธิภาพ

๓.๕.๔.๕. ใช้ Official Image จาก Docker Hub เพื่อความปลอดภัยและการอัปเดตอย่างสม่ำเสมอ ให้เลือกใช้งาน Image ที่เป็นทางการ (Official) เท่านั้น เช่น node:๑๔-alpine กำหนดเวอร์ชันของ Image อย่างชัดเจน ห้ามใช้ latest tag ให้ระบุเวอร์ชันที่แน่นอน เช่น node:๑๔.๑๗.๐-alpine๓.๑๓ เพื่อความสามารถในการ reproduce และลดความเสี่ยงจากการเปลี่ยนแปลงที่ไม่คาดคิด

๓.๕.๔.๖. ใช้ Multi-Stage Build ผู้รับจ้างต้องใช้เทคนิค Multi-Stage Build ในการสร้าง Docker Image โดยแยกขั้นตอนการ Build เช่น Compile, Build Library ออกจากขั้นตอนสำหรับ Production เพื่อลดขนาดของ Image ที่นำไปใช้งานจริง และลดไฟล์หรือ Package ที่ไม่จำเป็นในขั้นตอนการทำงานจริง

๓.๕.๔.๗. ต้องกำหนดไฟล์ .dockerignore เพื่อยกเว้นไฟล์หรือโฟลเดอร์ที่ไม่จำเป็นไม่ให้ถูกเพิ่มเข้าไปใน Image เช่น node_modules/, npm-debug.log หรือไฟล์ที่เกี่ยวข้องกับ Local Environment เพื่อป้องกันขนาด Image โตโดยไม่จำเป็น

๓.๕.๔.๘. ต้องกำหนดค่าพารามิเตอร์ เช่น Path, Token, URL ผ่าน Environment Variables แทนการเขียนค่าคงที่ (Hardcode) ภายใน Image เพื่อให้สามารถเปลี่ยนแปลงค่าต่าง ๆ ได้ง่ายเมื่อนำไปใช้ในแต่ละสภาพแวดล้อม (Environment)

๓.๕.๔.๙. ต้องกำหนด Label ให้กับ Image เช่น maintainer, version, description เพื่อช่วยให้สามารถบริหารจัดการ Image ได้ง่ายขึ้นในระยะยาว และช่วยติดตามข้อมูลสำคัญเกี่ยวกับ Image

๓.๕.๔.๑๐. สแกนหาช่องโหว่ใน Image ก่อนใช้งานจริงก่อนนำ Docker Image ไปใช้งานในระบบจริง ผู้รับจ้างต้องสแกนหาช่องโหว่ (Vulnerability Scan) ด้วยเครื่องมือที่เป็นที่ยอมรับ เช่น docker scan, Trivy หรือเครื่องมืออื่นที่เทียบเท่า และต้องแนบรายงานผลการสแกนเพื่อประกอบการตรวจสอบ

๓.๕.๔.๑๑. ต้องปฏิบัติตามแนวทางการตั้งค่าความปลอดภัยอย่างน้อยระดับ Level ๑ ของ CIS Docker Benchmark โดยมีข้อกำหนดหลักที่ต้องปฏิบัติ เช่น

๑) ต้องกำหนดขีดจำกัดการใช้ CPU และ Memory ให้กับแต่ละ Container อย่างเหมาะสม เพื่อลดความเสี่ยงที่ Container ใด ๆ จะใช้ทรัพยากรเกินความจำเป็นและส่งผลกระทบต่อระบบโดยรวม

๒) ต้องกำหนดให้ Container ทุกตัวส่ง Log ออกไปในรูปแบบที่เหมาะสม เช่น JSON และต้องเชื่อมต่อกับระบบ Logging กลาง เช่น EFK Stack หรือเทียบเท่า เพื่อให้สามารถเก็บ Log ตรวจสอบย้อนหลัง และทำ Audit Trail ได้

๓) ห้าม Mount Directory ที่อาจถูกนำไปใช้โจมตีหรือดัดแปลง Docker Engine โดยเด็ดขาด เช่น /var/run/docker.sock หรือ Directory System อื่น ๆ ที่สำคัญ

๓.๕.๕. ห้ามฝัง Secrets, Token, User, Password ไว้ใน Dockerfile หรือ Image

๓.๕.๖. ต้องใช้วิธีการจัดเก็บ Secrets อย่างปลอดภัย เช่นผ่าน Secret Manager หรือ Environment Variables ที่แยกจาก Source Code

๓.๕.๗. ต้องมีการจำกัดสิทธิ์ของ Container ที่รันอยู่ เช่น ตั้งค่า read-only filesystem การปิดการใช้งานการยกระดับสิทธิ์(privilege escalation)

๓.๕.๘. ต้องมีแนวทางป้องกัน Container Escape เช่น เปิดใช้งาน AppArmor, SELinux หรือ Seccomp Profiles (ถ้าหากระบบนั้นรองรับ)

๓.๕.๙. ต้องสามารถระบุได้ว่า Container ใดกำลังทำงาน อ้างอิงกับเวอร์ชันที่ผ่านการทดสอบแล้ว

๓.๕.๑๐. Container ต้องถูกตรวจสอบสถานะและจัดเก็บ Log อย่างเหมาะสม เพื่อให้สามารถติดตามพฤติกรรมผิดปกติได้

๓.๕.๑๑. Log ต้องสามารถเชื่อมโยงกับระบบ Monitoring ของ DevSecOps (เช่น Alert เมื่อ Container ล่มหรือทำงานผิดปกติ)

๓.๕.๑๒. ขั้นตอน Build Container Image ต้องผสานกับ Pipeline ของ DevSecOps โดยทำการตรวจสอบสุขภาพความปลอดภัยอัตโนมัติก่อน Deploy

๓.๕.๑๓. ระบบต้องสามารถระงับการ Deploy Image ที่ไม่ผ่านเกณฑ์ความปลอดภัย (เช่น ถ้าพบช่องโหว่ระดับวิกฤติ)

๓.๕.๑๔. ต้องจัดทำรายงานสรุปผลการตรวจสอบ Container Security ที่ดำเนินการในแต่ละรอบ เช่น รายงานการสแกน Image การตั้งค่าที่ใช้ การป้องกันการรันด้วย Root และ ต้องสามารถตรวจสอบย้อนหลังได้ผู้รับจ้างต้องดำเนินการจัดเก็บ Source Code สคริปต์สำหรับจัดการฐานข้อมูล และไฟล์คอนฟิกทั้งหมดไว้ในระบบควบคุมเวอร์ชันที่รองรับการทำงานแบบทีม สามารถตรวจสอบย้อนหลัง (Audit) ได้ และสามารถบูรณาการกับกระบวนการพัฒนาแบบต่อเนื่อง (CI/CD Pipeline) ได้อย่างเหมาะสม

๓.๖. ผู้รับจ้างต้องออกแบบการตรวจสอบความปลอดภัยของซอร์สโค้ด (Secure Code Review) โดยใช้เครื่องมือหรือวิธีการที่สามารถ ดังนี้

๓.๖.๑. ทำการวิเคราะห์โค้ดต้นฉบับโดยอัตโนมัติ (Automated Static Code Analysis) ได้ทั้งสำหรับภาษาโปรแกรมที่ใช้ในการพัฒนา เช่น JavaScript, Python หรือภาษาอื่นที่เกี่ยวข้อง

๓.๖.๒. ตรวจจับจุดอ่อนด้านความปลอดภัย (Security Vulnerabilities) ได้ครอบคลุมตามแนวทาง OWASP Top ๑๐ เช่น SQL Injection, Cross-Site Scripting (XSS), Broken Authentication และอื่น ๆ

๓.๖.๓. แสดงระดับความรุนแรงของปัญหา (Severity Levels) และสามารถจัดลำดับความสำคัญเพื่อแก้ไขได้

๓.๖.๔. วิเคราะห์คุณภาพโค้ด (Code Quality) เช่น ความซับซ้อนของโค้ด (Code Complexity), Code Duplication และ Code Smell

๓.๖.๕. สร้างรายงานอัตโนมัติ (Automated Reports) ที่สามารถส่งมอบได้ในรูปแบบ PDF หรือ HTML โดยระบุปัญหาพร้อมคำแนะนำในการปรับปรุง

๓.๖.๖. รองรับการทำงานแบบต่อเนื่อง (Continuous Integration) กับระบบพัฒนา เช่น GitLab หรือเทียบเท่า เพื่อให้สามารถทำ Secure Code Review ทุกครั้งที่มีการ Commit หรือ Merge โค้ด

๓.๖.๗. รองรับการตรวจสอบ Source Code ทั้งฝั่ง Frontend และ Backend รวมถึงโค้ดที่เกี่ยวข้องกับการเชื่อมต่อ API หรือฐานข้อมูล

๓.๖.๘. เครื่องมือหรือแนวทางที่เลือกใช้จะต้องเป็นที่ยอมรับในอุตสาหกรรม และผลการวิเคราะห์ต้องสามารถตรวจสอบย้อนหลังได้ เพื่อประกอบการประเมินผลและรับรองคุณภาพของระบบ

๓.๗. ผู้รับจ้างต้องดำเนินการจัดเก็บ Source Code สคริปต์สำหรับจัดการฐานข้อมูล และไฟล์คอนฟิกทั้งหมดไว้ในระบบควบคุมเวอร์ชันที่รองรับการทำงานแบบทีม สามารถตรวจสอบย้อนหลัง (Audit) ได้ และสามารถบูรณาการกับกระบวนการพัฒนาแบบต่อเนื่อง (CI/CD Pipeline) ได้อย่างเหมาะสม โดยมีคุณสมบัติอย่างน้อยดังนี้

๓.๗.๑. รองรับการทำงานแบบ Branch/Merge

๓.๗.๒. จัดเก็บประวัติการเปลี่ยนแปลงของโค้ดและฐานข้อมูลแบบละเอียด

๓.๗.๓. มีระบบตรวจสอบความแตกต่างของเวอร์ชัน (Diff)

๓.๗.๔. รองรับการ Rollback กรณี Migration ผิดพลาด

๓.๗.๕. รองรับการแยก Environment สำหรับการ Deploy

๓.๗.๖. มีระบบกำหนดสิทธิ์การเข้าถึงของผู้พัฒนา

๓.๗.๗. บันทึกเหตุการณ์หรือ Log การเปลี่ยนแปลงที่สำคัญ

๓.๘. ผู้รับจ้างต้องออกแบบให้สามารถจัดการข้อมูลแบบ Cache เพื่อเพิ่มประสิทธิภาพในการเข้าถึงข้อมูล โดยต้องรองรับการทำงานต่อเนื่อง (High Availability) และสามารถจัดการกับการทำงานที่มีปริมาณสูงได้

๓.๙. ผู้รับจ้างต้องออกแบบให้รองรับการเข้ารหัสข้อมูลตามมาตรฐานความปลอดภัยระดับสากล เช่น TLS 1.2 หรือสูงกว่า และต้องรองรับการพิสูจน์ตัวตนแบบ Token-Based เพื่อป้องกันการโจมตีและการเข้าถึงโดยไม่ได้รับอนุญาต

๓.๑๐. ผู้รับจ้างต้องออกแบบการบริหารการจัดการสิทธิ์รวมถึงจัดทำตารางกำหนดสิทธิของผู้ใช้งาน (User Authorization Matrix) ในแต่ละ ระบบงานสามารถนำออก (Export) มาเป็นรูปแบบไฟล์ PDF และรูปแบบไฟล์อิเล็กทรอนิกส์อื่น ๆ ที่เหมาะสม เช่น Word, Excel, CSV, JSON และ Text File เป็นต้น และ Print ได้ และสามารถเลือกนำออก ข้อมูลดังกล่าวเป็นความถี่รายเดือน รายปี หรือตามที่สำนักงานการตรวจเงินแผ่นดินกำหนดได้

๓.๑๑. ผู้รับจ้างต้องออกแบบและจัดทำโครงสร้างพื้นฐานของระบบโดยใช้แนวทาง Infrastructure as Code (IaC) เพื่อให้สามารถสร้าง ติดตั้ง และปรับขนาด (Provision & Scale) สภาพแวดล้อมได้โดยอัตโนมัติ และสามารถควบคุมเวอร์ชันได้อย่างมีประสิทธิภาพ โดยโครงสร้างพื้นฐานที่พัฒนาด้วย IaC ต้องสามารถสร้างระบบในรูปแบบ Cluster ได้เป็นอย่างน้อย

๓.๑๒. ผู้รับจ้างต้องออกแบบระบบให้สามารถเชื่อมต่อและทำงานร่วมกับเครื่องมือสำหรับตรวจสอบประสิทธิภาพ (Performance Monitoring) และติดตามการทำงานของระบบใน Environment UAT และ Production โดยเครื่องมือที่เลือกใช้ต้องมีคุณสมบัติดังนี้

๓.๑๒.๑. ระบบต้องสามารถรองรับการติดตามการทำงานแบบกระจาย (Distributed Tracing) โดยมีคุณสมบัติอย่างน้อยดังนี้

๓.๑๒.๑.๑. สามารถตรวจสอบและติดตามเส้นทางการทำงานของแต่ละ Request/Transaction ที่ผ่านระบบ Microservices หรือส่วนประกอบต่าง ๆ ได้

๓.๑๒.๑.๒. สามารถระบุคอขวดหรือจุดที่เกิดความหน่วง (Latency) แบบ End-to-End ได้

๓.๑๒.๒. รองรับ Real-time Monitoring บน Environment UAT และ Production โดยมีคุณสมบัติอย่างน้อยดังนี้

