

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 และกระบวนการทำงานที่เกี่ยวข้องสำหรับการพัฒนาระบบคอมพิวเตอร์ปัญญาประดิษฐ์เพื่อเพิ่มประสิทธิภาพการตรวจเงินแผ่นดินและแหล่งข้อมูลอื่นที่น่าเชื่อถือได้เพื่อใช้ร่วมในการวิเคราะห์และออกแบบระบบได้อย่างครบถ้วนถูกต้องพร้อมทั้งสรุปเป็นเอกสารข้อกำหนดความต้องการระบบ

๒.๒ ผู้รับจ้างต้องจัดทำเอกสารข้อกำหนดความต้องการระบบ (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

๒.๒.๓ ข้อกำหนดที่ไม่ใช่เชิงหน้าที่ (Non-Functional Requirements: NFRs) ผู้รับจ้างต้องวิเคราะห์ กำหนด และจัดทำข้อกำหนดที่ไม่ใช่เชิงหน้าที่ของระบบ โดยอย่างน้อยต้องครอบคลุมด้านความพร้อมใช้งาน (Availability) ความสามารถในการขยายระบบ (Scalability) ประสิทธิภาพ (Performance) ความสามารถในการบำรุงรักษา (Maintainability) ความน่าเชื่อถือและความทนทานของระบบ (Reliability and Resilience) ความมั่นคงปลอดภัย (Security) การสำรองและกู้คืนระบบ (Backup and Recovery) การ

ตรวจสอบและติดตามระบบ (Observability) ความสามารถในการทำงานร่วมกับระบบอื่น (Interoperability) และความสามารถในการรองรับการเปลี่ยนแปลงหรือขยายความสามารถของระบบ (Extensibility)

ผู้รับจ้างต้องจัดทำ Non-Functional Requirements Specification โดยกำหนดค่าเป้าหมาย ตัวชี้วัด วิธีวัด วิธีทดสอบ และเกณฑ์ผ่านของแต่ละข้อให้ชัดเจน และต้องเชื่อมโยงข้อกำหนดดังกล่าวกับ Requirement Traceability Matrix (RTM) ตามข้อ 2.3.4

รายละเอียดด้านการออกแบบและการทดสอบให้เป็นไปตามข้อกำหนดที่เกี่ยวข้องในภาคผนวกนี้ ตามที่เกี่ยวข้อง

สำหรับข้อกำหนดที่เกี่ยวข้องกับองค์ประกอบด้านปัญญาประดิษฐ์ ให้สอดคล้องกับข้อกำหนดเฉพาะด้านปัญญาประดิษฐ์ตามภาคผนวก ง ตามที่เกี่ยวข้อง

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

๒.๓.๑. ผู้รับจ้างต้องจัดทำ Prototype ของระบบ โดยจัดทำรูปแบบไฟล์แก้ไขได้ เช่น Figma หรือเทียบเท่า

๒.๓.๒. ผู้รับจ้างต้องจัดทำเอกสารสถาปัตยกรรมระบบ (System Architecture Document) และแผนภาพสถาปัตยกรรม (Architecture Diagrams) เพื่อแสดงภาพรวม โครงสร้าง องค์ประกอบ ความสัมพันธ์ระหว่างองค์ประกอบ และการเชื่อมต่อหรือแลกเปลี่ยนข้อมูลภายในและภายนอก ระบบ โดยสถาปัตยกรรมต้องสอดคล้องกับข้อกำหนดด้านความพร้อมใช้งานสูง (High Availability) ความสามารถในการขยายระบบ (Scalability) ความมั่นคงและทนทานของระบบ (Reliability and Resilience) ความมั่นคงปลอดภัย (Security) และข้อกำหนดด้านเทคนิคที่เกี่ยวข้องตามภาคผนวกนี้ เอกสาร และแผนภาพดังกล่าวอย่างน้อยต้องประกอบด้วย

๒.๓.๒.๑. System Context Diagram เพื่อแสดงขอบเขตของระบบ ผู้ใช้งาน ระบบภายในและภายนอก ตลอดจนความสัมพันธ์และการเชื่อมต่อที่เกี่ยวข้อง

๒.๓.๒.๒. Reference Architecture เพื่อแสดงแนวทางและโครงสร้าง สถาปัตยกรรมอ้างอิง องค์ประกอบหลัก และหลักการที่ใช้เป็นกรอบในการออกแบบและพัฒนาระบบ

๒.๓.๒.๓. Layered Architecture Diagram เพื่อแสดงการแบ่งชั้นและความสัมพันธ์ขององค์ประกอบหลักของระบบ เช่น ชั้นผู้ใช้งาน ชั้นแอปพลิเคชันและบริการ ชั้นข้อมูล ชั้นการประมวลผลและปัญญาประดิษฐ์ และชั้นโครงสร้างพื้นฐาน ตามสถาปัตยกรรมที่นำมาใช้จริง

๒.๓.๒.๔. Integration Architecture Diagram เพื่อแสดงการเชื่อมโยงและแลกเปลี่ยนข้อมูลระหว่างองค์ประกอบภายในระบบ ระบบของสำนักงานการตรวจเงินแผ่นดิน ระบบภายนอก และบริการที่เกี่ยวข้อง ผ่าน API, Event, Message หรือกลไกอื่นตามสถาปัตยกรรมที่ออกแบบ

๒.๓.๒.๕. Deployment Architecture Diagram เพื่อแสดงการจัดวางและติดตั้งองค์ประกอบของระบบบนโครงสร้างพื้นฐานในแต่ละสภาพแวดล้อมที่เกี่ยวข้อง เช่น Development (Dev), System Integration Testing (SIT), User Acceptance Testing (UAT) และ Production รวมถึงองค์ประกอบด้าน Network, Security, Load Balancing, Compute, Storage และองค์ประกอบอื่นตามสถาปัตยกรรมที่นำมาใช้จริง ตลอดจนการออกแบบที่รองรับ High Availability, Scalability, Failover และ Recovery ตามความเหมาะสม

ผู้รับจ้างต้องจัดทำคำอธิบายประกอบแผนภาพ โดยระบุหน้าที่ ความสัมพันธ์ Interface และหลักการหรือเหตุผลในการออกแบบขององค์ประกอบสำคัญ เพื่อให้เอกสารดังกล่าวสามารถใช้เป็นข้อมูลอ้างอิงสำหรับการพัฒนา การติดตั้ง การทดสอบ การบำรุงรักษา การแก้ไขปรับปรุง และการขยายระบบในอนาคตได้

สำหรับ องค์ ประกอบ และ กระบวนการ ด้าน ปัญญาประดิษฐ์ สถาปัตยกรรมระบบตามข้อนี้ต้องแสดงการเชื่อมโยงและความสัมพันธ์กับสถาปัตยกรรมการประมวลผลปัญญาประดิษฐ์ และต้องสอดคล้องกับข้อกำหนดที่กำหนดไว้ในภาคผนวก ง การพัฒนาระบบ ปัญญาประดิษฐ์และการกำกับดูแลการใช้ปัญญาประดิษฐ์ ข้อ ๒ โดยรายละเอียดเฉพาะของ AI Model, AI Processing Pipeline, Model Serving, Environment และองค์ประกอบด้านปัญญาประดิษฐ์ ให้เป็นไปตามภาคผนวกดังกล่าว

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

๒.๓.๔. ผู้รับจ้างต้องจัดทำ Requirement Traceability Matrix (RTM) เพื่อแสดงความสัมพันธ์และการตรวจสอบย้อนกลับของข้อกำหนดตลอดวงจรการพัฒนาระบบ ตั้งแต่ข้อกำหนดของระบบ กรณีใช้งาน (Use Case) ข้อกำหนดเชิงหน้าที่ (Functional Requirement: FR) การออกแบบและองค์ประกอบที่เกี่ยวข้อง การพัฒนา การทดสอบ เกณฑ์และวิธีการตรวจรับ (Acceptance Criteria) ตลอดจนหลักฐานหรือผลการทดสอบที่ใช้ประกอบการตรวจรับ

สำหรับกรณีใช้งานหรือข้อกำหนดที่เกี่ยวข้องกับปัญญาประดิษฐ์หรือการวิเคราะห์ข้อมูล RTM ต้องสามารถเชื่อมโยงเพิ่มเติมกับข้อมูลนำเข้า (Input Data) วิธีการประมวลผลหรือการวิเคราะห์ด้วย AI/Analytics องค์ประกอบหรือเทคนิคที่ใช้ ผลลัพธ์ (Output) เกณฑ์การประเมินหรือเกณฑ์ตรวจรับ จุดตรวจทานโดยมนุษย์ (Human Review / Human-in-the-Loop) และมาตรการควบคุมหรือบริหารความเสี่ยงจากการใช้ปัญญาประดิษฐ์ (AI Risk Controls) ตามความเกี่ยวข้องกับแต่ละกรณีใช้งาน

การจัดทำ RTM ต้องครอบคลุมข้อกำหนดและกรณีใช้งานที่อยู่ในขอบเขตของโครงการ และต้องสามารถใช้เป็นเครื่องมือในการตรวจสอบความครบถ้วนและความสอดคล้องระหว่าง System Requirement Specification (SRS), Use Case, การออกแบบระบบ การพัฒนา Test Case ผลการทดสอบ และการตรวจรับระบบได้ตลอดวงจรการพัฒนา

ทั้งนี้ รูปแบบ โครงสร้าง และรายการข้อมูลของ RTM ให้เป็นไปตามภาคผนวก RTM ที่สำนักงานการตรวจเงินแผ่นดินกำหนด และสำหรับข้อกำหนดที่เกี่ยวข้องกับปัญญาประดิษฐ์ ให้ RTM เชื่อมโยงกับข้อกำหนดที่เกี่ยวข้องตามภาคผนวก ง การพัฒนาระบบปัญญาประดิษฐ์และการกำกับดูแลการใช้ปัญญาประดิษฐ์ ตามความเหมาะสม

๒.๓.๕. ผู้รับจ้างต้องจัดทำ Activity Diagram เพื่อแสดงลำดับการทำงานของแต่ละกระบวนการในระบบ ตั้งแต่การเริ่มต้นจนถึงสิ้นสุด รวมถึงการตรวจสอบเงื่อนไขและการตัดสินใจ