๓.๑๒.๒.๑. สามารถตรวจสอบสถานะต่างๆ ได้แบบเรียลไทม์ โดยมีรายละเอียดอย่างน้อยดังนี้

๑) สถานะ ของ Service/Component ต่าง ๆ

๒) สถานะ CPU

๓) สถานะ Memory

๔) สถานะ Disk แต่ละ Partition

๓.๑๒.๒.๒. สามารถแจ้งเตือนเมื่อค่าประสิทธิภาพเกินเกณฑ์ที่กำหนด เช่น Response Time, Error Rate หรือ Resource Utilization

๓.๑๒.๓. รองรับการทำงานร่วมกับเครื่องมือมาตรฐานอุตสาหกรรม โดยมีคุณสมบัติอย่างน้อยดังนี้

๓.๑๒.๓.๑. ต้องสามารถเชื่อมโยงหรือส่งข้อมูลให้กับระบบ Trace/Monitoring ที่เป็นมาตรฐาน เช่น Jaeger, Dynatrace หรือเทียบเท่าได้

๓.๑๒.๓.๒. ไม่จำกัดยี่ห้อหรือผลิตภัณฑ์ เพื่อเปิดโอกาสให้สามารถเลือกใช้ Open Source หรือ Commercial Solution ได้ตามความเหมาะสม

๓.๑๒.๔. รองรับการบันทึกและตรวจสอบข้อมูลย้อนหลัง (Historical Analysis) โดยมีคุณสมบัติอย่างน้อยดังนี้

๓.๑๒.๔.๑. ผลการมอนิเตอร์ต้องสามารถบันทึกและตรวจสอบย้อนหลังได้

๓.๑๒.๔.๒. ต้องสามารถ Export ข้อมูลหรือเชื่อมต่อกับ Dashboard เช่น Grafana, Kibana หรือเทียบเท่า เพื่อให้ผู้ว่าจ้างสามารถเข้าถึงข้อมูลได้

๓.๑๒.๕. ผู้รับจ้างต้องออกแบบและกำหนดการเข้าถึงระบบไฟล์และโฟลเดอร์ โดยมีการแบ่งเส้นทางการใช้งานและกำหนดกลุ่มผู้ใช้งานให้เหมาะสมกับหน้าที่ความรับผิดชอบและความมั่นคงปลอดภัยของระบบ

๓.๑๒.๖. ผู้รับจ้างต้องออกแบบการวางสิทธิ์และบทบาทผู้ใช้ในระบบตาม Role Metric ที่ได้จัดทำในขั้นตอน Requirement Gathering โดยต้องมีรายละเอียดอย่างน้อยดังนี้

๓.๑๒.๗. การออกแบบโครงสร้างบทบาท (Role Structure) เช่น Role-based Access Control (RBAC)

๓.๑๒.๘. การจัดกลุ่มสิทธิ์ (Permission Grouping) เช่น กลุ่มอ่าน กลุ่มเขียน กลุ่มอนุมัติ

๓.๑๒.๙. การออกแบบการ Mapping ระหว่างบทบาทกับกลุ่มสิทธิ์ เช่น Creator ⟶ กลุ่มเขียน + กลุ่มอนุมัติ

๓.๑๒.๑๐. การระบุเส้นทางหรือโซนของระบบที่แต่ละบทบาทสามารถเข้าถึง เช่น Content Repository, Approval Dashboard, Public Portal

๓.๑๒.๑๑. การแนบ Diagram หรือแผนผัง Role-Permission Mapping (เช่น Unified Modeling Language: UML หรือ Diagram ชัดเจนอื่น ๆ) ที่อธิบายการเชื่อมโยงระหว่างผู้ใช้ บทบาท สิทธิ์ และส่วนประกอบของระบบ

๓.๑๒.๑๒. การออกแบบทั้งหมดนี้ต้องเป็นไปตามมาตรฐานความมั่นคงปลอดภัยที่เป็นที่ยอมรับ เช่น หลักสิทธิ์น้อยที่สุด(Least Privilege) หลักการแยกหน้าที่ (Separation of Duties) และต้องสามารถตรวจสอบย้อนหลัง (Audit) ได้

๓.๑๓. ผู้รับจ้างต้องออกแบบและกำหนดการเข้าถึงระบบไฟล์และโฟลเดอร์ โดยมีการแบ่งเส้นทางการใช้งานและกำหนดกลุ่มผู้ใช้งานให้เหมาะสมกับหน้าที่ความรับผิดชอบและความมั่นคงปลอดภัยของระบบ ซึ่งอย่างน้อยต้องมีรายละเอียดดังนี้

๓.๑๓.๑. การทำ Mapping กลุ่มผู้ใช้และโฟลเดอร์ จัดทำ Mapping ระหว่างกลุ่มผู้ใช้ (User Group) กับโฟลเดอร์ที่เกี่ยวข้อง เช่น

๓.๑๓.๑.๑. กลุ่ม Jenkins Admin ⟶ ดูแล /jenkins

๓.๑๓.๑.๒. กลุ่ม Apache Admin ⟶ ดูแล /apache_log, /apache_dump, /apache_archive

๓.๑๓.๑.๓. กลุ่ม Elastic Admin ⟶ ดูแล /elasticdata๑, /elasticdata๒

๓.๑๓.๒. การกำหนดสิทธิ์การเข้าถึง กำหนดสิทธิ์ที่เหมาะสม เช่น read,write, execute ตามบทบาทหน้าที่

๓.๑๓.๓. การจัดเตรียม Script หรือแนวทางสร้าง Group และกำหนดสิทธิ์จัดเตรียม Script เช่น Shell Script, Ansible Playbook หรือเทียบเท่า เพื่อให้สามารถปรับปรุงซ้ำได้ในอนาคต (Infrastructure as Code)

๓.๑๓.๔. การออกแบบทั้งหมดนี้ต้องเป็นไปตามมาตรฐานความมั่นคงปลอดภัยที่เป็นที่ยอมรับ เช่น หลักสิทธิ์น้อยที่สุด(Least Privilege) หลักการแยกหน้าที่ (Separation of Duties) และต้องสามารถตรวจสอบย้อนหลัง (Audit) ได้

๓.๑๔. การออกแบบและจัดการซอร์สโค้ดและ Micro Frontend ผู้รับจ้างต้องออกแบบโครงสร้างคลังเก็บซอร์สโค้ด (Source Code Repositories) และการพัฒนาระบบ Frontend ตามหลักการดังต่อไปนี้

๓.๑๔.๑. การแยกส่วนของซอร์สโค้ดและโดเมนเนม ต้องแยกซอร์สโค้ดของส่วนหน้าบ้าน (Frontend สำหรับผู้ใช้งานทั่วไป) และส่วนหลังบ้าน (Frontend สำหรับผู้ดูแลระบบ - Admin) ออกจากกันอย่างชัดเจน และต้องแยกโดเมนเนม (DNS) สำหรับแต่ละส่วนให้เป็นอิสระจากกัน เช่น portal.example.com สำหรับผู้ใช้งานทั่วไป และ admin.example.com สำหรับผู้ดูแลระบบ

๓.๑๔.๒. การใช้แนวคิด Micro Frontend โดย Frontend ต้องถูกแยกออกเป็นโมดูล (Modules) เช่นเดียวกับการออกแบบ Microservices โดยแต่ละโมดูลต้องสามารถพัฒนา ทดสอบ และ Deploy ได้อย่างอิสระ

๓.๑๕. การออกแบบและพัฒนาเว็บไซต์ให้มีประสิทธิภาพด้าน Frontend (Frontend Performance Optimization) ผู้รับจ้างต้องออกแบบและพัฒนาเว็บไซต์ให้สามารถโหลดและแสดงผลได้อย่างรวดเร็ว มีประสิทธิภาพสูงสุดต่อผู้ใช้งาน โดยต้องพิจารณาใช้แนวทางการเพิ่มประสิทธิภาพ (Performance Optimization Techniques) อย่างน้อยดังต่อไปนี้

๓.๑๕.๑. Compression

๓.๑๕.๑.๑. บีบอัดไฟล์ (HTML, CSS, JS, SVG) ก่อนส่งผ่านเครือข่ายด้วย Gzip หรือ Brotli

๓.๑๕.๑.๒. ลดขนาด Payload และ Network Latency

๓.๑๕.๒. Selective Rendering / Windowing

๓.๑๕.๒.๑. แสดงเฉพาะองค์ประกอบที่อยู่ในพื้นที่มองเห็น (Above the Fold)

๓.๑๕.๒.๒. ใช้ Virtual Scrolling หรือ Lazy Rendering สำหรับรายการที่มีจำนวนมาก ตามความเหมาะของเนื้อหา (Content) ในแต่ละหน้าจอ

๓.๑๕.๓. Modular Architecture & Code Splitting

๓.๑๕.๓.๑. แยก JavaScript ออกเป็นโมดูลย่อยเพื่อให้โหลดเฉพาะส่วนที่จำเป็น เช่น Webpack SplitChunks หรือ ES Modules

๓.๑๕.๓.๒. ลดขนาด Initial JS Bundle

๓.๑๕.๔. Priority-Based Loading

๓.๑๕.๔.๑. โหลดเฉพาะทรัพยากรสำคัญก่อน เช่น HTML, Critical CSS, JS ที่จำเป็น

๓.๑๕.๔.๒. ใช้เทคนิค อย่างน้อยดังนี้

๑) defer เพื่อโหลด script พร้อมกัน รันทีหลัง HTML

๒) async เพื่อโหลดและรันทันที (ถ้าโหลดเสร็จ)

๓) preload เพื่อบอก browser ให้โหลด resource สำคัญล่วงหน้า

๔) resource hints เพื่อบอก browser ให้เตรียมการเชื่อมต่อหรือลิงก์ล่วงหน้า

๓.๑๕.๕. Dynamic Imports

๓.๑๕.๕.๑. โหลดโมดูลแบบ Lazy ตามการใช้งานจริง เช่น import('module')

๓.๑๕.๕.๒. ลดเวลาโหลดหน้าแรก (TTI – Time to Interactive)

๓.๑๕.๖. Tree Shaking / Dead Code Removal

๓.๑๕.๖.๑. ตัดโค้ดที่ไม่ถูกใช้งานออกจาก Final JS Bundle

๓.๑๕.๖.๒. ใช้กับ Bundler เช่น Webpack, Rollup

๓.๑๕.๗. Pre-Fetching

๓.๑๕.๗.๑. ดึงข้อมูลล่วงหน้าก่อนที่ผู้ใช้งานจะร้องขอจริง เช่น ข้อมูลในหน้า Next Page

๓.๑๕.๗.๒. ใช้ <link rel="prefetch"> หรือ Browser Cache API

๓.๑๕.๘. Pre-loading / Preconnect / DNS-Prefetch

๓.๑๕.๘.๑. ให้ Browser เตรียมตัวโหลดไฟล์ที่จำเป็นได้เร็วขึ้นสำหรับ CDN, Fonts และ API Endpoint

๓.๑๖. กรณีที่ผู้รับจ้างมีการพัฒนาการประมวลผลชุดข้อมูล (Batch Processing) โดยต้องมีการออกแบบอย่างน้อยดังนี้

๓.๑๖.๑. ผู้รับจ้างต้องออกแบบและพัฒนาระบบประมวลผลชุดข้อมูล (Batch Job) ให้สามารถดำเนินการได้อย่างมีเสถียรภาพ มีความยืดหยุ่นต่อปริมาณข้อมูลที่เปลี่ยนแปลง และรองรับการประมวลผลขนาดใหญ่ได้อย่างมีประสิทธิภาพ

๓.๑๖.๒. ระบบ Batch ต้องสามารถทำงานแบบอัตโนมัติ (Automated Scheduling) โดยรองรับการตั้งเวลาแบบกำหนดเองได้ (Flexible Scheduling) และรองรับการทำงานซ้ำอัตโนมัติกรณีล้มเหลว (Automatic Retry with Configurable Policy)

๓.๑๖.๓. ผู้รับจ้างต้องออกแบบการทำงานของ Batch ให้รองรับการยืนยันตัวตนและกำหนดสิทธิ์ผู้ใช้งาน (Authentication and Authorization) อย่างเหมาะสม

๓.๑๖.๔. ต้องมีการเข้ารหัสข้อมูลที่ใช้ระหว่างการประมวลผล ทั้งข้อมูลที่อยู่ในระหว่างการส่ง (In-Transit) และข้อมูลที่จัดเก็บ (At-Rest) ตามมาตรฐานความปลอดภัยสากล เช่น TLS ๑.๒ หรือสูงกว่า และ AES-๒๕๖

๓.๑๖.๕. ต้องมีการตรวจสอบและบันทึกการทำงาน (Audit Logging) ของ Batch แต่ละรอบ เพื่อให้สามารถตรวจสอบย้อนหลังได้ เช่น เวลาเริ่มต้น เวลาสิ้นสุด สถานะสำเร็จ/ล้มเหลว และข้อผิดพลาด (Error Details)

๓.๑๖.๖. ระบบต้องสามารถรองรับกรณีการล่ม (Failure) ได้โดยต้องมีฟังก์ชันดังต่อไปนี้

๓.๑๖.๖.๑. รองรับการทำ Retry และ Recovery อัตโนมัติ

๓.๑๖.๖.๒. สามารถทำ Resume/Restart ได้จากจุดที่ล้มเหลว (Checkpoint / Fault-Tolerant Mechanism)

๓.๑๖.๖.๓. กรณีผิดพลาดร้ายแรง ต้องสามารถสั่ง Rollback หรือทำ Data Recovery ได้อย่างปลอดภัย

๓.๑๖.๗. ต้องออกแบบ Batch ให้สามารถทำงานแบบ Parallel หรือ Distributed ได้ (หากลักษณะงานเอื้ออำนวย) เพื่อเพิ่มความเร็วและลดความเสี่ยงจุดล้มเหลวเดียว (Single Point of Failure)

๓.๑๖.๘. ต้องมีระบบแจ้งเตือนอัตโนมัติ (Alert) เมื่อเกิดความล้มเหลวหรือผิดปกติในการประมวลผล เช่น ผ่าน Email, Webhook หรือ Integration กับ Monitoring Tools เช่น Grafana, Zabbix, Prometheus หรือเทียบเท่า

๓.๑๖.๙. ผู้รับจ้างต้องจัดทำรายงานสรุปผลการทำงานของ Batch ประจำรอบ (Batch Job Summary Report) และต้องสามารถนำเสนอข้อมูลเชิงวิเคราะห์ได้ เช่น Job Success Rate, Average Processing Time, Bottleneck Analysis เป็นต้น

๓.๑๖.๑๐. ต้องพัฒนา Batch ตามหลัก Secure Coding เพื่อป้องกันความเสี่ยงด้านความปลอดภัย เช่น Input Validation, Exception Handling และ Resource Management

๓.๑๖.๑๑. ผู้รับจ้างต้องทดสอบการทำงานของ Batch ทั้งในสถานการณ์ปกติและสถานการณ์ผิดพลาด (Normal and Exception Scenarios) พร้อมจัดทำเอกสารผลการทดสอบ (Test Report)

๓.๑๖.๑๒. การพัฒนาระบบ Batch ต้องสามารถบูรณาการเข้ากับกระบวนการ DevSecOps ได้ เช่น รองรับการตรวจสอบช่องโหว่ก่อน Deployment

๓.๑๖.๑๓. ระบบ Batch ต้องสามารถขยายตัวได้ในอนาคต (Scalable Design) เช่น การปรับจำนวน Process, Thread หรือ Node ตามปริมาณข้อมูลที่เพิ่มขึ้น

๓.๑๖.๑๔. Source Code ของระบบ Batch ทั้งหมด ต้องจัดเก็บในระบบ Version Control และสามารถตรวจสอบย้อนหลังได้ เช่น Git หรือเทียบเท่า

๓.๑๗. รองรับการใช้งาน WebSocket สำหรับงานที่ต้องการการเชื่อมต่อระยะยาว ในกรณีที่ระบบมีความจำเป็นต้องรองรับการประมวลผลแบบต่อเนื่อง เช่น การอัปโหลดไฟล์ขนาดใหญ่ หรือการประมวลผลข้อมูลจำนวนมาก (Bulk Processing) ผู้รับจ้างต้องออกแบบให้สามารถรองรับการเชื่อมต่อผ่าน WebSocket ได้อย่างมีประสิทธิภาพ โดยผู้รับจ้างต้องดำเนินการออกแบบและกำหนดค่าการเชื่อมต่อ WebSocket ทั้งในฝั่ง Client และ Server ให้เป็นไปตามมาตรฐาน และสามารถประเมินผลได้ โดยมีข้อกำหนดอย่างน้อยดังนี้

๓.๑๗.๑. ผู้รับจ้างต้องออกแบบและตั้งค่าการเชื่อมต่อ WebSocket ทั้งฝั่ง Client และ Server ให้เป็นไปตามมาตรฐานอุตสาหกรรม โดยมีข้อกำหนดขั้นต่ำ ดังนี้

๓.๑๗.๑.๑. มาตรฐาน IETF RFC ๖๔๕๕ (The WebSocket Protocol) เช่น ต้องสามารถรักษาการเชื่อมต่อแบบถาวร (Persistent Connection) ต้องสอดคล้องกับโดยไม่เกิด Timeout หรือถูกตัดการเชื่อมต่อก่อนการประมวลผลเสร็จสิ้น

๓.๑๗.๑.๒. ต้องอ้างอิงและปฏิบัติตามแนวปฏิบัติด้านความปลอดภัยจากแหล่งข้อมูลมาตรฐานอย่างน้อยดังต่อไปนี้

๑) OWASP Web Security Testing Guide (WSTG) สำหรับกระบวนการทดสอบด้านความปลอดภัยของเว็บ

๒) OWASP Cheat Sheet Series สำหรับแนวทางการออกแบบ/เขียนโค้ดให้ปลอดภัย เช่น Authentication, Input Validation, Secure Headers

๓) Common Weakness Enumeration (CWE) โดยอ้างอิงจากฐานข้อมูลของ MITRE

๓.๑๗.๒. การจัดการช่องโหว่ (Vulnerability Management) โดยผู้รับจ้างต้องดำเนินการอย่างน้อยดังนี้

๓.๑๗.๒.๑. รวบรวม วิเคราะห์ และจัดทำรายการ CWE ที่เกี่ยวข้องกับระบบและเทคโนโลยีที่ใช้จริง

๓.๑๗.๒.๒. ดำเนินการแก้ไข (Mitigation) ช่องโหว่ที่ตรวจพบทั้งหมดและไม่อยู่ในระดับ Critical หรือ High Severity

๓.๑๗.๒.๓. ต้องมีหลักฐานการตรวจสอบโดยสามารถตรวจสอบย้อนหลังได้

๓.๑๗.๓. ข้อกำหนดฝั่ง Server โดยผู้รับจ้างต้องออกแบบและตั้งค่าระบบฝั่ง Server ให้รองรับการเชื่อมต่อผ่าน WebSocket ได้อย่างมีประสิทธิภาพ ปลอดภัย และรองรับปริมาณผู้ใช้งานได้ตามที่กำหนด โดยมีข้อกำหนดดังนี้

๓.๑๗.๓.๑. รองรับจำนวนผู้ใช้งานพร้อมกัน (Concurrent Connections) โดยต้องรองรับการเชื่อมต่อพร้อมกันไม่น้อยกว่า จำนวนที่ระบบคาดว่าจะใช้งานพร้อมกันสูงสุดที่กำหนด + buffer ๓๐%

๓.๑๗.๓.๒. การรักษาสถานะการเชื่อมต่อ (Connection Keep-Alive) โดยต้องมีการส่ง Ping/Pong หรือ Heartbeat ตามมาตรฐาน WebSocket เพื่อรักษาการเชื่อมต่อ และตรวจสอบสถานะของทั้งฝั่ง Client และ Server

๓.๑๗.๓.๓. การตั้งค่า Timeout สำหรับหากไม่มีการใช้งาน (Idle Connection) ต้องสามารถตั้งค่า timeout ได้อย่างน้อย ๕ นาที เพื่อป้องกันการเชื่อมต่อที่ไม่มีการใช้งานกินทรัพยากรเกินความจำเป็น

๓.๑๗.๓.๔. การป้องกันการโจมตี โดยต้องสามารถดำเนินอย่างน้อยดังนี้

๑) ต้องรองรับการตั้งค่าจำกัดจำนวนการเชื่อมต่อ (Connection Limit)

๒) ต้องสามารถตั้งค่า Rate Limit เช่น จำกัด request ต่อวินาที เพื่อป้องกันการโจมตี

๓) ต้องสามารถจำกัด Memory Usage ต่อการเชื่อมต่อ เพื่อไม่ให้ผู้ไม่หวังดีเชื่อมต่อหลาย session แล้วกินทรัพยากรจนระบบล่ม

๓.๑๗.๓.๕. การบันทึกข้อมูลการเชื่อมต่อ (Connection Logging) โดยต้องสามารถดำเนินอย่างน้อยดังนี้

๑) ต้องมีระบบบันทึก Log ทุกครั้งที่มีการเชื่อมต่อหรือตัดการเชื่อมต่อ

๒) ข้อมูลที่ต้องบันทึกอย่างน้อยต้องมี เช่น

๓) ที่เกิดเหตุการณ์ (timestamp)

๔) หมายเลขไคลเอนต์หรือ IP Address ของผู้เชื่อมต่อ (Client ID/IP)

๕) เหตุผลของข้อผิดพลาดหรือการตัดการเชื่อมต่อ (Error Reason)

๓.๑๗.๔. ข้อกำหนดฝั่ง Client โดย ผู้รับจ้างต้องออกแบบและพัฒนาระบบฝั่ง Client ให้สามารถเชื่อมต่อกับ WebSocket ได้อย่างมีประสิทธิภาพ มั่นคง และรองรับสถานการณ์ต่าง ๆ ตามข้อกำหนดดังนี้

๓.๑๗.๔.๑. การกลับมาเชื่อมต่อใหม่อัตโนมัติ (Auto Reconnect) หากเมื่อผู้ใช้ (Client) หลุดจากการเชื่อมต่อโดยไม่ตั้งใจ เช่น สัญญาณเน็ตหลุด โดยระบบต้องสามารถ เชื่อมต่อกลับ (Resume Connection) ได้โดยอัตโนมัติ ภายใน ๕ วินาที เพื่อให้ผู้ใช้งานใช้งานได้อย่างต่อเนื่อง ไม่สะดุด

๓.๑๗.๔.๒. ระบบจับเวลาตรวจสอบการตอบสนอง (Timeout Handling) หาก Client ส่งข้อมูลไปยัง Server แล้วไม่ได้รับการตอบกลับภายในเวลาที่กำหนด เช่น ๓๐ วินาที ระบบต้องมีการตัดการเชื่อมต่อและแจ้งเตือน หรือพยายาม reconnect ใหม่โดยอัตโนมัติ เพื่อไม่ให้เกิด session ค้างหรือเปลืองทรัพยากร

๓.๑๗.๔.๓. ตรวจสอบความถูกต้องของข้อมูลและสถานะการเชื่อมต่อ (Data & Connection Integrity) โดยระบบฝั่ง Client ต้องตรวจสอบว่า ข้อมูลที่รับ-ส่งกับ Server ถูกต้องครบถ้วน และต้องติดตามสถานะการเชื่อมต่อแบบเรียลไทม์ หากเกิดความผิดปกติต้องรู้ตัวทันที

๓.๑๗.๔.๔. การแจ้งเตือนผู้ใช้งาน (User Notification) ในกรณีที่เกิดปัญหาการเชื่อมต่อ เช่น หลุดบ่อย ไม่มีการตอบสนอง Client ต้องสามารถแสดงข้อความแจ้งเตือน (Notification) ให้ผู้ใช้งานทราบอย่างเหมาะสม เช่น "กำลังเชื่อมต่อใหม่" หรือ "เกิดข้อผิดพลาดในการเชื่อมต่อ"

๓.๑๗.๔.๕. รองรับผู้ใช้งานจำนวนมาก (Concurrent Connections) โดย Client ต้องออกแบบให้สามารถบริหารจัดการ การเชื่อมต่อพร้อมกันจำนวนมาก ได้โดยไม่กระทบต่อประสิทธิภาพการทำงานของระบบ เช่น กรณีที่มีผู้ใช้งานหลายร้อยคน การเชื่อมต่อของแต่ละคนต้องไม่ทำให้ระบบหน่วงหรือกระตุก

๓.๑๘. ฐานข้อมูล (Database)

๓.๑๘.๑. ผู้รับจ้างต้องออกแบบและพัฒนาฐานข้อมูลโดยเลือกใช้ระบบฐานข้อมูลเชิงสัมพันธ์ (RDBMS) หรือ NoSQL ให้เหมาะสมกับลักษณะข้อมูลและการใช้งาน พร้อมรองรับสถาปัตยกรรม Master-Slave หรือ Clustering เพื่อให้มีความพร้อมใช้งานสูง (High Availability) และขยายขนาดได้ (Scalability) อย่างมีประสิทธิภาพ โดยโครงสร้างต้องรับมือกับปริมาณข้อมูลและจำนวนผู้ใช้งานจำนวนมากได้ตามแนวทางสากลด้านการเพิ่มประสิทธิภาพ (Performance Optimization) และกลยุทธ์การขยายฐานข้อมูล (Database Scaling Strategies) โดยมีข้อกำหนดอย่างน้อยดังต่อไปนี้

๓.๑๘.๑.๑. Indexing

๑) ต้องมีการวิเคราะห์และสร้างดัชนี (Index) อย่างเหมาะสมกับรูปแบบการใช้งาน (Query Pattern)

๒) ต้องหลีกเลี่ยงการสร้าง Index ซ้ำซ้อนหรือมากเกินไป ซึ่งอาจกระทบต่อประสิทธิภาพการเขียนข้อมูล

๓) ต้องมีการตรวจสอบและปรับปรุง Index อย่างสม่ำเสมอ เช่น ผ่าน EXPLAIN, ANALYZE

๓.๑๘.๑.๒. Query Optimization

๑) หลีกเลี่ยง SELECT * และใช้ฟิลด์เฉพาะที่จำเป็น

๒) เขียน WHERE, JOIN, ORDER BY, GROUP BY ให้มีประสิทธิภาพ

๓) ตรวจสอบแผนการประมวลผล (Query Plan) โดยใช้คำสั่งเช่น EXPLAIN, EXPLAIN ANALYZE

๓.๑๘.๑.๓. Connection Pooling

๑) ต้องใช้เทคนิค Connection Pooling เช่น HikariCP, pgBouncer เพื่อลดการเปิด/ปิด connection บ่อยครั้ง