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

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

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

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

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 ที่แสดงขั้นตอนทั้งหมดพร้อมอธิบายวัตถุประสงค์ของแต่ละขั้น และระบุว่าตรวจสอบความถูกต้องได้อย่างไร

๒.๓.๑๓. ต้องจัดทำเอกสารแสดงแผนผังการเชื่อมโยง (Integration Architecture) กับระบบ Monitoring โดยแสดงตัวอย่างการตั้งค่าระบบ Monitoring

๒.๓.๑๔. ผู้รับจ้างต้องจัดทำเอกสารตามข้อ 2.3.2 ถึงข้อ 2.3.13 ในรูปแบบดังต่อไปนี้

๑) ไฟล์ PDF และไฟล์แก้ไขได้ (เช่น Word) หรือใช้ Web-based Documentation/ Version Control System (เช่น Confluence, GitLab Pages, MkDocs) ในการจัดเก็บและควบคุมเวอร์ชันเอกสารต้นฉบับ (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 โดยต้องระบุว่าในแต่ละกรณีสามารถทดสอบอะไรได้/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 ในกรณีที่สำนักงานการตรวจเงินแผ่นดินมีการขอแก้ไขหรือปรับปรุงการออกแบบที่ได้จัดทำและ

๒.๖ ส่งมอบแล้ว ผู้รับจ้างต้องดำเนินการแก้ไขและปรับปรุงเอกสารที่เกี่ยวข้องทั้งหมด (เช่น SRS, SDS, Wireframe, Diagram ต่าง ๆ) ให้สอดคล้องกับการแก้ไขนั้นโดยไม่คิดค่าใช้จ่ายเพิ่มเติม ทั้งนี้ การแก้ไขต้องอยู่ภายในขอบเขตงานเดิมที่ระบุไว้ใน TOR และไม่เกินกว่าขอบเขตงานเดิมหรือเงื่อนไขที่ผู้รับจ้างได้เสนอราคาไว้ หากการแก้ไขหรือปรับปรุงมีขอบเขตเกินกว่างานเดิมที่ตกลงกันไว้ หรือเป็นการเปลี่ยนแปลงสาระสำคัญของ

ระบบ ผู้ว่าจ้างและผู้รับจ้างจะต้องหารือร่วมกันเพื่อกำหนดวิธีการดำเนินงานและการชดเชยที่เป็นธรรมต่อทั้งสองฝ่าย

๒.๗ จัดทำเอกสารอธิบายสิทธิการใช้งาน (Role Metric)

๒.๘ การจัดการสิทธิการใช้งานซอฟต์แวร์และไลบรารี (Software License Compliance) เพื่อให้มั่นใจว่าการทำงานของซอฟต์แวร์ในโครงการถูกต้องตามกฎหมาย และไม่มีความเสี่ยงด้านลิขสิทธิ์ ผู้รับจ้างต้องรับรองว่าซอฟต์แวร์ ไลบรารี หรือส่วนประกอบจากบุคคลที่สามที่นำมาใช้ มีการจัดการสิทธิการใช้งานอย่างเหมาะสม โดยมีข้อกำหนดอย่างน้อยดังนี้

๒.๘.๑. ซอฟต์แวร์และไลบรารีทั้งหมดที่นำมาใช้ต้องมีสิทธิการใช้งานที่ถูกต้องตามกฎหมาย และสอดคล้องกับรูปแบบการใช้งานในโครงการ หากเป็นซอฟต์แวร์โอเพนซอร์ส ต้องใช้ License ที่ได้รับการรับรองโดย Open Source Initiative (OSI) เช่น MIT, BSD, Apache 2.0 หรือที่เทียบเท่า

๒.๘.๒. ต้องจัดทำรายการซอฟต์แวร์ ไลบรารี และส่วนประกอบที่ใช้ในงานทั้งหมดในโครงการ โดยระบุชื่อ เวอร์ชัน และประเภท License ที่เกี่ยวข้อง เพื่อใช้เป็นหลักฐานประกอบการตรวจสอบ

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

๒.๘.๔. อนุญาตให้ใช้ซอฟต์แวร์หรือไลบรารีภายใต้สัญญาอนุญาตแบบ GPL หรือ AGPL ได้ กรณีที่

๒.๘.๔.๑. การใช้งานเป็นการเรียกผ่าน API, Web Service หรือเป็นโมดูลแยกส่วน (ไม่ผนวกโค้ดเข้าในระบบหลักโดยตรง)

๒.๘.๔.๒. ไม่ส่งผลลัพธ์ให้ซอร์สโค้ดของระบบหลักต้องถูกเปิดเผยหรือกระทบสิทธิการใช้งานเชิงพาณิชย์

๒.๘.๔.๓. ผู้รับจ้างต้องจัดทำ “รายการซอฟต์แวร์/ไลบรารี” พร้อมรายละเอียดวิธีการใช้งาน และส่งให้ผู้ว่าจ้างพิจารณาและขออนุมัติล่วงหน้า

๒.๘.๕. ผู้รับจ้างต้องใช้เครื่องมือหรือกระบวนการที่เหมาะสมในการตรวจสอบและติดตามการปฏิบัติตามสิทธิการใช้งานซอฟต์แวร์ (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. 2560
- ๓.๑.๒. พระราชบัญญัติคุ้มครองข้อมูลส่วนบุคคล พ.ศ. 2562
- ๓.๑.๓. พระราชบัญญัติการรักษาความมั่นคงปลอดภัยไซเบอร์ พ.ศ. 2562
- ๓.๑.๔. พระราชบัญญัติว่าด้วยธุรกรรมทางอิเล็กทรอนิกส์ พ.ศ. 2544
- 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 ที่เกี่ยวข้องกับการพัฒนาระบบคอมพิวเตอร์ปัญญาประดิษฐ์

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
- 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 UserDetails

๓.๒.๕. แยก Entity และ Value Object อย่างชัดเจน เช่น

๓.๒.๖. Entity : มี ID และสามารถเปลี่ยนแปลงสถานะได้

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

๓.๒.๘. ใช้แนวคิด Layered Architecture โดยแบ่งระบบออกเป็น 4 ชั้น ได้แก่ Presentation, Application, Domain, Infrastructure เพื่อให้โครงสร้างมีความชัดเจนและแยกความรับผิดชอบ

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

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

๓.๒.๑๐.๑. Factory: สำหรับสร้าง Object

๓.๒.๑๐.๒. Service: สำหรับประมวลผล Logic

๓.๒.๑๐.๓. Repository: สำหรับจัดการ Persistence เช่น DB

๓.๓ ผู้รับจ้างต้องออกแบบและพัฒนาระบบโดยใช้สถาปัตยกรรมที่มีความยืดหยุ่นและสามารถขยายขนาดได้ (Scalable Architecture) ในลักษณะแบบแยกส่วน (Modular Design) เพื่อรองรับการเพิ่มจำนวนผู้ใช้งาน ปริมาณข้อมูล และภาระการประมวลผลที่เพิ่มขึ้นในอนาคต รวมทั้งสนับสนุนการพัฒนา การบำรุงรักษา และการปรับปรุงระบบได้อย่างมีประสิทธิภาพ โดยอาจใช้แนวทางการพัฒนาแบบ Microservices หรือสถาปัตยกรรมหรือเทคโนโลยีอื่นที่มีคุณสมบัติเทียบเท่าหรือเหมาะสมกับลักษณะของระบบ

สำหรับองค์ประกอบด้านปัญญาประดิษฐ์ ผู้รับจ้างต้องออกแบบให้สามารถทำงานร่วมกับสถาปัตยกรรมของระบบโดยรวมได้อย่างเหมาะสม และต้องสอดคล้องกับข้อกำหนดด้านสถาปัตยกรรมการประมวลผลปัญญาประดิษฐ์ (AI/ML Processing Architecture) ตามภาคผนวก ง ข้อ ๒ โดยรายละเอียดเฉพาะด้าน AI Model, Model Serving, AI Processing Pipeline, การบริหารจัดการกระบวนการประมวลผล และการแยกสภาพแวดล้อมสำหรับองค์ประกอบด้านปัญญาประดิษฐ์ ให้เป็นไปตามภาคผนวก ง ตามที่เกี่ยวข้อง

๓.๓.๑. ใช้ Lightweight Protocols

- ๓.๓.๑.๑. ใช้ REST, gRPC หรือ Message Queue (MQTT, Kafka) ตามความเหมาะสม
- ๓.๓.๑.๒. หลีกเลี่ยงการใช้ SOAP หรือโปรโตคอลหนักโดยไม่จำเป็น
- ๓.๓.๒. ต้องรองรับ Service Discovery
- ๓.๓.๒.๑. ต้องใช้ระบบ Service Registry เช่น Consul, Eureka หรือ Kubernetes Service
- ๓.๓.๒.๒. เพื่อให้สามารถค้นหาและเชื่อมต่อระหว่าง Services ได้แบบ Dynamic
- ๓.๓.๓. มีคุณสมบัติ Containers โดยระบบต้องออกแบบให้แต่ละ Service ย่อยถูกบรรจุและจัดการภายใน container เช่น Docker เพื่อให้สามารถนำไปติดตั้ง (Deploy) บนเครื่องเซิร์ฟเวอร์ใดก็ได้ ไม่ว่าจะเป็นสภาพแวดล้อม Dev ,SIT, UAT หรือ Production โดยไม่ต้องปรับแต่งหรือแก้ไขการตั้งค่าของเครื่องนั้น ทำให้มั่นใจได้ว่าการทำงานของ service
- ๓.๓.๔. มีคุณสมบัติการจัดการ Container หลายตัวพร้อมกัน (Orchestration) โดยมีรายละเอียดอย่างน้อยดังนี้
- ๓.๓.๔.๑. ระบบต้องสามารถจัดการ container หลายตัวพร้อมกันได้ โดยใช้เครื่องมือ Orchestration เช่น Kubernetes
- ๓.๓.๔.๒. ระบบต้องสามารถเพิ่มหรือลดจำนวน container ของบริการย่อย เช่น user-service ได้อัตโนมัติตามจำนวนผู้ใช้งานที่เพิ่มขึ้นหรือลดลง
- ๓.๓.๔.๓. ระบบต้องมีการตรวจสอบและติดตามสถานะ active user session เพื่อใช้เป็นข้อมูลประกอบการเพิ่มหรือลดจำนวน container อย่างเหมาะสม
- ๓.๓.๔.๔. ระบบต้องรองรับการปรับขนาด (horizontal scaling) แบบ dynamic โดยสามารถเพิ่มหรือลดจำนวน instance ของ service ได้อัตโนมัติตามภาระงาน (load) ที่เปลี่ยนแปลง
- ๓.๓.๔.๕. หากไม่มีผู้ใช้งาน Service ติดต่อกันนานเกิน 2 ชั่วโมง หรือระยะเวลาที่ผู้ว่าจ้างกำหนด ระบบต้องสามารถปรับจำนวน Replica ของ Service ดังกล่าวลงเป็นค่าต่ำสุด (minimum) หรือปรับลดการใช้ทรัพยากรโดยอัตโนมัติ (Scale down / Scale-to-Zero) ตามนโยบาย Resource Management เพื่อคืนทรัพยากรให้ระบบ โดยไม่กระทบต่อการทำงานของบริการอื่น