๒) ช่วยลด Latency และเพิ่ม Throughput ของระบบ

๓.๑๘.๑.๔. Caching

๑) ต้องมีการพิจารณาใช้ Cache ทั้งในระดับ Application เช่น Redis, Memcached และระดับ Database เช่น Query Cache

๒) ต้องจัดการ Expiry และ Invalidation อย่างเหมาะสม

๓.๑๘.๑.๕. Sharding และ Partitioning

๑) สำหรับระบบที่มี Dataset ใหญ่หรือปริมาณการเขียนสูง ต้องพิจารณาใช้ Sharding หรือ Partitioning

๒) Sharding: แยกฐานข้อมูลตาม Key เช่น UserID

๓) Partitioning: แยกตามเงื่อนไข เช่น Range หรือ Hash

๔) ต้องมีแผนการจัดการและตรวจสอบความสมดุลระหว่าง Shard

๓.๑๘.๑.๖. Replication

๑) ต้องออกแบบการทำ Replication เพื่อรองรับ Read Scaling และ High Availability เช่น Master-Slave, Multi-Master

๒) ต้องจัดการ Latency, Conflict และ Read Consistency ให้เหมาะสม

๓.๑๘.๑.๗. Hardware และ Resource

๑) ต้องพิจารณาทรัพยากรของฐานข้อมูลให้เพียงพอ เช่น RAM, CPU และ Storage

๒) หากเป็น Cloud ต้องระบุประเภท instance หรือ storage ที่มี IOPS สูง

๓.๑๘.๑.๘. Monitoring

๑) ต้องมีระบบตรวจสอบสถานะและประสิทธิภาพฐานข้อมูล เช่น Query Latency, CPU/RAM/Disk I/O, Connection Utilization และควรแสดงการแจ้งเตือนเมื่อเกิด Query ที่ช้าหรือมีปัญหา เช่น Prometheus, Grafana, pg_stat_statements, MySQL Performance Schema

๓.๑๘.๑.๙. Normalization / Denormalization

๑) ต้องออกแบบ Schema ให้สมดุลระหว่าง Normalization (ลดซ้ำซ้อน) และ Denormalization (เพิ่มประสิทธิภาพ Read) ตาม Use Case

๒) ต้องมีการประเมินผลกระทบต่อประสิทธิภาพและความสมบูรณ์ของข้อมูล (Data Integrity)

๓.๑๘.๑.๑๐. Materialized Views

๑) พิจารณาใช้ Materialized View สำหรับคำสั่ง Query ที่ซับซ้อนและใช้บ่อย โดยแยกการคำนวณล่วงหน้า

๒) ต้องมีการกำหนดการ Refresh View ที่เหมาะสม เช่น On-Demand หรือ Scheduled

๓.๑๘.๑.๑๑. Vertical Scaling

๑) ในกรณีที่ฐานข้อมูลต้องการความเร็วสูงทันที (short-term scale up) ต้องสามารถเพิ่ม CPU, RAM หรือ Disk I/O ได้

๒) ควรออกแบบให้สามารถย้ายจาก Vertical ไปสู่ Horizontal ได้ในอนาคต

๓.๑๘.๒. คุณสมบัติของธุรกรรมฐานข้อมูล (ACID Compliance) ผู้รับจ้างต้องออกแบบและพัฒนาฐานข้อมูลให้รองรับคุณสมบัติ ACID (Atomicity, Consistency, Isolation, Durability) เพื่อรับประกันความถูกต้อง ความสมบูรณ์ และความเชื่อถือได้ของข้อมูล โดยมีข้อกำหนดอย่างน้อยดังต่อไปนี้

๓.๑๘.๒.๑. Atomicity – การทำธุรกรรมแบบรวมทั้งหมด

๑) การดำเนินการภายในแต่ละธุรกรรม (Transaction) ต้องสำเร็จทั้งหมดหรือไม่ดำเนินการเลย (All-or-Nothing)

๒) ในกรณีที่เกิดข้อผิดพลาด ต้องสามารถ Rollback ฐานข้อมูลกลับไปสู่สถานะก่อนเริ่มธุรกรรมนั้นได้อย่างสมบูรณ์

๓.๑๘.๒.๒. Consistency – ความสอดคล้องของข้อมูล

๑) ต้องกำหนด Constraints ที่เหมาะสม เช่น Foreign Key, Unique และ Check Constraint บนตารางฐานข้อมูล โดยพิจารณาจากหลักการออกแบบฐานข้อมูลที่ถูกต้อง และตามความจำเป็นที่เหมาะสมกับลักษณะข้อมูลและกฎเกณฑ์ของธุรกิจ (Business Rules) ของระบบได้อย่างถูกต้องครบถ้วน

๒) ทุกธุรกรรมที่เกิดขึ้น ต้องไม่ทำให้ข้อมูลผิดกฎหรือขัดกับเงื่อนไขของระบบ ทั้งก่อนและหลังการทำธุรกรรมเสมอ

๓.๑๘.๒.๓. Isolation – ความเป็นอิสระของธุรกรรม

๑) ธุรกรรมแต่ละชุดต้องถูกดำเนินการแยกจากกัน แม้มีการทำงานพร้อมกัน (Concurrent Transaction)

๒) เลือกใช้ Isolation Level ที่เหมาะสม เช่น REPEATABLE READ หรือ SERIALIZABLE เพื่อป้องกันปัญหาการอ่านข้อมูลที่ยังไม่ได้ commit (Dirty Read) หรือข้อมูลที่เปลี่ยนแปลงในระหว่างการอ่าน (Phantom Read) ซึ่งอาจทำให้ผลลัพธ์ไม่ถูกต้อง

๓) กรณีที่ระบบมีการใช้งานพร้อมกันจำนวนมาก (High Load) ต้องออกแบบเพื่อรองรับการเกิดความขัดแย้งระหว่างการอ่านและเขียนข้อมูล (Read-Write Conflict) และหลายธุรกรรมแย่งกันล็อกข้อมูลเดียวกัน (Lock Contention)

๓.๑๘.๒.๔. Durability – ความคงทนของข้อมูล

๑) ข้อมูลที่ถูก Commit แล้วต้องไม่สูญหาย แม้เกิดระบบล้มเหลว เช่น ไฟดับ ระบบรีสตาร์ต

๒) ต้องมีการจัดเก็บ (Persist) ข้อมูลลงใน Disk หรือ Log ที่เชื่อถือได้ และควรมีการทำ Replication อย่างน้อย ๑ ชุด (High Availability)

๓) ในกรณีระบบที่มีหลายเครื่องแม่ข่าย (Distributed System) ต้องใช้เทคนิคการ Re-sync, WAL (Write-Ahead Logging) หรือ Consensus Protocol เช่น Raft เพื่อรับประกัน Durability

๓.๑๘.๓. ผู้รับจ้างต้องเลือกประเภทของฐานข้อมูล (Database Type) ให้เหมาะสมกับลักษณะของข้อมูลและลักษณะการใช้งาน (Workload) ของแต่ละฟังก์ชันในระบบ เพื่อให้สามารถตอบสนองความต้องการเฉพาะด้านของงานได้อย่างมีประสิทธิภาพ โดยต้องพิจารณาใช้ประเภทฐานข้อมูลที่แตกต่างกันตามลักษณะการใช้งานทั้งนี้ การเลือกใช้งานต้องสามารถให้เหตุผลประกอบที่ชัดเจน และสามารถตรวจสอบได้ตามหลักเกณฑ์ที่ยอมรับในอุตสาหกรรม ต้องยึดหลักเกณฑ์อย่างน้อยดังต่อไปนี้

๓.๑๘.๓.๑. Relational Database (SQL)

๑) ใช้สำหรับระบบที่มีข้อมูลโครงสร้างชัดเจน (Structured Data) เช่น ระบบจัดการผู้ใช้งาน ระบบสิทธิ์ ระบบการแจ้งเตือน

๒) ต้องรองรับการทำ Transaction, ACID, Indexing และ Foreign Key Constraints

๓) เช่น MySQL, PostgreSQL

๓.๑๘.๓.๒. In-memory Store

๑) ใช้สำหรับข้อมูลที่มีการเข้าถึงบ่อยและต้องการความเร็วสูง เช่น Session, Flash Sale, Caching

๒) ควรใช้ร่วมกับกลไก Expiration/TTL เพื่อจัดการหน่วยความจำ

๓) เช่น Redis, Memcached

๓.๑๘.๓.๓. Time-series Database

๑) ใช้สำหรับการเก็บข้อมูลตามเวลา เช่น Stock Market,Sensor, Monitoring

๒) ต้องรองรับฟีเจอร์การเก็บตามเวลา เช่น Retention Policy, Rollup, Compression

๓) เช่น InfluxDB, TimescaleDB, Prometheus, Graphite

๓.๑๘.๓.๔. Graph Database

๑) ใช้สำหรับข้อมูลที่มีความสัมพันธ์ซับซ้อน เช่น ความสัมพันธ์ผู้ใช้งาน, Fraud Detection, Recommendation Engine

๒) ต้องสามารถ Query ข้าม Node ได้อย่างมีประสิทธิภาพ

๓) เช่น Neo๔j, Amazon Neptune

๓.๑๘.๓.๕. Document Store

๑) เหมาะกับข้อมูลที่ไม่แน่นอนและมีขนาดใหญ่ เช่น Metadata, ไฟล์ JSON, ไฟล์ภาพ

๒) เหมาะกับระบบที่ไม่ต้อง Join หลายตาราง เช่น CMS, ระบบจัดเก็บเอกสาร

๓) เช่น MongoDB, CouchDB

๓.๑๘.๓.๖. Wide Column Store

๑) สำหรับงานด้าน Big Data, Analytics, Reporting ที่ต้องการการอ่านข้อมูลขนาดใหญ่จากตารางแบบ Denormalized

๒) รองรับ Query แบบ Scan หรือ Aggregate ได้

๓) เช่น Apache Cassandra, HBase, Google Bigtable

๓.๑๘.๓.๗. เงื่อนไขทั่วไปเพิ่มเติม

๑) ต้องเลือกประเภทฐานข้อมูลตามลักษณะข้อมูล ไม่ใช้ฐานข้อมูลประเภทเดียวสำหรับทุก Use Case โดยไม่วิเคราะห์

๒) หากใช้ NoSQL หรือ Time-Series ต้องมีเหตุผลประกอบด้านประสิทธิภาพ ปริมาณข้อมูล หรือรูปแบบการเข้าถึง

๓.๑๘.๔. ผู้รับจ้างต้องใช้กระบวนการปรับใช้ฐานข้อมูลและการควบคุมเวอร์ชัน (Database Deployment & Version Control) และจัดการการเปลี่ยนแปลงของฐานข้อมูล (Database Schema Migration) อย่างมีระบบ โดยมีข้อกำหนดอย่างน้อยดังต่อไปนี้

๓.๑๘.๔.๑. การควบคุมเวอร์ชันของฐานข้อมูล

๑) ต้องใช้เครื่องมือสำหรับการควบคุมเวอร์ชันของฐานข้อมูล เช่น Liquibase หรือเครื่องมือที่มีคุณสมบัติเทียบเท่า เพื่อจัดการการเปลี่ยนแปลงของโครงสร้างฐานข้อมูลอย่างเป็นระบบ

๒) การเปลี่ยนแปลงของฐานข้อมูลต้องถูกบันทึกในรูปแบบของไฟล์ที่สามารถควบคุมเวอร์ชันได้ เช่น SQL, XML, YAML และจัดเก็บร่วมกับซอร์สโค้ดของแอปพลิเคชันในระบบ Version Control เช่น Git

๓) ต้องสามารถติดตามประวัติการเปลี่ยนแปลงของฐานข้อมูลได้อย่างชัดเจน และสามารถย้อนกลับ (Rollback) ไปยังสถานะก่อนหน้าได้ในกรณีที่เกิดปัญหา

๓.๑๘.๔.๒. การปฏิบัติตามแนวทางมาตรฐาน

๑) การดำเนินการเปลี่ยนแปลงฐานข้อมูลต้องเป็นไปตามแนวทางปฏิบัติที่ยอมรับในระดับสากล เช่น การแยกการเปลี่ยนแปลงโครงสร้างฐานข้อมูลออกจากการเปลี่ยนแปลงข้อมูล (Data Migration) และการจัดการการเปลี่ยนแปลงอย่างเป็นระบบ (Database Schema Migration Best Practices)

๒) การรวมกระบวนการเปลี่ยนแปลงฐานข้อมูลเข้ากับกระบวนการ CI/CD เพื่อให้สามารถปรับใช้การเปลี่ยนแปลงได้อย่างรวดเร็วและปลอดภัย (Continuous Integration/Continuous Deployment (CI/CD)

๓) การประสานงานระหว่างทีมพัฒนาและทีมปฏิบัติการเพื่อให้การเปลี่ยนแปลงฐานข้อมูลเป็นไปอย่างราบรื่น (DevSecOps Practices)

๓.๑๘.๕. การใช้งาน Database Middleware เพื่อแยกคำสั่งอ่าน-เขียน (Read-Write Splitting) และจัดการ Replica ผู้รับจ้างต้องออกแบบและติดตั้งระบบฐานข้อมูลให้สามารถทำงานร่วมกับ Database

Middleware เพื่อแยกการประมวลผลคำสั่งอ่านและเขียนออกจากกันอย่างมีประสิทธิภาพ รองรับการกระจายโหลด (Load Balancing) และเพิ่มความสามารถในการรองรับการขยายระบบและความทนทานของระบบฐานข้อมูล โดยมีข้อกำหนดอย่างน้อยดังต่อไปนี้