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

๓.๓.๔.๗. 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

๓.๓.๔.๘. ระยะเวลาในการปลุก service และให้พร้อมรับคำสั่งใช้งานได้ต้องไม่เกิน 10 วินาที หรือระยะเวลาที่ผู้ว่าจ้างยอมรับ โดยต้องทำได้โดยอัตโนมัติ

๓.๓.๔.๙. ระบบต้องได้รับการออกแบบให้สามารถทำงานต่อได้แม้เกิดความล้มเหลวของบริการบางส่วน (Partial Failure)

๓.๓.๔.๑๐. ในกรณีที่บริการสำคัญ เช่น user-service ล้มเหลวหรือไม่สามารถตอบสนองได้ ระบบที่เกี่ยวข้องต้องสามารถทำงานต่อไปได้โดยไม่ล้มตาม โดยทำการบันทึกข้อมูลที่จำเป็น เช่น ข้อมูลผู้ใช้งาน ไว้ในระบบ Queue ชั่วคราว

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

๓.๓.๔.๑๒. ระบบต้องมีการออกแบบกระบวนการ Retry และ Dead-letter queue (DLQ) เพื่อให้สามารถจัดการข้อมูลที่ประมวลผลล้มเหลวได้อย่างเหมาะสม

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

๑) แสดงให้เห็นว่า Service ใดถูกปรับจำนวน Replica ลงตามค่าขั้นต่ำ (minimum)

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 2 เป็น 10 ตัวในช่วงที่มีการเข้าสู่ระบบเป็นจำนวนมาก

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

๓.๓.๔.๑๕. ความปลอดภัย (Security) ระบบต้องได้รับการออกแบบโดยคำนึงถึงความปลอดภัยเป็นหลัก (Secure by Design) โดยมีข้อกำหนดอย่างน้อยดังนี้

๑) ข้อมูลที่จัดเก็บในระบบ เช่น ข้อมูลส่วนบุคคลและข้อมูลรับรอง ต้องได้รับการเข้ารหัสด้วยมาตรฐาน AES-256 หรือที่ดีกว่า

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

๓) สำหรับองค์ประกอบด้านปัญญาประดิษฐ์และการประมวลผลอัจฉริยะ ผู้รับจ้างต้องออกแบบและพัฒนาให้มีมาตรการด้านความมั่นคงปลอดภัยและการบริหารความเสี่ยงที่เหมาะสมกับประเภทของเทคโนโลยี ข้อมูล ทรัพย์สิน และระดับความเสี่ยง โดยมาตรการดังกล่าวต้องทำงานร่วมกับมาตรการด้านความมั่นคงปลอดภัยของระบบโดยรวมตามภาคผนวกนี้ และต้องเป็นไปตามข้อกำหนดด้านการบริหารความเสี่ยงและความมั่นคงปลอดภัยของระบบปัญญาประดิษฐ์ที่กำหนดไว้ใน ภาคผนวก ง ข้อ ๗ และข้อ ๘ ทั้งนี้ การออกแบบและดำเนินการอาจอ้างอิงหลักการหรือแนวทางที่เกี่ยวข้อง เช่น ISO/IEC 27001, ISO/IEC 42001, ISO/IEC 23894, NIST Artificial Intelligence Risk Management Framework (NIST AI RMF) หรือแนวทางอื่น

๔) การจัดการและคุ้มครองข้อมูลก็นำไปใช้กับองค์ประกอบด้านปัญญาประดิษฐ์ ต้องอยู่ภายใต้มาตรการด้านความมั่นคงปลอดภัยและการควบคุมข้อมูลของระบบโดยรวม และต้องสามารถกำหนดมาตรการให้เหมาะสมกับประเภทข้อมูล ระดับชั้นความลับ สิทธิของผู้ใช้งาน วัตถุประสงค์ในการประมวลผล และทรัพย์สิน โดยครอบคลุมอย่างน้อยการควบคุมการเข้าถึง การป้องกันการเปิดเผยหรือรั่วไหลของข้อมูล การรักษาความถูกต้องและความครบถ้วนของข้อมูล และการปกปิดหรือ

ลดทอนข้อมูลเมื่อมีความเหมาะสม โดยรายละเอียดด้านการปกป้องข้อมูล การปกปิดข้อมูล และการควบคุม การสืบค้นข้อมูลตามสิทธิสำหรับระบบปัญญาประดิษฐ์ ให้เป็นไปตามภาคผนวก ง ข้อ ๘.๓

๕) การเชื่อมต่อหรือเรียกใช้แบบจำลอง บริการ หรือ องค์ประกอบด้านปัญญาประดิษฐ์ ทั้งที่ให้บริการภายในและภายนอกสภาพแวดล้อมของสำนักงานการตรวจเงินแผ่นดิน ต้องอยู่ภายใต้กลไกควบคุมสิทธิ การยืนยันตัวตน การรับส่งข้อมูล การบันทึกการใช้งาน และ มาตรการด้านความมั่นคงปลอดภัยของระบบตามภาคผนวกนี้ สำหรับการส่งข้อมูลไปยังบริการ AI ภายนอก ต้องสามารถควบคุมประเภทและขอบเขตของข้อมูลที่อนุญาตให้ส่งออก และต้องไม่อนุญาตให้ส่งข้อมูลที่ไม่ เป็นไปตามเงื่อนไขหรือสิทธิที่สำนักงานการตรวจเงินแผ่นดินกำหนด ทั้งนี้ รายละเอียดด้านการใช้ Private AI/Local LLM และ External AI/External LLM การปกป้องข้อมูล การควบคุมการสืบค้นตามสิทธิ และ มาตรการด้านความมั่นคงปลอดภัยเฉพาะของ AI ให้เป็นไปตามภาคผนวก ง ข้อ ๘.๑ ข้อ ๘.๒ และข้อ ๘.๓ ตามที่เกี่ยวข้อง โดยการบันทึกการใช้งานและการตรวจสอบย้อนกลับขององค์ประกอบด้านปัญญาประดิษฐ์ ให้ เป็นไปตามข้อกำหนดด้าน AI Usage Log และ Traceability ที่กำหนดไว้ใน ภาคผนวก ง ข้อ ๕.๗ และข้อ ๕.๘ ตามที่เกี่ยวข้อง

๓.๓.๔.๑๖. 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

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

๒) ใช้ OAuth2 หรือ JWT เพื่อมอบสิทธิ (authorization) ให้แอปพลิเคชันเข้าถึงข้อมูลผู้ใช้โดยไม่ต้องเปิดเผยรหัสผ่าน

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

๔) ใช้การกำหนดสิทธิ์ (Authorization) เพื่อควบคุมว่าใคร สามารถทำอะไรได้บ้าง เช่น ผู้อำนวยการสามารถดูข้อมูลผู้ใต้บังคับบัญชาได้แต่ไม่สามารถแก้ไขข้อมูลได้

๕) ใช้ Rate Limiting เพื่อจำกัดจำนวนคำขอที่ client สามารถส่งเข้ามาได้ในช่วงเวลาหนึ่ง เช่น จำกัด API สำหรับการค้นหาไม่เกิน 100 ครั้ง/นาที/ไอพี เพื่อป้องกันการโจมตีแบบ DDoS

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

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

๘) การจัดการข้อผิดพลาด (Error Handling) ระบบต้องออกแบบการจัดการข้อผิดพลาดให้ปลอดภัยและสอดคล้องกับมาตรฐาน HTTP โดยมีข้อกำหนดอย่างน้อยดังนี้

๘.๑) 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

หลีกเลี่ยงการใช้รหัสสถานะที่ไม่เป็นมาตรฐานหรือกำหนดเองเพื่อความเข้ากันได้กับเครื่องมือและไคลเอนต์ต่าง ๆ

๙) เปิดเผยแพร่รายละเอียดภายในของระบบ เช่น Stack Trace หรือข้อมูลโครงสร้างภายใน เพื่อป้องกันการเปิดเผยข้อมูลที่อาจนำไปสู่ช่องโหว่ด้านความปลอดภัย

๑๐) สำหรับข้อผิดพลาดที่ไม่คาดคิดหรือข้อผิดพลาดภายในของเซิร์ฟเวอร์ ควรตอบกลับด้วยรหัสสถานะ 500 Internal Server Error และข้อความที่เป็นมิตรต่อผู้ใช้ เช่น "An unexpected error occurred. Please try again later."

๑๑) บันทึกข้อผิดพลาดภายในระบบ (Log) สำหรับการวิเคราะห์และแก้ไขปัญหาในภายหลัง โดยไม่เปิดเผยข้อมูลดังกล่าวต่อผู้ใช้

๑๒) ระบบควรมีการตรวจสอบและแจ้งเตือนเมื่อเกิดข้อผิดพลาดที่สำคัญ เช่น การเชื่อมต่อฐานข้อมูลล้มเหลว หรือบริการภายนอกไม่สามารถเข้าถึงได้ เพื่อให้สามารถดำเนินการแก้ไขได้อย่างรวดเร็ว

๑๓) ทำ Input Validation เพื่อป้องกันข้อมูลไม่ถูกต้องหรือการโจมตี เช่น เมื่อรับหมายเลขโทรศัพท์ผ่าน API ต้องตรวจสอบว่าเป็นตัวเลขเท่านั้นและความยาวไม่เกิน 10 หลัก

๑๔) การแบ่งหน้า (Pagination) เพื่อเพิ่มประสิทธิภาพในการจัดการชุดข้อมูลขนาดใหญ่ โดยการส่งข้อมูลกลับไปยังผู้ใช้แบบไหลต่อเนื่อง (streaming) เพื่อเพิ่มความเร็วในการให้บริการ

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

๑๖) การแคชข้อมูล (Data Caching) ข้อมูลที่มีการเรียกใช้งานบ่อย ๆ จะถูกเก็บไว้ในแคช เพื่อให้สามารถดึงข้อมูลได้รวดเร็วยิ่งขึ้น ลูกค้าน่าจะทำการตรวจสอบแคชก่อนที่จะทำการดึงข้อมูลจากฐานข้อมูล โดยระบบเก็บข้อมูลในหน่วยความจำ (เช่น Redis) ทำให้การเข้าถึงข้อมูลเร็วขึ้นมาก

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

๑๘) การใช้พูลการเชื่อมต่อ (Connection Pooling) เป็นการใช้กลุ่มการเชื่อมต่อฐานข้อมูลที่เปิดค้างไว้ เพื่อหลีกเลี่ยงการเสียเวลาสร้างและปิดการเชื่อมต่อใหม่ทุกครั้งที่ต้องโหลดข้อมูล ระบบจะบริหารจัดการการเชื่อมต่ออย่างมีประสิทธิภาพ

๓.๓.๔.๑๗. ต้องแยกความเป็นเจ้าของข้อมูล (Data Ownership) โดยมีความสัมพันธ์อย่างน้อยดังนี้

๑) แต่ละ Service ย่อยต้องเป็นเจ้าของ Database ของตัวเอง

๒) ห้ามแชร์ Database โดยตรงระหว่าง Services

๓.๓.๔.๑๘. ระบบต้องออกแบบตามแนวคิด Event-Driven Architecture (EDA) โดยให้บริการต่าง ๆ ภายในระบบสามารถสื่อสารกันผ่านการส่งเหตุการณ์ (Event) เช่น เมื่อมีการ

เปลี่ยนแปลงข้อมูลสำคัญในบริการหนึ่ง ระบบต้องสามารถสร้างและส่งเหตุการณ์ (Event) ไปยังระบบกลาง (Message Broker) เช่น RabbitMQ, Apache Kafka หรือเทคโนโลยีอื่นที่มีความสามารถดีกว่า ที่ทำหน้าที่จัดการการส่งต่อเหตุการณ์นี้ไปยังบริการอื่น ๆ ที่เกี่ยวข้องได้ เพื่อให้บริการปลายทางสามารถนำเหตุการณ์ดังกล่าวไปประมวลผลต่อได้ เช่น การแจ้งเตือนผู้ใช้ การอัปเดตข้อมูลเชิงสถิติ หรือกระบวนการอื่น ๆ ที่เกี่ยวข้อง ทั้งนี้ ผู้รับจ้างสามารถเลือกเทคโนโลยีหรือเครื่องมือที่เหมาะสม โดยต้องมั่นใจว่าโครงสร้างงานรองรับการขยายระบบ (Scalability) และการสื่อสารแบบแยกส่วน (Decoupling) ได้อย่างมีประสิทธิภาพ

๓.๓.๔.๑๙. 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 ไลบรารีหรือแพ็คเกจที่เป็น 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

๓.๓.๔.๒๐. 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

๓.๓.๔.๒๑. 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 90 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) และควบคู่กับการตรวจสอบความปลอดภัย โดยต้องนำแนวคิด Shift Left มาใช้ตั้งแต่ระยะเริ่มต้นของวงจรชีวิตการพัฒนา (SDLC) โดยมีคุณสมบัติอย่างน้อย ดังนี้

๓.๔.๑. ต้องมีการออกแบบ Workflow หรือ Pipeline ที่ประกอบด้วยขั้นตอนอย่างน้อย ดังนี้

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) ผู้รับจ้างต้องจัดให้มีกระบวนการควบคุมเวอร์ชันและการบริหารจัดการการเปลี่ยนแปลง (Version Control and Change Control) สำหรับการพัฒนา ระบบ โดยใช้ระบบ Version Control และกำหนดแนวทางการบริหาร Branch ที่เหมาะสม เช่น Git Flow หรือแนวทางอื่นที่มีคุณสมบัติเทียบเท่า เพื่อให้สามารถบริหารจัดการ Source Code, Configuration, Infrastructure as Code (IaC), Database Schema, Build/Deployment Artifact, Release และองค์ประกอบอื่นที่เกี่ยวข้องกับการพัฒนาระบบได้อย่างเป็นระบบ

กระบวนการดังกล่าวต้องสามารถตรวจสอบย้อนกลับการเปลี่ยนแปลงได้ โดยอย่างน้อยต้องสามารถระบุ Version หรือ Release ที่เกี่ยวข้อง ผู้ดำเนินการ วันที่และเวลา รายละเอียดหรือเหตุผลของการเปลี่ยนแปลง ตลอดจนสามารถเชื่อมโยงกับการทดสอบ การอนุมัติ และการนำขึ้นใช้งาน (Deployment) ที่เกี่ยวข้องได้

สำหรับองค์ประกอบด้านปัญญาประดิษฐ์ การควบคุมเวอร์ชัน การเปลี่ยนแปลง การลงทะเบียน การอนุมัติ การนำขึ้นใช้งาน และการตรวจสอบย้อนกลับ ให้เป็นไปตามภาคผนวก ง โดยเฉพาะข้อ ๒.๖ ข้อ ๒.๘ ข้อ ๕.๓ ถึงข้อ ๕.๕ และข้อ ๕.๘ ตามที่เกี่ยวข้อง

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 โดยมีเป้าหมายเพื่อป้องกันไม่ให้เกิดที่มีความเสี่ยงถูกนำไปใช้งานในระบบ ซึ่งอาจส่งผลกระทบต่อความปลอดภัยหรือเสถียรภาพของระบบในอนาคต

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 และป้องกันไม่ให้เกิดปัญหาซ้ำซ้อนหรือกระบวนในอนาคต

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 ใช้เพื่อเปลี่ยนระบบไปยังเวอร์ชันใหม่โดยไม่กระทบผู้ใช้งาน และสามารถย้อนกลับไปยังเวอร์ชันเดิมได้ทันทีหากพบปัญหา

๒) เป็นการเตรียมระบบขึ้น 2 ชุด คือ ชุดปัจจุบัน (Blue) และชุดใหม่ (Green) โดยไม่กระทบกับผู้ใช้งานระหว่างเตรียมระบบใหม่

๒.๑) เมื่อระบบใหม่ทดสอบเรียบร้อยแล้ว จะทำการสลับผู้ใช้งานมายัง Green ทันที

๒.๒) หากพบปัญหา ต้องสามารถ ย้อนกลับไปยัง Blue ได้อย่างรวดเร็ว (Rollback)

๒.๓) Canary Deployment ใช้เพื่อทยอยปล่อยระบบใหม่ให้ผู้ใช้งานทีละกลุ่ม เพื่อตรวจสอบความเสถียรก่อนเปิดให้ผู้ใช้งานทั้งหมด

๒.๔) ปล่อยระบบใหม่ให้ กลุ่มผู้ใช้งานจำนวนน้อย เช่น 10% - 25% ใช้งานก่อน

๒.๕) ตรวจสอบว่าระบบใหม่ไม่มีปัญหา จึงค่อย ๆ ขยายการใช้งานไปยังผู้ใช้งานทั้งหมด

๒.๖) ต้องสามารถ หยุดและถอยกลับ (Rollback) ได้ทันทีหากเกิดข้อผิดพลาด

- ๓) A/B Testing ใช้เพื่อทดสอบและเปรียบเทียบประสิทธิภาพของฟีเจอร์หรือการออกแบบระหว่างเวอร์ชัน A และ B กับผู้ใช้งานจริง
- ๓.๑) ใช้สำหรับการทดลองฟีเจอร์ใหม่ หรือเปรียบเทียบการออกแบบ (UI/UX) โดย แบ่งกลุ่มผู้ใช้งานเป็นกลุ่ม A และ B
- ๓.๒) ต้องสามารถเก็บข้อมูลพฤติกรรมผู้ใช้งานแต่ละกลุ่มเพื่อนำมาวิเคราะห์ก่อนตัดสินใจใช้งานจริง
- ๔) Feature Flag Deployment ใช้เพื่อควบคุมการเปิด-ปิดฟีเจอร์ใหม่ได้แบบยืดหยุ่น โดยไม่ต้อง deploy โค้ดใหม่ทุกครั้ง
- ๔.๓) ฟีเจอร์ใหม่สามารถ เปิด-ปิดได้แบบ Dynamic โดยไม่ต้อง Deploy Code ใหม่
- ๔.๔) สามารถเลือกเปิดใช้ฟีเจอร์ใหม่ให้เฉพาะบางกลุ่มผู้ใช้งาน (เช่น 10%-20%) ได้
- ๔.๕) ต้องมีระบบบริหารจัดการ Feature Flag ที่ปลอดภัย และต้องลบ Feature Flag ที่ไม่ใช้งานแล้วเสมอ
- ๕) Rolling Deployment ใช้เพื่ออัปเดตระบบทีละส่วน (Node/Instance) อย่างต่อเนื่อง โดยไม่ทำให้ระบบหยุดให้บริการ
- ๕.๑) ค่อย ๆ อัปเดตระบบใหม่ ทีละ Node/Instance แบบต่อเนื่อง โดยไม่หยุดให้บริการทั้งหมด
- ๕.๒) ต้องสามารถ ตรวจสอบสถานะและ Rollback ได้ทันทีหากพบปัญหา
- ๖) Multi-Service Deployment ใช้เพื่อปรับใช้หลาย Service พร้อมกันอย่างเป็นระบบ โดยคำนึงถึง Dependency ระหว่าง Service ที่เกี่ยวข้อง
- ๖.๑) ใช้สำหรับระบบที่ประกอบด้วยหลาย Service ต้อง Deploy พร้อมกันเป็นชุดเดียว
- ๖.๒) ต้องจัดทำแผนลำดับการ Deploy อย่างชัดเจน เนื่องจากบาง Service อาจขึ้นอยู่กับ Service อื่น (Dependency)
- ๖.๓) ต้องมีแผนสำรองหากบาง Service ล้มเหลว โดยไม่ให้เกิดกระทบระบบทั้งหมด