๓.๑๘.๕.๑. คุณสมบัติของ Database Middleware

๑) ทำหน้าที่เป็นชั้นกลาง (Middleware Layer) ระหว่างระบบ Application กับระบบฐานข้อมูล (Database Cluster)

๒) รองรับการสื่อสารผ่าน Database Network Protocol ที่เป็นมาตรฐานและสามารถใช้งานร่วมกับไคลเอนต์ที่มีอยู่ได้ เช่น TCP-based protocol ของฐานข้อมูลที่เลือกใช้

๓) สามารถตั้งกฎการส่งต่อ (Routing Rules) ตามประเภทคำสั่ง เช่น

๓.๑) คำสั่งเขียน (INSERT, UPDATE, DELETE) ⟶ ไปยัง Primary Database

๓.๒) คำสั่งอ่าน (SELECT) ⟶ ไปยัง Replica Database ที่เหมาะสม

๔) รองรับการกำหนด Routing อย่างน้อยดังนี้

๔.๑) ประเภทคำสั่ง (Statement Type)

๔.๒) ผู้ใช้งาน (User Context)

๔.๓) ฐานข้อมูลหรือสคีมา (Schema)

๔.๔) ปริมาณโหลด (Connection Load)

๓.๑๘.๕.๒. ความสามารถด้าน Read-Write Splitting และ Replica

๑) Middleware ต้องสามารถแยก Read และ Write ไปยัง Database Node ที่เหมาะสมโดยอัตโนมัติ

๒) รองรับการทำ Replication Monitoring และสามารถจัดการปัญหา Replica lag ได้ เช่น Redirect ไป Primary เมื่อจำเป็น

๓) สามารถปรับเปลี่ยนจำนวน Replica ได้แบบยืดหยุ่นโดยไม่กระทบต่อแอปพลิเคชัน

๓.๑๘.๕.๓. ความสามารถด้านความทนทานและความพร้อมใช้งานสูง (High Availability)

๑) Middleware ต้องรองรับการทำงานแบบ Cluster หรือ Active-Standby เพื่อหลีกเลี่ยง Single Point of Failure

๒) รองรับการตรวจสอบสถานะของ Database Node และทำ Failover อัตโนมัติเมื่อมีปัญหา

๓) สามารถบันทึก Audit Log และ Query Routing Log เพื่อตรวจสอบย้อนหลัง

๓.๑๙. OpenAPI

๓.๑๙.๑. การออกแบบและจัดเก็บ OpenAPI Specification

๓.๑๙.๑.๑. ผู้รับจ้างต้องออกแบบ API โดยใช้ OpenAPI Specification เวอร์ชัน ๓.๐ ขึ้นไปเป็นมาตรฐานหลัก

๓.๑๙.๑.๒. ผู้รับจ้างต้องจัดทำ ไฟล์ .yaml หรือ .json ของ OpenAPI สำหรับ ทุก API Service

๓.๑๙.๑.๓. ต้องใช้ Semantic Versioning (MAJOR.MINOR.PATCH) ในการจัดการเวอร์ชันของ API

๓.๑๙.๑.๔. ออกแบบ API ให้สอดคล้องและรองรับสถาปัตยกรรม Microservices

๓.๑๙.๒. ข้อกำหนดด้านการจัดเก็บและบริหารจัดการ API Specification

๓.๑๙.๒.๑. จัดเก็บ API Specification ใน Git-based Repository โดยแยกออกจากซอร์สโค้ดหลัก

๓.๑๙.๒.๒. ต้องจัดการเวอร์ชันของ Specification ด้วยวิธี Branching Strategy ที่ชัดเจน เช่น Git Flow

๓.๑๙.๓. ข้อกำหนดด้านการแก้ไขและปรับปรุง API Specification

๓.๑๙.๓.๑. ใช้เครื่องมือ Visual Editor (เช่น Swagger Editor, Stoplight Studio หรือเครื่องมือเทียบเท่า) เพื่ออำนวยความสะดวกในการปรับแก้ Specification

๓.๑๙.๓.๒. ต้องมี Workflow ในการจัดการและตรวจสอบคุณภาพ Specification เช่น การใช้ OpenAPI Validator, Spectral หรือเครื่องมือที่เทียบเท่า

๓.๑๙.๔. ข้อกำหนดด้านการนำ API Specification ไปใช้ (API Deployment)

๓.๑๙.๔.๑. มีแนวทางหรือเครื่องมือในการนำ Specification ไปใช้งานจริงโดยไม่ต้องแก้ไขหรือคอมไพล์ซอร์สโค้ดหลัก เช่น API Gateway, Auto-Generation Framework, Mock Server หรือ Low-Code Platform

๓.๑๙.๔.๒. Pipeline CI/CD ต้องรองรับการอ่านและ Deploy Specification ได้แบบอัตโนมัติและรองรับ Zero Downtime Deployment

๓.๑๙.๕. ข้อกำหนดการเปลี่ยนแปลง API (Change Management)

๓.๑๙.๕.๑. ต้องกำหนดแนวทางการจัดการ Breaking Change อย่างชัดเจน พร้อมการทำ Diff Report ของ API Specification เวอร์ชันใหม่และเก่า

๓.๑๙.๕.๒. ระบบต้องรองรับการ Synchronize Specification ใหม่ไปยัง API Gateway และ Mock Server โดยอัตโนมัติ

๓.๑๙.๖. ข้อกำหนดด้านเอกสารประกอบและตัวอย่างการใช้งาน API (API Documentation)

๓.๑๙.๖.๑. ต้องจัดทำเอกสารในรูปแบบ Swagger UI หรือ ReDoc พร้อมทั้งตัวอย่างโค้ดครบถ้วน (เช่น Curl, Postman Collection, SDK Stub)

๓.๑๙.๖.๒. เอกสารต้องระบุขั้นตอนการปรับปรุง Specification อย่างละเอียดและชัดเจน

๓.๑๙.๗. ข้อกำหนดเกี่ยวกับซอฟต์แวร์และลิขสิทธิ์ (Software Licensing)

๓.๑๙.๗.๑. เครื่องมือและเทคโนโลยีที่นำมาใช้ต้องเป็น Open Source หรือมีลิขสิทธิ์แบบ Perpetual License ยกเว้นได้รับอนุมัติจากผู้ว่าจ้าง

๓.๑๙.๗.๒. ข้อกำหนดด้านการทำงานแบบอัตโนมัติและ DevSecOps (Automation & DevSecOps Requirements)

๓.๑๙.๘. ผู้รับจ้างต้องจัดทำระบบ CI/CD Pipeline อัตโนมัติสำหรับ API โดยครอบคลุมขั้นตอนการตรวจสอบคุณภาพ API Specification, การทดสอบความปลอดภัยอัตโนมัติ (Security Automation Testing), และการ Deploy API Specification ไปยัง API Gateway หรือระบบกลาง โดยอ้างอิงจาก OpenAPI Specification ที่กำหนดไว้

๓.๑๙.๙. ต้องใช้เครื่องมือ Automation Testing ที่เหมาะสม เช่น Postman CLI, Newman, หรือเทียบเท่า ในการทดสอบ API อัตโนมัติทุกครั้งก่อนการ Deploy

๓.๑๙.๑๐. Pipeline ต้องมีขั้นตอน Security Scanning โดยใช้เครื่องมือที่เป็นที่ยอมรับในระดับสากล เช่น OWASP ZAP, SonarQube หรือเทียบเท่า และต้องตรวจสอบตาม OWASP API Security Top 10

๓.๑๙.๑๑. ต้องมีขั้นตอนการทำ Code Review และ Specification Review แบบอัตโนมัติภายใน Pipeline ก่อนที่จะนำ API ไปใช้งานจริง

๓.๑๙.๑๒. กระบวนการ Deployment จะต้องรองรับการทำ Zero Downtime Deployment และสามารถ Rollback กลับไปเวอร์ชันก่อนหน้าได้อย่างอัตโนมัติและรวดเร็ว

๓.๑๙.๑๓. กระบวนการทำงานทั้งหมดต้องเป็นไปตามหลักการของ DevSecOps โดยผนวก Security Testing ไว้ในทุกขั้นตอนของการพัฒนาและ Deployment (Shift-left Security)

๓.๑๙.๑๔. ข้อกำหนดเพิ่มเติมด้านความปลอดภัยและประสิทธิภาพ (Security and Performance)

๓.๑๙.๑๔.๑. ระบบต้องรองรับ OAuth ๒.๐, JWT, OWASP API Top ๑๐ และแนวทาง Secure Coding Practices

๓.๑๙.๑๔.๒. มีการวัดประสิทธิภาพ API ด้วย Prometheus หรือ OpenTelemetry

๓.๑๙.๑๔.๓. สามารถติดตามสถานะ API และ Microservices ผ่าน Dashboard กลาง เช่น Grafana แสดงข้อมูล Metrics แบบ Real-time

๓.๒๐. การจัดเก็บข้อมูลแบบ Object Storage (Object Storage Design and Compliance) ผู้รับจ้างต้องออกแบบและพัฒนาโซลูชันจัดเก็บข้อมูลโดยใช้ Object Storage ให้เหมาะสมกับระบบงาน โดยเฉพาะระบบแบบ Microservices และระบบที่มีข้อมูลไม่เป็นโครงสร้าง (Unstructured Data) โดยมีข้อกำหนดดังต่อไปนี้

๓.๒๐.๑. ลักษณะของ Object Storage ที่ต้องรองรับอย่างน้อยดังนี้

๓.๒๐.๑.๑. ต้องจัดเก็บข้อมูลในลักษณะ “Object” โดยแต่ละ Object ต้องประกอบด้วยข้อมูลอย่างน้อยดังนี้

๑) Data Payload

๒) Metadata (Custom Metadata ได้)

๓) Unique Identifier (Object ID / URI)

๓.๒๐.๑.๒. ต้องรองรับโครงสร้างแบบ Flat Structure (ไม่มี Directory Hierarchy)

๓.๒๐.๑.๓. ต้องรองรับการเข้าถึงผ่าน RESTful API ได้

๓.๒๐.๒. ความเหมาะสมกับงาน Microservices

๓.๒๐.๒.๑. แต่ละ Microservice ต้องสามารถส่ง/ดึงข้อมูลผ่าน API ไปยัง Object Store ได้อย่างอิสระ (Decoupled Storage)

๓.๒๐.๒.๒. ต้องออกแบบให้รองรับการใช้งานร่วมกับระบบ Cloud Native และ Containerized Services เช่น Kubernetes

๓.๒๐.๒.๓. ต้องสามารถตั้ง Policy ตาม Bucket หรือ Service เช่น TTL, Access Control

๓.๒๐.๓. ความปลอดภัยในการเข้าถึง (Authentication & Authorization)

๓.๒๐.๓.๑. การเข้าถึง Object Store ต้องรองรับมาตรฐานความปลอดภัย ได้แก่

๑) Token-based Access (เช่น OAuth ๒.๐, JWT)

๒) Pre-signed URL สำหรับการอนุญาตแบบจำกัดเวลา

๓) Access Control Policy (ACL) และ Bucket Policy

๓.๒๐.๓.๒. ต้องมีการแยกสิทธิ์ระหว่าง Service และ User แต่ละรายอย่างชัดเจน (Least Privilege)

๓.๒๐.๔. มาตรฐานและผลิตภัณฑ์ที่ยอมรับ

๓.๒๐.๔.๑. ผู้รับจ้างสามารถใช้ผลิตภัณฑ์ Object Storage ที่ได้มาตรฐานและรองรับการทำ High Durability เช่น MinIO, Ceph Object Gateway

๓.๒๐.๔.๒. มาตรฐานอ้างอิง

๑) NIST ๘๐๐-๒๐๙ (Security Guidelines for Storage Infrastructure)

๒) CNCF Cloud Native Storage Principles

๓.๒๐.๕. รองรับ Use Cases อย่างน้อยดังนี้

๓.๒๐.๕.๑. การจัดเก็บข้อมูลไม่เป็นโครงสร้าง (Unstructured Media: ภาพ, วิดีโอ, เสียง, เอกสาร)

๓.๒๐.๕.๒. การเก็บข้อมูลแบบ Data Archiving

๓.๒๐.๕.๓. การทำ Cloud-Native Storage สำหรับระบบ Microservices

๓.๒๐.๕.๔. การเก็บข้อมูลจาก IoT หรือ Sensor (Internet of Things)

๓.๒๐.๕.๕. รองรับระบบ Backup & Recovery

๓.๒๐.๕.๖. รองรับการทำงานร่วมกับ Data Lake, Data Warehouse และ Machine Learning Pipelines

๓.๒๑. การวัดและปรับปรุงประสิทธิภาพเว็บไซต์ (Website Performance Metrics & Optimization) ผู้รับจ้างต้องดำเนินการออกแบบ พัฒนา และทดสอบเว็บไซต์หรือ Web Application ให้มีประสิทธิภาพสูง และสามารถตรวจวัดตามดัชนีชี้วัดสำคัญ (Performance Metrics) ได้อย่างน้อยดังต่อไปนี้

๓.๒๑.๑. ดัชนีชี้วัดประสิทธิภาพหลัก (Core Performance Metrics)