๓.๔.๑.๑๖. ต้องสามารถตรวจสอบประวัติการ Deploy ได้ครบถ้วน (Audit Trail) ระบุ อย่างน้อยได้ว่า 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

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

๓.๔.๗. ต้องมีขั้นตอนการตรวจสอบความปลอดภัยของ Container Image ที่ใช้ทั้งในขั้นตอนการพัฒนาและการนำขึ้นใช้งานจริง

๓.๔.๘. ต้องใช้ Container Image ที่มาจากแหล่งที่เชื่อถือได้ (Trusted Registry) เท่านั้น

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

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

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

๓.๔.๑๐.๒. ใช้ COPY แทน ADD เมื่อไม่จำเป็น

๓.๔.๑๐.๓. ลดจำนวน Layer โดยรวมคำสั่งหลาย ๆ คำสั่งเข้าด้วยกัน

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

๓.๔.๑๐.๕. ใช้ Official Image จาก Docker Hub เพื่อความปลอดภัยและการอัปเดตอย่างสม่ำเสมอ ให้เลือกใช้งาน Image ที่เป็นทางการ (Official) และ เป็น Version LTS เท่านั้น และ กำหนดเวอร์ชันของ Image อย่างชัดเจน ห้ามใช้ latest tag ให้ระบุเวอร์ชันที่แน่นอน เพื่อความสามารถในการ reproduce และลดความเสี่ยงจากการเปลี่ยนแปลงที่ไม่คาดคิด

๓.๔.๑๐.๖. 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

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

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

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

๓.๔.๑๐.๑๐. สแกนหาช่องโหว่ใน Image ก่อนใช้งานจริงก่อนนำ Docker Image ไปใช้งานในระบบจริง ผู้รับจ้างต้องสแกนหาช่องโหว่ (Vulnerability Scan) ด้วยเครื่องมือที่เป็นที่ยอมรับ เช่น docker scan, Trivy หรือเครื่องมืออื่นที่เทียบเท่า และต้องแนบรายงานผลการสแกนเพื่อประกอบการตรวจสอบต้องปฏิบัติตามแนวทางการตั้งค่าความปลอดภัยอย่างน้อยระดับ Level 1 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

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

๑) ต้องใช้วิธีการจัดเก็บ Secrets อย่างปลอดภัย เช่นผ่าน Secret Manager หรือ Environment Variables ที่แยกจาก Source Code และต้องมีการบริหารจัดการข้อมูลความลับของระบบ (Secrets Management) โดยต้องคำนึงถึงความปลอดภัย เช่น

๒) ห้ามฝังข้อมูลความลับ (No Hardcoded Secrets): ห้ามมิให้มีการบันทึกหรือฝังข้อมูลความลับใดๆ ลงในซอร์สโค้ด (Source Code), ไฟล์ตั้งค่าที่ไม่มีการเข้ารหัส, หรือระบบจัดการเวอร์ชัน (Version Control System) โดยเด็ดขาด

๓) การจัดเก็บอย่างปลอดภัย (Secure Storage): ผู้รับจ้างต้องจัดเก็บข้อมูลความลับผ่านกลไกที่ปลอดภัย เช่น การใช้ตัวแปรสภาพแวดล้อม (Environment Variables) ที่ถูกจำกัดสิทธิ์การเข้าถึง ซึ่งแยกออกจากซอร์สโค้ดอย่างชัดเจน

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

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

๓.๔.๑๐.๑๔. ต้องสามารถระบุได้ว่า Container ใดกำลังทำงาน อ้างอิงกับเวอร์ชันที่ผ่านการทดสอบแล้ว

๓.๔.๑๐.๑๕. Container ต้องถูกตรวจสอบสถานะและจัดเก็บ Log อย่างเหมาะสม เพื่อให้สามารถติดตามพฤติกรรมผิดปกติได้

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

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

๓.๔.๑๐.๑๘. ระบบต้องสามารถระงับการ Deploy Image ที่ไม่ผ่านเกณฑ์ความปลอดภัย (เช่น ถ้าพบช่องโหว่ระดับวิกฤติ)

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

๓.๕ ผู้รับจ้างต้องออกแบบการตรวจสอบความปลอดภัยของซอร์สโค้ด (Secure Code Review) โดยใช้เครื่องมือหรือวิธีการที่สามารถ ดังนี้

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

๓.๕.๒. ตรวจจับจุดอ่อนด้านความปลอดภัย (Security Vulnerabilities) ได้ครอบคลุมตามแนวทาง OWASP Top 10 เช่น SQL Injection, Cross-Site Scripting (XSS), Broken Authentication และอื่น ๆ

๓.๕.๓. แสดงระดับความรุนแรงของปัญหา (Severity Levels) และสามารถจัดลำดับความสำคัญเพื่อแก้ไขได้

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

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

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

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

๓.๕.๘. เครื่องมือหรือแนวทางที่เลือกใช้จะต้องเป็นที่ยอมรับในอุตสาหกรรม และผลการวิเคราะห์ต้องสามารถตรวจสอบย้อนหลังได้ เพื่อประกอบการประเมินผลและรับรองคุณภาพของระบบ

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

๓.๕.๙.๑. รองรับการดำเนินงานแบบ Branch/Merge

๓.๕.๙.๒. จัดเก็บประวัติการเปลี่ยนแปลงของโค้ดและฐานข้อมูลแบบละเอียด

๓.๕.๙.๓. มีระบบตรวจสอบความแตกต่างของเวอร์ชัน (Diff)

๓.๕.๙.๔. รองรับการ Rollback กรณี Migration ผิดพลาด

๓.๕.๙.๕. รองรับการแยก Environment สำหรับการ Deploy

๓.๕.๙.๖. มีระบบกำหนดสิทธิ์การเข้าถึงของผู้พัฒนา

๓.๕.๙.๗. บันทึกเหตุการณ์หรือ Log การเปลี่ยนแปลงที่สำคัญ

๓.๖ ผู้รับจ้างต้องออกแบบให้สามารถจัดการข้อมูลแบบ Cache เพื่อเพิ่มประสิทธิภาพในการเข้าถึงข้อมูล โดยต้องรองรับการทำงานต่อเนื่อง (High Availability) และสามารถจัดการกับการทำงานที่มีปริมาณสูงได้

๓.๗ ผู้รับจ้างต้องออกแบบให้รองรับการเข้ารหัสข้อมูลตามมาตรฐานความปลอดภัยระดับสากล เช่น TLS 1.2 หรือสูงกว่า และต้องรองรับการพิสูจน์ตัวตนแบบ Token-Based เพื่อป้องกันการโจมตีและการเข้าถึงโดยไม่ได้รับอนุญาต

๓.๘ ผู้รับจ้างต้องออกแบบและจัดให้มีการบริหารจัดการบทบาทและสิทธิของผู้ใช้งาน (User Role and Authorization Management) รวมทั้งจัดทำตารางกำหนดบทบาทและสิทธิของผู้ใช้งาน (Role-Permission Matrix / User Authorization Matrix) ให้สอดคล้องกับหน้าที่ ความรับผิดชอบ และขอบเขตการใช้งานของแต่ละระบบ โดยอย่างน้อยต้องครอบคลุมรายละเอียดดังต่อไปนี้

๑) การกำหนดโครงสร้างบทบาทของผู้ใช้งาน (Role Structure) โดยรองรับแนวทางการควบคุมการเข้าถึงตามบทบาท (Role-Based Access Control: RBAC) หรือแนวทางอื่นที่เหมาะสม

๒) การกำหนดและจัดกลุ่มสิทธิ (Permission Grouping) เช่น สิทธิในการดูข้อมูล เพิ่ม แก้ไข ลบ อนุมัติ หรือดำเนินการอื่นตามหน้าที่

๓) การกำหนดความสัมพันธ์ระหว่างบทบาทกับสิทธิ (Role-Permission Mapping) รวมถึงระบบ โมดูล เมนู ฟังก์ชัน ข้อมูล หรือทรัพยากรที่แต่ละบทบาทสามารถเข้าถึงหรือดำเนินการได้

๔) การจัดทำแผนภาพหรือตารางแสดงความสัมพันธ์ระหว่างผู้ใช้งาน บทบาท สิทธิ และส่วนประกอบของระบบ เพื่อให้สามารถตรวจสอบและทำความเข้าใจโครงสร้างการกำหนดสิทธิได้อย่างชัดเจน

๕) การออกแบบและกำหนดสิทธิต้องเป็นไปตามหลักสิทธิเท่าที่จำเป็น (Least Privilege) หลักการแยกหน้าที่ (Separation of Duties) และหลักการด้านความมั่นคงปลอดภัยที่เกี่ยวข้อง รวมทั้งต้องสามารถตรวจสอบย้อนหลังการกำหนดหรือเปลี่ยนแปลงสิทธิได้

๖) ระบบต้องสามารถจัดทำและส่งออกตาราง Role-Permission Matrix หรือ User Authorization Matrix ในรูปแบบ PDF และรูปแบบไฟล์อิเล็กทรอนิกส์ที่สามารถนำไปใช้งานหรือวิเคราะห์ต่อได้ เช่น Excel, CSV หรือรูปแบบอื่นที่เหมาะสม รวมทั้งสามารถพิมพ์เอกสารดังกล่าวได้