๓.๒๑.๑.๑. Load Time

๑) ที่ใช้ในการโหลดหน้าเว็บทั้งหมดจนสามารถแสดงผลได้

๒) ต้องใช้เครื่องมือวัด เช่น Lighthouse, WebPageTest หรือเทียบเท่า

๓.๒๑.๑.๒. Time to First Byte (TTFB)

๑) ระยะเวลาที่เบราว์เซอร์ได้รับไบต์ข้อมูลแรกจากเซิร์ฟเวอร์

๒) ต้องมีค่า < ๕๐๐ms สำหรับผู้ใช้งานในภูมิภาคเป้าหมาย

๓.๒๑.๑.๓. Request Count

๑) จำนวนคำขอ (HTTP Requests) ที่ต้องใช้เพื่อโหลดหน้าเว็บ

๒) ควรมีการรวมไฟล์หรือปรับลดเพื่อไม่เกินเกณฑ์ ๗๐-๘๐ requests ต่อหน้า

๓.๒๑.๑.๔. DOMContentLoaded (DCL)

๑) เวลาที่ HTML โหลดครบ ไม่รวม CSS/JS/รูปภาพ

๒) ใช้สำหรับประเมินความพร้อมใช้งานเบื้องต้นของโครงสร้าง DOM

๓.๒๑.๑.๕. Time to Above-the-Fold Load

๑) เวลาที่เนื้อหาบนสุดของเว็บเพจ (ก่อนการ scroll) ปรากฏครบ

๒) ควรน้อยกว่า ๒ วินาที

๓.๒๑.๑.๖. First Contentful Paint (FCP)

๑) เวลาแรกที่เบราว์เซอร์แสดงเนื้อหา เช่น ข้อความหรือรูปภาพ

๒) เป้าหมาย < ๑.๘ วินาที

๓.๒๑.๑.๗. Page Size

๑) ขนาดรวมของ HTML, CSS, JS, ภาพ และ assets ทั้งหมด

๒) ควรไม่เกิน ๒.๕ MB ต่อหน้า หรือมีเทคนิค Lazy Loading

๓.๒๑.๒. ข้อกำหนดเพิ่มเติม

๓.๒๑.๒.๑. ต้องแสดงผลลัพธ์จากเครื่องมือ Performance Audit อย่างน้อย ๒ เครื่องมือ พร้อมค่าเปรียบเทียบก่อน/หลังการปรับปรุง

๓.๒๑.๒.๒. จัดทำรายงานการวัดผลพร้อมคำอธิบายแนวทางการปรับปรุง

๓.๒๑.๒.๓. เว็บไซต์ต้องผ่านการทดสอบทั้งบน Desktop และ Mobile โดยใช้ค่าเกณฑ์ตาม Google Lighthouse หรือ Core Web Vitals เป็นเกณฑ์ขั้นต่ำ

๓.๒๒. ในกรณีที่ผู้รับจ้างมีแนวทางการออกแบบหรือแนวปฏิบัติ (Best Practice) ที่ดีกว่า มีประสิทธิภาพมากกว่า หรือทันสมัยกว่าที่ระบุไว้ในข้อกำหนดนี้ ผู้รับจ้างสามารถเสนอแนวทางดังกล่าวให้ผู้ว่าจ้างพิจารณาได้ ทั้งนี้ ผู้รับจ้างต้องจัดทำข้อมูลเปรียบเทียบเชิงประจักษ์ อย่างชัดเจน เช่น ตารางเปรียบเทียบคุณสมบัติ รายงานผลการทดสอบ หรือกรณีศึกษา (Case Study) เพื่อแสดงให้เห็นว่าแนวทางที่เสนอนั้นเหนือกว่าข้อกำหนดเดิมในด้านคุณภาพ ประสิทธิภาพ ความปลอดภัย หรือความสามารถในการบำรุงรักษา ทั้งนี้ การนำไปใช้ต้องได้รับความเห็นชอบจากผู้ว่าจ้างก่อนดำเนินการ

๓.๒๓. การออกแบบตามข้อ ๓.๑ ถึงข้อ ๓.๒๒ ต้องได้รับการเห็นชอบจากสำนักงานการตรวจเงินแผ่นดิน

๔. การพิสูจน์แนวคิดและสาธิต (Proof of concept and Demo)

๔.๑. พิสูจน์แนวคิดในด้าน DevSecOps

๔.๑.๑. ทดสอบกระบวนการ CI/CD และความสามารถในการทำ Automation

๔.๑.๒. ทดสอบระบบรักษาความปลอดภัยในระดับ Source Code และ Infrastructure

๔.๑.๓. พิสูจน์ว่า DevSecOps สามารถทำงานร่วมกับสถาปัตยกรรมของระบบได้จริง

๔.๒. พิสูจน์แนวคิดในด้าน Restore และ Disaster Recovery

๔.๒.๑. ทดสอบกระบวนการสำรองข้อมูล (Backup) และกู้คืนข้อมูล (Restore)

๔.๒.๒. ประเมินความสามารถในการกู้คืนระบบจากความเสียหาย (Disaster Recovery)

๔.๒.๓. ทดสอบการทำ Failover และการทำงานต่อเนื่อง (High Availability)

๔.๓. พิสูจน์แนวคิดในด้าน Load Balancing และ Auto Scaling

๔.๓.๑. ทดสอบความสามารถในการรองรับจำนวนผู้ใช้พร้อมกัน (Concurrent Users)

๔.๓.๒. ทดสอบการทำงานของระบบภายใต้ภาระงานสูง (Stress Test)

๔.๓.๓. ทดสอบระบบ Auto Scaling ให้ขยายขนาดตาม Workload อัตโนมัติ

๔.๔. พิสูจน์แนวคิดในด้าน Security และ Zero Trust Architecture

๔.๔.๑. ทดสอบการยืนยันตัวตนของผู้ใช้ (Authentication)

๔.๔.๒. ทดสอบการปฏิเสธการเข้าถึงโดยไม่มีสิทธิ์ (Authorization)

๔.๔.๓. ทดสอบการป้องกันการโจมตี เช่น Brute Force, SQL Injection และ XSS

๔.๕. พิสูจน์แนวคิดในด้าน Compatibility และ Cross-Platform

๔.๕.๑. ทดสอบความสามารถของระบบในการทำงานข้ามแพลตฟอร์ม (Cross-Platform)

๔.๕.๒. ทดสอบการทำงานร่วมกับ Browser และอุปกรณ์ที่แตกต่างกัน

๔.๖. พิสูจน์แนวคิดในด้าน Performance และ Response Time

๔.๖.๑. ทดสอบความเร็วในการตอบสนองของระบบ

๔.๖.๒. ทดสอบประสิทธิภาพในการเรียก API และฐานข้อมูล

๕. การพัฒนา (Implementation)

๕.๑. ผู้รับจ้างพัฒนาระบบให้เป็นไปตามรายละเอียดคุณลักษณะเฉพาะตามภาคผนวก ก รายละเอียดคุณลักษณะเฉพาะระบบสารสนเทศของโครงการ

๕.๒. ผู้รับจ้างต้องพัฒนาระบบโดยใช้ภาษาโปรแกรมที่เป็นที่ยอมรับในอุตสาหกรรม ได้แก่ Python, Go, Java หรือ PHP ตามความเหมาะสมกับงานแต่ละส่วน ทั้งนี้ ต้องเลือกใช้ framework หรือเครื่องมือที่รองรับการจัดการเวอร์ชันของโครงสร้างฐานข้อมูล (Database Schema Version Control หรือ Database Migration) ได้อย่างเป็นระบบ เช่น หากใช้ Python ให้ใช้ Django หากใช้ Java ให้ใช้ Spring Boot + Liquibase/Flyway หากใช้ Go ให้ใช้ Gorm/Goose หรือเทียบเท่า หากใช้ PHP ให้ใช้ Laravel หรือเทียบเท่า โดย framework หรือเครื่องมือที่เลือกใช้ต้องมีคุณสมบัติอย่างน้อยดังนี้

๕.๒.๑. บันทึกการเปลี่ยนแปลงโครงสร้างฐานข้อมูล (Schema Migration)

๕.๒.๒. รองรับการ Rollback/Migrate ข้อมูล

๕.๒.๓. บูรณาการกับกระบวนการพัฒนา (CI/CD Pipeline) ได้

๕.๓. ผู้รับจ้างต้องพัฒนาระบบโดยใช้เครื่องมือหรือเฟรมเวิร์คที่รองรับการพัฒนาในลักษณะ RESTful API และสามารถทำงานได้อย่างรวดเร็วและมีความเสถียร โดยต้องสามารถรองรับผู้ใช้จำนวนมากในเวลาเดียวกันได้และทำงานได้บนแพลตฟอร์มที่หลากหลาย

๕.๔. ผู้รับจ้างต้องพัฒนาระบบให้รองรับการยืนยันตัวตนด้วยมาตรฐานที่เป็นที่ยอมรับในระดับสากล เช่น OAuth2, OpenID Connect หรือวิธีการอื่นที่มีความปลอดภัยสูงและรองรับการยืนยันตัวตนแบบหลายขั้นตอน (Multi-Factor Authentication)

๕.๕. ผู้รับจ้างต้องพัฒนาระบบให้รองรับการสื่อสารระหว่างบริการด้วยวิธีที่มีความปลอดภัยสูง เช่น RESTful API หรือ RPC และต้องมีการเข้ารหัสข้อมูลระหว่างการรับ - ส่งข้อมูล

๕.๖. ผู้รับจ้างต้องพัฒนาระบบให้มีความสามารถในการส่งการแจ้งเตือนของระบบผ่านช่องทางต่าง ๆ เช่น Email และ Push Notification โดยสามารถทำงานร่วมกับระบบอื่น ๆ ได้ผ่าน Webhook หรือ API หรือ Integration กับ Monitoring Tools เช่น Grafana, Zabbix, Prometheus หรือเทียบเท่า

๕.๗. ผู้รับจ้างต้องพัฒนาระบบให้สามารถทำ Load Balancing และกระจายทราฟฟิกขาเข้าได้อย่างมีประสิทธิภาพ โดยรองรับผู้ใช้งานพร้อมกันไม่น้อยกว่า ๔,๐๐๐ ราย พร้อมมีการตรวจสอบสถานะของเซิร์ฟเวอร์ (Health Check) และสามารถขยายขนาดระบบได้อัตโนมัติเมื่อตรวจพบปริมาณงานที่เพิ่มขึ้น

๕.๘. ผู้รับจ้างต้องพัฒนาระบบให้สามารถในการบันทึกและวิเคราะห์ข้อมูล Log การทำงานของระบบแบบเรียลไทม์ และสามารถแจ้งเตือนเมื่อระบบมีปัญหา โดยต้องสามารถค้นหาข้อมูลย้อนหลังได้

๕.๙. ผู้รับจ้างต้องพัฒนาระบบให้สามารถรองรับการขยายขนาดได้อัตโนมัติ (Auto Scaling) โดยต้องสามารถปรับเพิ่มหรือลดจำนวนเซิร์ฟเวอร์ได้ตามปริมาณงานที่เปลี่ยนแปลงไปและสามารถรองรับปริมาณการใช้งานที่เพิ่มขึ้นได้โดยไม่มี Downtime

๕.๑๐. ผู้รับจ้างจะต้องดำเนินการพัฒนาและจัดทำระบบให้เป็นไปตามแนวทางหรือข้อเสนอแนะด้านความปลอดภัยที่เป็นที่ยอมรับในระดับสากล เช่น แนวทางจาก OWASP (เช่น OWASP Top Ten, OWASP API Security, หรือ OWASP Application Security Guidelines) หรือแนวทางอื่นที่เทียบเท่า ทั้งนี้เพื่อ

ป้องกันช่องโหว่ที่อาจส่งผลกระทบต่อความมั่นคงปลอดภัยของระบบ โดยผู้รับจ้างจะต้องติดตามและปรับปรุงให้สอดคล้องกับแนวทางที่อัปเดตในอนาคตอย่างเหมาะสม

๕.๑๑. ผู้รับจ้างต้องทำ Unit Test สำหรับระบบที่พัฒนาขึ้นใหม่ ให้ครอบคลุมฟังก์ชันหลักทุกตัวอย่างน้อย ๘๐% และสามารถใช้เครื่องมือที่สามารถแสดงในรูปแบบที่ตรวจสอบย้อนหลังได้

๕.๑๒. กรณีระบบที่พัฒนาขึ้นใหม่ผู้รับจ้างต้องทำการประเมินและแก้ไขช่องโหว่ (Mitigation Criteria) และสามารถใช้เครื่องมือที่สามารถแสดงในรูปแบบที่ตรวจสอบย้อนหลังได้ โดยต้องดำเนินการอย่างน้อยดังนี้

๕.๑๒.๑. ผู้รับจ้างต้องดำเนินการตรวจสอบและ แก้ไขช่องโหว่ทั้งหมด ที่อยู่ในระดับ Critical และ High Severity ตามนิยามของเครื่องมือที่เลือกใช้

๕.๑๒.๒. ช่องโหว่ระดับต่ำกว่า (Medium/Low/Info) ต้องมีการจัดทำแผนการแก้ไข หรือเอกสารแสดงเหตุผลในการยอมรับความเสี่ยง (Risk Acceptance)

๕.๑๒.๓. เกณฑ์คุณภาพของโค้ดหลังการปรับปรุง ระบบต้องมีผลการวิเคราะห์โค้ดผ่านเกณฑ์ (Pass Quality Criteria) โดยมีค่าขั้นต่ำดังนี้

๕.๑๒.๓.๑. ช่องโหว่ระดับ Critical และ High ไม่มีคงค้าง

๕.๑๒.๓.๒. ระดับความสามารถในการบำรุงรักษา (Maintainability) และ ความน่าเชื่อถือ (Reliability) อยู่ในระดับ A หรือเทียบเท่า

๕.๑๒.๓.๓. Code Coverage (จาก Unit Test) ไม่ต่ำกว่า ๘๐% ของโค้ดที่เพิ่มหรือเปลี่ยนแปลง

๕.๑๓. ผู้รับจ้างต้องออกแบบและพัฒนาระบบ Caching เพื่อช่วยเพิ่มประสิทธิภาพในการเข้าถึงข้อมูลของระบบ โดยกลยุทธ์ที่เลือกใช้ต้องมีความเหมาะสมกับลักษณะข้อมูลและรูปแบบการเข้าถึง (Read/Write Pattern) และสามารถควบคุมความสอดคล้องของข้อมูล (Data Consistency) ได้อย่างเหมาะสม โดยมีข้อกำหนดอย่างน้อยดังนี้

๕.๑๓.๑. กลยุทธ์ฝั่ง Read (Read Caching Strategies)

๕.๑๓.๑.๑. Cache Aside

๑) แอปพลิเคชันตรวจสอบ Cache ก่อน หากไม่พบ (cache miss) จึงไปดึงข้อมูลจากฐานข้อมูล และทำการอัปเดต Cache

๒) เหมาะกับข้อมูลที่ไม่เปลี่ยนแปลงบ่อย

๕.๑๓.๑.๒. Read Through

๑) การอ่านทั้งหมดทำผ่าน Cache Layer โดย Cache จะทำหน้าที่โหลดข้อมูลจากฐานข้อมูลให้เองเมื่อเกิด cache miss

๒) เหมาะกับการใช้งานแบบอ่านบ่อย (read-heavy)

๕.๑๓.๒. กลยุทธ์ฝั่ง Write (Write Caching Strategies)

๕.๑๓.๒.๑. Write Around

๑) เขียนข้อมูลลงฐานข้อมูลโดยตรง โดยไม่อัปเดต Cache

๒) เหมาะกับข้อมูลที่ไม่ค่อยถูกอ่านหลังจากเขียน

๕.๑๓.๒.๒. Write Through

๑) เขียนข้อมูลลงทั้ง Cache และฐานข้อมูลพร้อมกัน

๒) เหมาะกับข้อมูลที่ต้องการความสอดคล้องสูงทันทีหลังการเขียน

๕.๑๓.๒.๓. Write Back

๑) เขียนข้อมูลลง Cache ก่อน แล้วค่อยเขียนลงฐานข้อมูลภายหลังเป็นรอบ (batch/interval)

๒) เหมาะกับระบบที่เน้นประสิทธิภาพการเขียนและสามารถยอมรับความหน่วงเวลาในการ Persist ข้อมูลได้

๕.๑๓.๓. ข้อกำหนดทั่วไป โดยมีข้อกำหนดอย่างน้อยดังต่อไปนี้

๕.๑๓.๓.๑. กำหนดค่าการจัดการอายุของข้อมูลในแคช (Time to Live - TTL) โดยผู้รับจ้างต้องสามารถกำหนดช่วงเวลาที่ข้อมูลในแคชจะมีอายุและหมดอายุอัตโนมัติได้ เพื่อป้องกันการใช้ข้อมูลที่ล้าสมัย

๕.๑๓.๓.๒. กำหนดกลยุทธ์การล้างข้อมูลในแคช (Invalidation Strategy) และการ Refresh ข้อมูล โดยระบบต้องสามารถล้างหรือทำให้ข้อมูลในแคชไม่ถูกต้องเมื่อมีการอัปเดตจากฐานข้อมูลต้นทาง และสามารถโหลดข้อมูลใหม่เข้ามาทดแทนโดยอัตโนมัติ

๕.๑๓.๓.๓. มีการจัดการกรณีที่ข้อมูลไม่อยู่ในแคช (Cache Miss) เมื่อไม่มีข้อมูลในแคช ระบบต้องสามารถเรียกข้อมูลจากแหล่งที่มา เช่น ฐานข้อมูล และนำกลับมาเก็บในแคชอย่างเหมาะสม

๕.๑๓.๓.๔. ปัญหาข้อมูลไม่ตรงกันระหว่างแคชกับฐานข้อมูล (Cache Inconsistency) โดยผู้รับจ้างต้องออกแบบกลไกให้สามารถตรวจจับและแก้ไขข้อมูลแคชที่ไม่ตรงกับฐานข้อมูลได้ เพื่อความถูกต้องของข้อมูลที่นำมาแสดง

๕.๑๓.๓.๕. สามารถดำเนินการกู้คืนข้อมูลในกรณีที่เกิดปัญหา (Recovery)ระบบต้องสามารถฟื้นฟูข้อมูลจากแคช หรือเรียกข้อมูลจากแหล่งต้นทางในกรณีที่ระบบแคชล่มหรือสูญหาย

๕.๑๓.๓.๖. สามารถเลือกใช้กลยุทธ์การแคชแบบผสมผสานตามความเหมาะสม เช่น Cache Aside และ Write Through

๑) Cache Aside: ดึงข้อมูลจากฐานข้อมูลเมื่อไม่มีในแคช แล้วบันทึกเข้าแคชภายหลัง

๒) Write Through: บันทึกข้อมูลเข้าสู่แคชและฐานข้อมูลพร้อมกันทุกครั้งที่มีการอัปเดต

๕.๑๓.๓.๗. หากใช้เครื่องมือจัดการแคช เช่น Redis หรือเทียบเท่า ต้องมีคุณสมบัติอย่างน้อยดังต่อไปนี้

๑) รองรับการทำงานแบบ Distributed Cache เพื่อกระจายข้อมูลและโหลดการใช้งาน

คณพศ หงสวรางกูร ปริญญา เพ็ญไพรัตน์กุล ธีรยุทธ วัฒนะธรรมศิริ ศิร์รัฐ ศรีสุนทรพินิต สาลิเดช สัมฤทธิ์
มธุรส ไพบูลย์วรชาติ วิมลมาศ วัฒนจรูญโรจน์ รัฐธรรม ชวนเชย พันธุ์นิชิต โปษยานนท์ นัฐพล สีสวย
วิชาญ จินดารัตน์ สิทธิศักดิ์ ไชยสุข อำนาจวิทย์ หมู่ศิลป์ จุฑามาศ วงศ์เหมอนันต์ ทักษิณ หลังสตา

๒) รองรับ High Availability (HA) เพื่อให้บริการได้อย่างต่อเนื่องแม้มี Node บางส่วนขัดข้อง

๕.๑๔. ผู้รับจ้างต้องพัฒนาระบบให้เป็นไปตามข้อ ๓ การออกแบบ

๖. การทดสอบ (Testing)

๖.๑. Integration Testing

๖.๑.๑. ระบบจะต้องถูกทดสอบการทำงานร่วมกันระหว่างโมดูลต่าง ๆ เช่น จากโมดูลจัดการผู้ใช้ไปยังระบบแจ้งเตือน และต่อเนื่องไปยังการส่งอีเมล เพื่อให้มั่นใจว่าข้อมูลสามารถส่งต่อกันได้ถูกต้องและเป็นไปตามลำดับการใช้งานจริง และจัดทำรายงานผลการทดสอบประกอบการส่งมอบระบบ

๖.๑.๒. ระบบจะต้องถูกตรวจสอบตามรายละเอียดคุณลักษณะเฉพาะตามภาคผนวก ก รายละเอียดคุณลักษณะของระบบสารสนเทศของโครงการ

๖.๒. System Testing

๖.๒.๑. ผู้พัฒนาต้องทำการทดสอบระบบทั้งหมดจากต้นทางถึงปลายทางตามกรณีใช้งานที่กำหนด (End-to-End Test) และจัดทำรายงานผลการทดสอบประกอบการส่งมอบระบบ

๖.๒.๒. ระบบจะต้องถูกตรวจสอบตามรายละเอียดคุณลักษณะเฉพาะตามภาคผนวก ก รายละเอียดคุณลักษณะของระบบสารสนเทศของโครงการ

๖.๓. Automated Testing

๖.๓.๑. ต้องสามารถใช้เครื่องมือในการทดสอบอัตโนมัติ (Automated Testing Tools) ได้อย่างน้อย ๗๐% ของจำนวน Test Case ทั้งหมด โดยต้องสามารถแสดงผลการทดสอบย้อนหลังได้

๖.๔. User Acceptance Testing (UAT)

๖.๔.๑. ต้องมีการดำเนินการ UAT ร่วมกับเจ้าหน้าที่หรือผู้ใช้งานปลายทางของหน่วยงาน โดยใช้กรณีทดสอบที่ผ่านการอนุมัติจากผู้ว่าจ้าง และต้องจัดทำรายงานข้อเสนอแนะจากผู้ใช้ปลายทาง

๖.๕. Performance Testing / Load Testing

ระบบต้องสามารถรองรับผู้ใช้งานพร้อมกันได้ไม่น้อยกว่า ๑,๐๐๐ คน โดย Response Time ต้องไม่เกิน ๕ วินาทีต่อคำขอ โดยต้องแนบผลการทดสอบ Performance ด้วยเครื่องมือที่ได้มาตรฐาน (เช่น JMeter หรือเทียบเท่า)

๖.๖. Security Testing

๖.๖.๑. ผู้รับจ้างต้องทำการทดสอบความปลอดภัยของระบบ เช่น การโจมตีแบบ SQL Injection, Cross-site Scripting (XSS), Broken Access Control และต้องแนบรายงาน Security Testing ที่มีผลเป็น “ผ่าน” ก่อนการส่งมอบ

๖.๖.๒. ผู้รับจ้างต้องทดสอบด้านความปลอดภัยให้เป็นไปตามข้อ ๕.๑๐

๖.๗. Backup & Recovery Testing

๖.๗.๑. ผู้รับจ้างต้องทำการทดสอบการ Backup และ Restore ข้อมูลระบบโดยไม่ทำให้ข้อมูลสูญหาย

๖.๘. Cross-Browser & Cross-Device Testing ระบบต้องสามารถใช้งานได้ปกติบน Web Browser อย่างน้อย Chrome, Firefox, Safari และ Edge รวมถึงสามารถใช้งานได้ทั้งบนเครื่องคอมพิวเตอร์ และ Mobile (iOS และ Android)

๖.๙. กรณีที่ซอฟต์แวร์ที่นำมาใช้งานร่วมกันกับระบบที่พัฒนาขึ้นใหม่เป็นซอฟต์แวร์สำเร็จรูป (Commercial Off-The-Shelf - COTS) ผู้รับจ้างต้องดำเนินการทดสอบระบบตามความเหมาะสมเท่าที่สามารถดำเนินการได้ โดยมีรายละเอียดการทดสอบที่ต้องดำเนินการ ดังนี้

๖.๙.๑. Configuration Testing ต้องทดสอบการตั้งค่าหรือกำหนดค่าต่าง ๆ ที่ปรับแต่งให้เหมาะกับการใช้งานในหน่วยงาน เช่น สิทธิ์ผู้ใช้, ภาษา, ฟังก์ชันเฉพาะ ฯลฯ เพื่อยืนยันว่าการกำหนดค่านั้นถูกต้องและไม่ทำให้ระบบทำงานผิดพลาด

๖.๙.๒. Integration Testing ต้องทดสอบการเชื่อมโยงกับระบบภายนอก เช่น ระบบบัญชี, ระบบบุคลากร, Email Gateway, LDAP หรือ SSO โดยต้องยืนยันว่าข้อมูลส่งต่อกันได้ถูกต้องและครบถ้วน

๖.๙.๓. User Acceptance Testing (UAT) ผู้รับจ้างต้องจัดทำกรณีทดสอบ (Test Case) ตาม Use Case ที่ตกลงร่วมกับผู้ว่าจ้าง และดำเนินการ UAT ร่วมกับผู้ใช้งานปลายทาง โดยต้องจัดทำรายงานผลทดสอบและข้อเสนอแนะจากผู้ใช้ปลายทาง

๖.๙.๔. Performance Testing (เฉพาะในส่วนที่สามารถวัดผลได้) หากระบบอนุญาตให้ทดสอบ Performance ได้ เช่น ผ่าน API หรือโหลดหน้าเว็บ สามารถทดสอบ Response Time และความสามารถในการรองรับผู้ใช้งานพร้อมกันได้ตามข้อตกลง

๖.๙.๕. Security & Access Control Testing ทดสอบสิทธิ์การเข้าถึงของผู้ใช้งานประเภทต่าง ๆ ตามที่กำหนด (Role-based Access Control) เพื่อยืนยันว่าไม่มีการเข้าถึงข้อมูลโดยไม่ได้รับอนุญาตหากระบบมีช่องให้สื่อสารข้อมูลหรือรับอินพุต ให้ทดสอบความเสี่ยงพื้นฐาน เช่น การป้อนข้อมูลผิดรูปแบบ, การเปลี่ยน URL

๖.๙.๖. Logging & Audit Trail Verification ตรวจสอบว่า ระบบมีการบันทึกการใช้งานของผู้ใช้ (User Activity Log) และสามารถตรวจสอบย้อนหลังได้ตามนโยบายของหน่วยงาน

ในกรณีที่ซอฟต์แวร์สำเร็จรูปไม่อนุญาตให้ทดสอบบางประเภท เช่น Penetration Test หรือ Load Test ที่อาจส่งผลกระทบต่อระบบ ผู้รับจ้างต้องแจ้งให้ผู้ว่าจ้างทราบล่วงหน้า และระบุข้อจำกัดของการทดสอบอย่างชัดเจน พร้อมเอกสารรับรองจากผู้ผลิตหรือตัวแทนจำหน่าย (ถ้ามี)