๗) ผู้รับจ้างต้องสามารถจัดทำหรือส่งออกรายงานบทบาทและสิทธิของผู้ใช้งานตามรอบระยะเวลา หรือ ณ ช่วงเวลาที่สำนักงานการตรวจเงินแผ่นดินกำหนด เพื่อใช้ในการตรวจสอบ ทบทวน และรับรองสิทธิการใช้งานของผู้ใช้งาน

๓.๙ ผู้รับจ้างต้องออกแบบและจัดทำโครงสร้างพื้นฐานของระบบโดยใช้แนวทาง Infrastructure as Code (IaC) เพื่อให้สามารถสร้าง ติดตั้ง และปรับขนาด (Provision & Scale) สภาพแวดล้อมได้โดยอัตโนมัติ

และสามารถควบคุมเวอร์ชันได้อย่างมีประสิทธิภาพ โดยโครงสร้างพื้นฐานที่พัฒนาด้วย IaC ต้องสามารถสร้างระบบในรูปแบบ Cluster ได้เป็นอย่างดี

๓.๑๐ ผู้รับจ้างต้องออกแบบระบบให้สามารถเชื่อมต่อและทำงานร่วมกับเครื่องมือสำหรับตรวจสอบประสิทธิภาพ (Performance Monitoring) และติดตามการทำงานของระบบใน Environment UAT และ Production โดยเครื่องมือที่เลือกใช้ต้องมีคุณสมบัติดังนี้

๓.๑๐.๑. ระบบต้องสามารถรองรับการติดตามการทำงานแบบกระจาย (Distributed Tracing) โดยมีคุณสมบัติอย่างน้อยดังนี้

๓.๑๐.๒. สามารถตรวจสอบและติดตามเส้นทางการทำงานของแต่ละ Request/Transaction ที่ผ่านระบบ Microservices หรือส่วนประกอบต่าง ๆ ได้

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

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

๓.๑๐.๕. สามารถตรวจสอบสถานะต่างๆ ได้แบบเรียลไทม์ โดยมีรายละเอียดอย่างน้อยดังนี้

๑) สถานะ ของ Service/Component ต่าง ๆ

๒) สถานะ CPU

๓) สถานะ Memory

๔) สถานะ Disk แต่ละ Partition

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

๓.๑๐.๖. รองรับการทำงานร่วมกับเครื่องมือมาตรฐานอุตสาหกรรม โดยมีคุณสมบัติอย่างน้อยดังนี้

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

๓.๑๐.๘. ไม่จำกัดยี่ห้อหรือผลิตภัณฑ์ เพื่อเปิดโอกาสให้สามารถเลือกใช้ Open Source หรือ Commercial Solution ได้ตามความเหมาะสม

๓.๑๐.๙. รองรับการบันทึกและตรวจสอบข้อมูลย้อนหลัง (Historical Analysis) โดยมีคุณสมบัติอย่างน้อยดังนี้

๓.๑๐.๙.๑. ผลการมอนิเตอร์ต้องสามารถบันทึกและตรวจสอบย้อนหลังได้

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

๓.๑๑ ผู้รับจ้างต้องออกแบบและกำหนดการเข้าถึงระบบไฟล์และโพลเดอร์ โดยมีการแบ่งเส้นทางการใช้งานและกำหนดกลุ่มผู้ใช้งานให้เหมาะสมกับหน้าที่ความรับผิดชอบและความมั่นคงปลอดภัยของระบบ ซึ่งอย่างน้อยต้องมีรายละเอียดดังนี้

๓.๑๑.๑. การทำ Mapping กลุ่มผู้ใช้และโพลเดอร์ จัดทำ Mapping ระหว่างกลุ่มผู้ใช้ (User Group) กับโพลเดอร์ที่เกี่ยวข้อง เช่น

๓.๑๑.๒. กลุ่ม Jenkins Admin → ดูแล /jenkins

๓.๑๑.๓. กลุ่ม Apache Admin → ดูแล /apache_log, /apache_dump, /apache_archive

๓.๑๑.๔. กลุ่ม Elastic Admin → ดูแล /elasticdata1, /elasticdata2

๓.๑๑.๕. การกำหนดสิทธิ์การเข้าถึง กำหนดสิทธิ์ที่เหมาะสม เช่น read, write, execute ตามบทบาทหน้าที่

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

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

๓.๑๒ การออกแบบและจัดการซอร์สโค้ดและ Micro Frontend ผู้รับจ้างต้องออกแบบโครงสร้างคลังเก็บซอร์สโค้ด (Source Code Repositories) และการพัฒนาระบบ Frontend ตามหลักการดังต่อไปนี้

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

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

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

๓.๑๒.๓.๑. Compression

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

๒) ลดขนาด Payload และ Network Latency

๓) Selective Rendering / Windowing

๔) แสดงเฉพาะองค์ประกอบที่อยู่ในพื้นที่มองเห็น (Above the Fold)

๕) ใช้ Virtual Scrolling หรือ Lazy Rendering สำหรับรายการที่มีจำนวนมาก ตามความเหมาะสมของเนื้อหา (Content) ในแต่ละหน้าจอ

๓.๑๒.๓.๒. Modular Architecture & Code Splitting

๑) แยก JavaScript ออกเป็นโมดูลย่อยเพื่อให้โหลดเฉพาะส่วนที่จำเป็น เช่น Webpack SplitChunks หรือ ES Modules

๒) ลดขนาด Initial JS Bundle

๓.๑๒.๓.๓. Priority-Based Loading

๑) โหลดเฉพาะทรัพยากรสำคัญก่อน เช่น HTML, Critical CSS, JS ที่จำเป็นใช้เทคนิค อย่างน้อยดังนี้

- defer เพื่อโหลด script พร้อมกัน รั้นที่หลัง HTML
- async เพื่อโหลดและรันทันที (ถ้าโหลดเสร็จ)
- preload เพื่อบอก browser ให้โหลด resource สำคัญล่วงหน้า
- resource hints เพื่อบอก browser ให้เตรียมการเชื่อมต่อหรือ

ลิงก์ล่วงหน้า

๓.๑๒.๓.๔. Dynamic Imports

๑) โหลดโมดูลแบบ Lazy ตามการใช้งานจริง เช่น
import('module')

๒) ลดเวลาโหลดหน้าแรก (TTI – Time to Interactive)
๓.๑๒.๓.๕. Tree Shaking / Dead Code Removal

๑) ตัดโค้ดที่ไม่ถูกใช้งานออกจาก Final JS Bundle
๒) ใช้กับ Bundler เช่น Webpack, Rollup

๓.๑๒.๓.๖. Pre-Fetching
๑) ดึงข้อมูลล่วงหน้าก่อนที่ผู้ใช้งานจะร้องขอจริง เช่น
ข้อมูลในหน้า Next Page

๒) ใช้ <link rel="prefetch"> หรือ Browser Cache API
๓.๑๒.๓.๗. Pre-loading / Preconnect / DNS-Prefetch
๑) ให้ Browser เตรียมตัวโหลดไฟล์ที่จำเป็นได้เร็วขึ้นสำหรับ

CDN, Fonts และ API Endpoint

๓.๑๓. กรณิที่ผู้รับจ้างมีการพัฒนาการประมวลผลชุดข้อมูล (Batch Processing) โดยต้องมีการ
ออกแบบอย่างน้อยดังนี้

๓.๑๓.๑. ผู้รับจ้างต้องออกแบบและพัฒนาระบบประมวลผลชุดข้อมูล (Batch Job)
ให้สามารถดำเนินการได้อย่างมีประสิทธิภาพ มีความยืดหยุ่นต่อปริมาณข้อมูลที่เปลี่ยนแปลง
และรองรับการประมวลผลขนาดใหญ่ได้อย่างมีประสิทธิภาพ

๓.๑๓.๒. ระบบ Batch ต้องสามารถทำงานแบบอัตโนมัติ (Automated Scheduling) โดย
รองรับการตั้งเวลาแบบกำหนดเองได้ (Flexible Scheduling) และรองรับการทำงานซ้ำอัตโนมัติกรณีล้มเหลว
(Automatic Retry with Configurable Policy)

๓.๑๓.๓. ผู้รับจ้างต้องออกแบบการทำงานของ Batch ให้รองรับการยืนยันตัวตนและ
กำหนดสิทธิ์ผู้ใช้งาน (Authentication and Authorization) อย่างเหมาะสม

๓.๑๓.๔. ต้องมีการเข้ารหัสข้อมูลที่ใช้ระหว่างการประมวลผล
ทั้งข้อมูลที่อยู่ในระหว่างการส่ง (In-Transit) และข้อมูลที่จัดเก็บ (At-Rest) ตามมาตรฐานความปลอดภัยสากล
เช่น TLS 1.2 หรือสูงกว่า และ AES-256

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

๓.๑๓.๖. ระบบต้องสามารถรองรับกรณีการล้ม (Failure) ได้โดยต้องมีฟังก์ชันดังต่อไปนี้

๓.๑๓.๖.๑. รองรับการทำ Retry และ Recovery อัตโนมัติ

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

๓.๑๓.๖.๓. กรณีผิดพลาดร้ายแรง ต้องสามารถสั่ง Rollback หรือทำ Data Recovery ได้อย่างปลอดภัย

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

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

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

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

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

๓.๑๓.๑๒. การพัฒนาระบบ Batch ต้องสามารถบูรณาการเข้ากับกระบวนการ DevSecOps ได้ เช่น รองรับการตรวจสอบช่องโหว่ก่อน Deployment

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

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

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

๓.๑๔.๑.๑. ผู้รับจ้างต้องออกแบบและตั้งค่าการเชื่อมต่อ WebSocket ทั้งฝั่ง Client และ Server ให้เป็นไปตามมาตรฐานอุตสาหกรรม โดยมีข้อกำหนดขั้นต่ำ ดังนี้

๓.๑๔.๑.๑.๑. มาตรฐาน IETF RFC 6455 (The WebSocket Protocol) เช่น ต้องสามารถรักษาการเชื่อมต่อแบบถาวร (Persistent Connection) ต้องสอดคล้องกับโดยไม่เกิด Timeout หรือถูกตัดการเชื่อมต่อก่อนการประมวลผลเสร็จสิ้น

๓.๑๔.๑.๑.๒. 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

๓.๑๔.๑.๒.๑. การจัดการช่องโหว่ (Vulnerability Management) โดยผู้รับจ้างต้องดำเนินการอย่างน้อยดังนี้

๓.๑๔.๑.๒.๑.๑. รวบรวม วิเคราะห์ และจัดทำรายการ CWE ที่เกี่ยวข้องกับระบบและเทคโนโลยีที่ใช้จริง

๓.๑๔.๑.๒.๑.๒. ดำเนินการแก้ไข (Mitigation) ช่องโหว่ที่ตรวจพบทั้งหมดและไม่อยู่ในระดับ Critical หรือ High Severity

๓.๑๔.๑.๒.๑.๓. ต้องมีหลักฐานการตรวจสอบโดยสามารถตรวจสอบย้อนหลังได้

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

๓.๑๔.๓.๑. รองรับจำนวนผู้ใช้งานพร้อมกัน (Concurrent Connections) โดยต้องรองรับการเชื่อมต่อพร้อมกันไม่น้อยกว่า จำนวนที่ระบบคาดว่าจะใช้งานพร้อมกันสูงสุดที่กำหนด + buffer 30%

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

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

๓.๑๔.๓.๔. การป้องกันการโจมตี โดยต้องสามารถดำเนินการอย่างน้อยดังนี้

- ๑) ต้องรองรับการตั้งค่าจำกัดจำนวนการเชื่อมต่อ (Connection Limit)

- ๒) ต้องสามารถตั้งค่า Rate Limit เช่น จำกัด request ต่อวินาที เพื่อป้องกันการโจมตี

- ๓) ต้องสามารถจำกัด Memory Usage ต่อการเชื่อมต่อ เพื่อไม่ให้ผู้ไม่หวังดีเชื่อมต่อหลาย session แล้วกินทรัพยากรจนระบบล่ม

๓.๑๔.๓.๕. การบันทึกข้อมูลการเชื่อมต่อ (Connection Logging) โดยต้องสามารถดำเนินการอย่างน้อยดังนี้

- ๑) ต้องมีระบบบันทึก Log ทุกครั้งที่มีการเชื่อมต่อหรือตัดการเชื่อมต่อ
- ๒) ข้อมูลที่ต้องบันทึกอย่างน้อยต้องมี เช่น
- ๓) ที่เกิดเหตุการณ์ (timestamp)
- ๔) หมายเลขไคลเอนต์หรือ IP Address ของผู้เชื่อมต่อ (Client ID/IP)
- ๕) เหตุผลของข้อผิดพลาดหรือการตัดการเชื่อมต่อ (Error Reason)

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

๓.๑๔.๔.๑. การกลับมาเชื่อมต่อใหม่อัตโนมัติ (Auto Reconnect) หากเมื่อผู้ใช้ (Client) หลุดจากการเชื่อมต่อโดยไม่ตั้งใจ เช่น สัญญาณเน็ตหลุด โดยระบบต้องสามารถ เชื่อมต่อกลับ (Resume Connection) ได้โดยอัตโนมัติ ภายใน 5 วินาที เพื่อให้ผู้ใช้งานใช้งานได้อย่างต่อเนื่อง ไม่สะดุด

๓.๑๔.๔.๒. ระบบจับเวลาตรวจสอบการตอบสนอง (Timeout Handling) หาก Client ส่งข้อมูลไปยัง Server แล้วไม่ได้รับการตอบกลับภายในเวลาที่กำหนด เช่น 30 วินาที ระบบต้องมีการตัดการเชื่อมต่อและแจ้งเตือน หรือพยายาม reconnect ใหม่โดยอัตโนมัติ เพื่อไม่ให้เกิด session ค้างหรือเปลืองทรัพยากร

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

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

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

๓.๑๕ ฐานข้อมูล (Database)

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

๓.๑๕.๑.๑. Indexing

- ๑) ต้องมีการวิเคราะห์และสร้างดัชนี (Index) อย่างเหมาะสมกับรูปแบบการใช้งาน (Query Pattern)
- ๒) ต้องหลีกเลี่ยงการสร้าง Index ซ้ำซ้อนหรือมากเกินไป ซึ่งอาจกระทบต่อประสิทธิภาพการเขียนข้อมูล

EXPLAIN, ANALYZE

๓) ต้องมีการตรวจสอบและปรับปรุง Index อย่างสม่ำเสมอ เช่น ผ่าน

๓.๑๕.๑.๒. Query Optimization

- ๑) หลีกเลี่ยง SELECT * และใช้ฟิลด์เฉพาะที่จำเป็น
- ๒) เขียน WHERE, JOIN, ORDER BY, GROUP BY ให้มีประสิทธิภาพ
- ๓) ตรวจสอบแผนการประมวลผล (Query Plan) โดยใช้คำสั่งเช่น

EXPLAIN, EXPLAIN ANALYZE

๓.๑๕.๑.๓. Connection Pooling

- เพื่อลดการเปิด/ปิด connection บ่อยครั้ง
- ๑) ต้องใช้เทคนิค Connection Pooling เช่น HikariCP, pgBouncer
 - ๒) ช่วยลด Latency และเพิ่ม Throughput ของระบบ

๓.๑๕.๑.๔. Caching

- ๑) ต้องมีการพิจารณาใช้ Cache ทั้งในระดับ Application เช่น Redis, Memcached และระดับ Database เช่น Query Cache
- ๒) ต้องจัดการ Expiry และ Invalidation อย่างเหมาะสม

๓.๑๕.๑.๕. Sharding และ Partitioning

- ต้องพิจารณาใช้ Sharding หรือ Partitioning
- ๑) สำหรับระบบที่มี Dataset ใหญ่หรือปริมาณการเขียนสูง
 - ๒) Sharding: แยกฐานข้อมูลตาม Key เช่น UserID
 - ๓) Partitioning: แยกตามเงื่อนไข เช่น Range หรือ Hash
 - ๔) ต้องมีแผนการจัดการและตรวจสอบความสมดุลระหว่าง Shard

๓.๑๕.๑.๖. Replication

- ๑) ต้องออกแบบการทำ Replication เพื่อรองรับ Read Scaling และ High Availability เช่น Master-Slave, Multi-Master
- ๒) ต้องจัดการ Latency, Conflict และ Read Consistency ให้เหมาะสม

๓.๑๕.๑.๗. Hardware และ Resource

- ๑) ต้องพิจารณาทรัพยากรของฐานข้อมูลให้เพียงพอ เช่น RAM, CPU และ Storage

๒) หากเป็น Cloud ต้องระบุประเภท instance หรือ storage ที่มี IOPS สูง

๓.๑๕.๑.๘. Monitoring

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

๓.๑๕.๑.๙. Normalization / Denormalization

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

๒) ต้องมีการประเมินผลกระทบต่อประสิทธิภาพและความสมบูรณ์ของข้อมูล (Data Integrity)

๓.๑๕.๑.๑๐. Materialized Views

๑) พิจารณาใช้ Materialized View สำหรับคำสั่ง Query ที่ซับซ้อนและใช้บ่อย โดยแยกการคำนวณล่วงหน้า

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

๓.๑๕.๑.๑๑. Vertical Scaling

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

๒) ควรออกแบบให้สามารถย้ายจาก Vertical ไปสู่ Horizontal ได้ในอนาคต

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

๓.๑๕.๒.๑. Atomicity – การทำธุรกรรมแบบรวมทั้งหมด

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

๒) ในกรณีที่เกิดข้อผิดพลาด ต้องสามารถ Rollback ฐานข้อมูลกลับไปสู่สถานะก่อนเริ่มธุรกรรมนั้นได้อย่างสมบูรณ์

๓.๑๕.๒.๒. Consistency – ความสอดคล้องของข้อมูล

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

๒) ทุกธุรกรรมที่เกิดขึ้น ต้องไม่ทำให้ข้อมูลผิดกฎหรือขัดกับเงื่อนไขของระบบ ธุรกรรมที่เกิดขึ้น ต้องไม่ทำให้ข้อมูลผิดกฎหรือขัดกับเงื่อนไขของระบบ ทั้งก่อนและหลังการทำธุรกรรมเสมอ

๓.๑๕.๒.๓. 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) กรณีที่ ระบบมีการใช้งานพร้อมกันจำนวนมาก (High Load) ต้องออกแบบเพื่อรองรับการเกิดความขัดแย้งระหว่างการอ่านและเขียนข้อมูล (Read-Write Conflict) และหลายธุรกรรมแย่งกันล็อกข้อมูลเดียวกัน (Lock Contention)

๔) การจำกัดระยะเวลาของธุรกรรม Transaction Timeout Control) ระบบฐานข้อมูล ต้องมีกลไกป้องกันธุรกรรมค้าง (Long-Running / Idle Transaction) ที่อาจถือครอง Lock จนกระทบต่อการทำงานของธุรกรรมอื่น โดยมีข้อกำหนดอย่างน้อยดังนี้

๔.๑) ต้องกำหนดค่าระยะเวลาสูงสุดที่ Session สามารถเปิดธุรกรรม ค้างไว้โดยไม่มีการทำงาน (Idle-in-Transaction Timeout) เมื่อเกินกำหนดระบบต้องยุติ Session หรือ Rollback ธุรกรรมนั้นโดยอัตโนมัติ เช่น การกำหนดค่า idle_in_transaction_session_timeout (PostgreSQL) หรือกลไก เทียบเท่าของระบบฐานข้อมูลที่ใช้

๔.๒) ต้องกำหนดค่าระยะเวลาสูงสุดในการรอ Lock (Lock Wait Timeout) เพื่อให้ธุรกรรมที่รอ Lock นานเกินกำหนดถูกยกเลิกพร้อมคืนค่าข้อผิดพลาดให้แอปพลิเคชันจัดการ แทนการรอโดยไม่มีการกำหนด