๖.๑๐. ผู้รับจ้างต้องดำเนินการในส่วนที่เกี่ยวข้องกับการการทดสอบ (Testing)ตามข้อ ๓ การออกแบบ

๗. การปรับใช้ (Deployment)

๗.๑. ผู้รับจ้างจะต้องดำเนินการปรับใช้ (Deployment) ระบบในแต่ละสภาพแวดล้อมที่แยกจากกัน (Environment Separation) ดังนี้

๗.๑.๑. Dev (Development Environment) สำหรับนักพัฒนาใช้พัฒนาและทดลองฟังก์ชันพื้นฐาน

๗.๑.๒. UAT (User Acceptance Test Environment) สำหรับผู้ใช้งานปลายทาง/เจ้าของระบบทำการตรวจสอบความถูกต้องของระบบ

๗.๑.๓. PROD (Production Environment) สำหรับการให้บริการจริง

๗.๒. การปรับใช้ในแต่ละ Environment ต้องใช้กระบวนการแบบอัตโนมัติ (Automated Deployment) ผ่าน Pipeline ที่ควบคุมด้วยระบบเวอร์ชัน เช่น Git หรือเทียบเท่า และต้องมีการควบคุมการเปลี่ยนแปลง (Change Control) ที่ตรวจสอบย้อนหลังได้

๗.๓. ผู้รับจ้างจะต้องจัดเตรียมและปรับใช้ระบบในสภาพแวดล้อมแยกออกจากกันโดยสิ้นเชิง ได้แก่ Dev, UAT และ Production ตามลำดับ เพื่อรองรับการพัฒนา ทดสอบ และการใช้งานจริง

๗.๔. กำหนดสิทธิ์การเข้าถึงแต่ละ Environment

๗.๕. ต้องเปิดใช้งาน Logging และ Monitoring อย่างเหมาะสม

๗.๖. ต้องสามารถ Rollback หากการ Deploy ล้มเหลว

๗.๗. ปรับปรุงการนำระบบไปใช้งาน โดยต้องไม่มีผลกระทบต่อการให้บริการ และไม่มีช่วงเวลาหยุดทำงาน (Zero Downtime Deployment)

๗.๘. ผู้รับจ้างต้องดำเนินการในส่วนที่เกี่ยวข้องกับการปรับใช้ (Deployment) ตามข้อ ๓ การออกแบบ

๘. การควบคุมการเปลี่ยนแปลง (Change Control)

สำหรับควบคุมการปรับแก้ไขระบบ หรือ Deployment ระบบใน Production Environment รายละเอียด ดังนี้

๘.๑. ขอบเขตของการเปลี่ยนแปลง (Scope of Change)

๘.๑.๑. เหตุผลในการเปลี่ยนแปลง (Rationale)

๘.๑.๒. ผลกระทบที่อาจเกิดขึ้น (Impact Assessment)

๘.๑.๓. แผนการสำรองข้อมูลและการกู้คืน (Backup & Rollback Plan)

๘.๑.๔. แผนทดสอบหลังการเปลี่ยนแปลง (Post-Deployment Testing Plan)

๘.๒. แผนดังกล่าวต้องนำเสนอต่อผู้ว่าจ้าง เพื่อรับทราบและอนุมัติก่อนดำเนินการทุกครั้ง เพื่อให้การดำเนินงานเป็นไปตามมาตรฐานการควบคุมการเปลี่ยนแปลง และลดความเสี่ยงต่อการหยุดชะงักของบริการ

๘.๓. กระบวนการควบคุมการเปลี่ยนแปลงต้องดำเนินการภายใต้มาตรฐานสากลที่ได้รับการยอมรับ ได้แก่ ITIL: แนวปฏิบัติที่ดีที่สุดในการบริหารจัดการบริการ IT โดยเฉพาะกระบวนการ Change Management,

COBIT: กรอบการกำกับดูแล IT ที่เน้นการควบคุมและความสอดคล้องกับเป้าหมายองค์กร, ISO/IEC ๒๐๐๐๐: มาตรฐานระบบบริหารจัดการบริการ IT ที่กำหนดให้มีการควบคุมการเปลี่ยนแปลงอย่างเป็นระบบ

๙. การซ่อมบำรุง (Maintenance)

๙.๑. ผู้รับจ้างจะต้องดำเนินการซ่อมบำรุงระบบ (Maintenance) เพื่อให้ระบบสามารถทำงานได้อย่างต่อเนื่อง มีเสถียรภาพ และสอดคล้องกับความต้องการของสำนักงานการตรวจเงินแผ่นดิน ภายหลังจากการส่งมอบระบบ โดยขอบเขตงานซ่อมบำรุงต้องครอบคลุมรายละเอียด ดังนี้

๙.๑.๑. ผู้รับจ้างต้องมีแผนการตรวจสอบและบำรุงรักษาระบบเป็นระยะ เช่น การเคลียร์ log ตรวจสอบฐานข้อมูล ตรวจสอบสถานะของระบบที่สำคัญ และการอัปเดตความปลอดภัย (Security Patch)

๙.๑.๒. ต้องมีการจัดทำบันทึกการดำเนินการบำรุงรักษา (Maintenance Record)

๙.๑.๓. ผู้รับจ้างต้องแก้ไขข้อผิดพลาดหรือบั๊ก (Bug) ของระบบที่เกิดขึ้นหลังการส่งมอบ โดยไม่คิดค่าใช้จ่ายเพิ่มเติมภายในระยะเวลารับประกัน

๙.๑.๔. กรณีระบบต้องปรับให้เข้ากับสภาพแวดล้อมที่เปลี่ยนไป เช่น เปลี่ยน API ของระบบภายนอก ปรับให้รองรับ Web Browser เวอร์ชันใหม่ ฯลฯ ผู้รับจ้างต้องดำเนินการให้อยู่ในขอบเขตของระบบที่ส่งมอบไว้เดิม

๙.๑.๕. ผู้รับจ้างต้องจัดให้มีช่องทางสำหรับการแจ้งปัญหา เช่น ผ่านระบบ Ticket, Email, Webform หรือเทียบเท่า

๙.๑.๖. ต้องมีระบบติดตามสถานะของปัญหา และจัดลำดับความสำคัญ (Priority) ได้

๙.๑.๗. ต้องให้บริการซ่อมบำรุงภายหลังการส่งมอบระบบ เป็นระยะเวลาไม่น้อยกว่า ๑๒ เดือน

๙.๑.๘. SLA อ้างอิงตาม ขอบเขตของงาน โครงการพัฒนาแพลตฟอร์มการตรวจเงินแผ่นดิน หัวข้อ ๑๒ การกำหนดระยะเวลารับประกันความชำรุดบกพร่อง

๙.๑.๙. ผู้รับจ้างต้องจัดทำ รายงานซ่อมบำรุงรายเดือน โดยระบุรายการปัญหาที่พบ วิธีการแก้ไข และผลการดำเนินงาน

๙.๑.๑๐. รายงานต้องสามารถตรวจสอบย้อนหลังได้ และอยู่ในรูปแบบที่ผู้ว่าจ้างสามารถตรวจสอบได้ง่าย เช่น PDF หรือเอกสารอิเล็กทรอนิกส์

๙.๑.๑๑. ผู้รับจ้างต้องรับประกันความสมบูรณ์ของระบบหลังการส่งมอบ ตามระยะเวลาที่กำหนด โดยความผิดพลาดที่เกิดจากระบบหรือการเขียนโปรแกรมต้องแก้ไขโดยไม่มีค่าใช้จ่าย

๙.๑.๑๒. การรับประกันต้องรวมถึงฐานข้อมูล ระบบสำรองข้อมูล และการสื่อสารภายในระบบ

๙.๑.๑๓. ผู้รับจ้างต้องสามารถให้การสนับสนุนกรณีฉุกเฉินได้ตามที่ตกลง เช่น ผ่านช่องทางโทรศัพท์หรือวิดีโอคอล โดยเฉพาะในเวลาทำการ หรือ ตามที่โครงการกำหนด

๙.๑.๑๔. การควบคุมการเปลี่ยนแปลง (Change Control) ก่อนการปรับแก้ไขระบบหรือ Deployment ขึ้นสู่ Production Environment ผู้รับจ้างต้องจัดทำแผนการเปลี่ยนแปลง (Change Plan) โดยอย่างน้อยต้องระบุดังนี้

๙.๑.๑๔.๑. ขอบเขตของการเปลี่ยนแปลง (Scope of Change)

๙.๑.๑๔.๒. เหตุผลในการเปลี่ยนแปลง (Rationale)

๙.๑.๑๔.๓. ผลกระทบที่อาจเกิดขึ้น (Impact Assessment)

๙.๑.๑๔.๔. แผนการสำรองข้อมูลและการกู้คืน (Backup & Rollback Plan)

๙.๑.๑๔.๕. แผนทดสอบหลังการเปลี่ยนแปลง (Post-Deployment Testing Plan)

แผนดังกล่าวต้องนำเสนอต่อผู้ว่าจ้าง เพื่อรับทราบและอนุมัติก่อนดำเนินการทุกครั้ง เพื่อให้การดำเนินงานเป็นไปตามมาตรฐานการควบคุมการเปลี่ยนแปลง และลดความเสี่ยงต่อการหยุดชะงักของบริการ

๙.๑.๑๕. การมอนิเตอร์ระบบ (System Monitoring) ผู้รับจ้างต้องจัดให้มีการมอนิเตอร์การทำงานของระบบอย่างต่อเนื่องตลอดระยะเวลาโครงการ ครอบคลุมการตรวจสอบดังนี้

๙.๑.๑๕.๑. สถานะระบบ (System Status) โดยมีรายละเอียดอย่างน้อยดังนี้

๑) สถานะ ของ Service/Component ต่าง ๆ

๒) สถานะ CPU

๓) สถานะ Memory

๔) สถานะ Disk แต่ละ Partition

๙.๑.๑๕.๒. ประสิทธิภาพ (Performance)

๙.๑.๑๕.๓. ความผิดปกติ (Anomalies)

๙.๑.๑๕.๔. เหตุการณ์ด้านความปลอดภัย (Security Incidents)

๙.๑.๑๖. การจัดการเหตุการณ์ผิดปกติ (Incident Management) เมื่อผู้รับจ้างตรวจพบเหตุการณ์ผิดปกติหรือสัญญาณเตือน (Alert) เช่น ความล้มเหลวของบริการ (Service Failure) การใช้ทรัพยากรเกินเกณฑ์ (Resource Utilization Overload) ค่าประสิทธิภาพตกต่ำ (Performance Degradation) หรือสัญญาณความเสี่ยงด้านความปลอดภัย (Security Incidents) ผู้รับจ้างต้องดำเนินการดังนี้

๙.๑.๑๖.๑. ตอบสนองและแก้ไขเบื้องต้น (Immediate Response) ภายในขอบเขตที่ควบคุมได้โดยไม่รออนุมัติ เพื่อป้องกันความเสียหายขยายวงกว้าง

๙.๑.๑๖.๒. แจ้งเหตุการณ์ต่อผู้ว่าจ้างโดยเร็ว

๙.๑.๑๖.๓. จัดทำรายงานสรุปเหตุการณ์ (Incident Report) ซึ่งรวมถึงสิ่งที่ตรวจพบ การดำเนินการแก้ไข และข้อเสนอแนะการป้องกันเหตุการณ์ซ้ำซ้อน (Preventive Recommendations)

๙.๑.๑๗. การจัดการปัญหาด้านประสิทธิภาพ (Performance Issue Management) ในกรณีที่พบว่าค่าประสิทธิภาพเกินกว่าเกณฑ์ เช่น Response Time เกินค่าที่กำหนด Error Rate สูงกว่ามาตรฐาน หรือ Resource Utilization สูงผิดปกติ ผู้รับจ้างต้องดำเนินการดังนี้

๙.๑.๑๗.๑. จัดทำแผนการแก้ไขปัญหา (Remediation Plan) ประกอบอย่างน้อยดังนี้

๑) สาเหตุของปัญหา (Root Cause)

๒) แนวทางการแก้ไข (Proposed Solution)

๓) ระยะเวลาในการดำเนินการ (Proposed Timeline)

๔) ผลกระทบที่อาจเกิดขึ้น (Impact)

๙.๑.๑๗.๒. เสนอต่อผู้ว่าจ้างเพื่อพิจารณาและอนุมัติก่อนดำเนินการแก้ไข

๙.๑.๑๘. ระยะเวลาในการแก้ไข (Remediation Timeline) การกำหนดระยะเวลาในการแก้ไขต้องเหมาะสมและเป็นธรรม โดยคำนึงถึงความเร่งด่วน ความสำคัญของปัญหาต่อการให้บริการ และข้อจำกัดทรัพยากรของผู้รับจ้าง ทั้งนี้ ผู้ว่าจ้างมีสิทธิร่วมพิจารณาและตกลงยอมรับระยะเวลาที่เสนอมา และการดำเนินการต้องไม่กระทบต่อความต่อเนื่องของบริการ

๙.๑.๑๙. การรายงานผลการแก้ไข (Remediation Reporting) หลังเสร็จสิ้นการแก้ไขปัญหา ผู้รับจ้างต้องจัดทำ รายงานสรุปผลการแก้ไข (Remediation Report) เพื่อส่งให้ผู้ว่าจ้างสำหรับตรวจสอบและยืนยันผลการแก้ไข รวมถึงใช้ประกอบการรับรองคุณภาพและความต่อเนื่องของการให้บริการ