๔.๓) ต้องกำหนดค่าระยะเวลาสูงสุดของการประมวลผลคำสั่ง (Statement Timeout) สำหรับคำสั่งที่รันนานผิดปกติ โดยค่า Timeout ตามข้อ ๔.๑) ถึง ๔.๓) ผู้รับจ้างต้องเสนอค่าที่เหมาะสมกับลักษณะงานแต่ละประเภท (เช่น งาน Online Transaction และงาน Batch แยกจากกัน) และต้องได้รับความเห็นชอบจากลูกค้าก่อนนำไปใช้งานจริง

๔.๔) ฝั่งแอปพลิเคชันต้องออกแบบให้ทุกธุรกรรมมีจุดสิ้นสุดที่ชัดเจน (COMMIT หรือ ROLLBACK) ภายในขอบเขตการทำงานเดียวกันโดยห้ามเปิดธุรกรรมค้างไว้ระหว่างรอการ

ตอบสนองจากผู้ใช้งานหรือการเรียกใช้บริการภายนอก (External Service Call) และต้องมีการจัดการข้อผิดพลาด (Exception Handling) ที่รับประกันว่าธุรกรรมจะถูก Rollback เสมอเมื่อเกิดข้อผิดพลาด

๕) การเฝ้าระวังธุรกรรมค้าง (Transaction Monitoring) ระบบ Monitoring ต้องสามารถตรวจจับและแจ้งเตือน (Alert) อย่างน้อยในกรณีดังต่อไปนี้

๕.๑) Session ที่เปิดธุรกรรมค้างเกินระยะเวลาที่กำหนด (Idle in Transaction) โดยแสดงข้อมูลอย่างน้อย ได้แก่ Session ID, ผู้ใช้งาน, IP Address, คำสั่งล่าสุดที่ประมวลผล และระยะเวลาที่ค้าง

๕.๒) สถานการณ์ Lock Contention หรือ Blocking Chain ที่มีธุรกรรมรอคิว เกินจำนวนหรือระยะเวลาที่กำหนด โดยต้องระบุได้ว่า Session ใด เป็นต้นเหตุของการ Block (Blocker Session)

๕.๓) เหตุการณ์ที่ระบบยุติ Session หรือ Rollback ธุรกรรมโดยอัตโนมัติ ตามข้อ ๖) ต้องถูกบันทึก Log เพื่อให้สามารถตรวจสอบย้อนหลัง (Audit Trail) และวิเคราะห์หาสาเหตุจากฝั่งแอปพลิเคชันได้

๖) Durability – ความคงทนของข้อมูล

๖.๑) ข้อมูลที่ถูก Commit แล้วต้องไม่สูญหาย แม้เกิดระบบล้มเหลว เช่น ไฟดับ ระบบรีสตาร์ท

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

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

ผู้รับจ้างต้องเลือกประเภทของฐานข้อมูล (Database Type) ให้เหมาะสมกับลักษณะของข้อมูลและลักษณะการใช้งาน (Workload) ของแต่ละฟังก์ชันในระบบ เพื่อให้สามารถตอบสนองความต้องการเฉพาะด้านของงานได้อย่างมีประสิทธิภาพ โดยต้องพิจารณาใช้ประเภทฐานข้อมูลที่แตกต่างกันตามลักษณะการใช้งาน ทั้งนี้ การเลือกใช้งานต้องสามารถให้เหตุผลประกอบที่ชัดเจน และสามารถตรวจสอบได้ตามหลักเกณฑ์ที่ยอมรับในอุตสาหกรรม ต้องยึดหลักเกณฑ์อย่างน้อยดังต่อไปนี้

๓.๑๕.๓. ประเภทฐานข้อมูลและระบบจัดเก็บ/บริหารข้อมูล (Database and Data Management Technologies)

ผู้รับจ้างต้องเลือกใช้ประเภทฐานข้อมูล ระบบจัดเก็บข้อมูล หรือองค์ประกอบสำหรับการจัดเก็บและบริหารข้อมูล (Data Storage and Data Management

Components) ให้เหมาะสมกับลักษณะของข้อมูล ลักษณะการใช้งาน (Workload) และความต้องการของแต่ละฟังก์ชันในระบบ เพื่อให้สามารถรองรับการทำงานได้อย่างมีประสิทธิภาพ รวมถึงสามารถรองรับการขยายตัวของข้อมูล ผู้ใช้งาน และภาระการประมวลผลในอนาคต

การเลือกใช้เทคโนโลยีแต่ละประเภทต้องพิจารณาจากลักษณะของข้อมูล รูปแบบการอ่านและเขียนข้อมูล ปริมาณข้อมูล ความถี่ในการเข้าถึง ความต้องการด้านประสิทธิภาพ ความสามารถในการขยายระบบ ความมั่นคงปลอดภัย และความเหมาะสมกับสถาปัตยกรรมโดยรวม โดยผู้รับจ้างต้องสามารถอธิบายเหตุผลและความเหมาะสมของการเลือกใช้เทคโนโลยีดังกล่าวได้อย่างชัดเจน และสามารถตรวจสอบย้อนกลับได้

ทั้งนี้ ต้องพิจารณาใช้ประเภทฐานข้อมูลหรือองค์ประกอบด้านการจัดเก็บและบริหารข้อมูลให้เหมาะสมกับลักษณะงาน อย่างน้อยดังต่อไปนี้

๓.๑๕.๓.๑. Relational Database (SQL)

๑) ใช้สำหรับระบบที่มีข้อมูลโครงสร้างชัดเจน (Structured Data) เช่น ระบบจัดการผู้ใช้งาน ระบบสิทธิ์ ระบบการแจ้งเตือน

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

๓) เช่น MySQL, PostgreSQL

๓.๑๕.๓.๒. In-memory Store

๑) ใช้สำหรับข้อมูลที่มีการเข้าถึงบ่อยและต้องการความเร็วสูง เช่น Session, Flash Sale, Caching

๒) ควรใช้ร่วมกับกลไก Expiration/TTL เพื่อจัดการหน่วยความจำ

๓) เช่น Redis, Memcached

๓.๑๕.๓.๓. Time-series Database

๑) ใช้สำหรับการเก็บข้อมูลตามเวลา เช่น Stock Market, Sensor, Monitoring

๒) ต้องรองรับฟีเจอร์การเก็บตามเวลา เช่น Retention Policy, Rollup, Compression

๓) เช่น InfluxDB, TimescaleDB, Prometheus, Graphite

๓.๑๕.๓.๔. Graph Database

๑) ใช้สำหรับข้อมูลที่มีความสัมพันธ์ซับซ้อน เช่น ความสัมพันธ์ผู้ใช้งาน, Fraud Detection, Recommendation Engine

๒) ต้องสามารถ Query ข้าม Node ได้อย่างมีประสิทธิภาพ

๓) เช่น Neo4j, Amazon Neptune

๓.๑๕.๓.๕. Document Store

- ๑) เหมาะกับข้อมูลที่ไม่แน่นอนและมีขนาดใหญ่ เช่น Metadata, ไฟล์ JSON, ไฟล์ภาพ
- ๒) เหมาะกับระบบที่ไม่ต้อง Join หลายตาราง เช่น CMS, ระบบจัดเก็บเอกสาร

๓) เช่น MongoDB, CouchDB

๓.๑๕.๓.๖. 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

๓.๑๕.๓.๗. เจือปนไขว้ไปเพิ่มเติม

- ๑) ต้องเลือกประเภทฐานข้อมูลตามลักษณะข้อมูล ไม่ใช่ฐานข้อมูลประเภทเดียวสำหรับทุก Use Case โดยไม่วิเคราะห์
- ๒) หากใช้ NoSQL หรือ Time-Series ต้องมีเหตุผลประกอบด้านประสิทธิภาพ ปริมาณข้อมูล หรือรูปแบบการเข้าถึง

๓.๑๕.๓.๘. Feature Store

- ๑) ในกรณีที่มีการพัฒนาหรือใช้งานแบบจำลอง Machine Learning หรือแบบจำลองอื่นที่มีการใช้ Feature ในกระบวนการฝึกสอนหรือการประมวลผล ผู้รับจ้างต้องพิจารณาใช้ Feature Store หรือกลไกการบริหารจัดการ Feature ที่เหมาะสม เพื่อสนับสนุนการจัดเก็บ บริหาร และนำ Feature กลับมาใช้ร่วมกันระหว่างกระบวนการ Model Training และ Model Inference ตามลักษณะการใช้งาน
- ๒) ต้องรองรับการบริหารจัดการ Feature และ Metadata ที่เกี่ยวข้อง รวมถึงแหล่งที่มา Version สิทธิการเข้าถึง คุณภาพของ Feature และการนำ Feature กลับมาใช้ซ้ำได้ตามความเหมาะสม
- ๓) ต้องสามารถเชื่อมโยงกับองค์ประกอบที่เกี่ยวข้อง เช่น Data Processing, Data Storage, MLOps Platform และ Model Serving เพื่อสนับสนุนการบริหารจัดการ Feature ตลอดวงจรชีวิตของแบบจำลอง
- ๔) การเลือกใช้ Feature Store หรือกลไกการบริหารจัดการ Feature ต้องพิจารณาความเหมาะสมกับลักษณะข้อมูล แบบจำลอง และ Workload โดยต้องสามารถตรวจสอบย้อนกลับแหล่งที่มา กระบวนการสร้าง และ Version ของ Feature ได้ตามความเหมาะสม

๕) รายละเอียดด้านการบริหารจัดการ Feature ได้แก่ Feature Metadata, Feature Version, Feature Lineage การควบคุมสิทธิการเข้าถึง การติดตามคุณภาพของ Feature การนำ Feature กลับมาใช้ซ้ำ และการนำ Feature ไปใช้ในกระบวนการ Model Training และ Model Inference ให้เป็นไปตามภาคผนวก ง ข้อ ๒.๗ การบริหารจัดการ Feature สำหรับแบบจำลอง (Feature Management) โดยผู้รับจ้างสามารถเลือกใช้เทคโนโลยี ผลิตภัณฑ์ ซอฟต์แวร์โอเพนซอร์ส หรือกลไกอื่นที่มีคุณสมบัติเทียบเท่าและเหมาะสมกับสถาปัตยกรรม ลักษณะข้อมูล และลักษณะการใช้งานของระบบ

๓.๑๕.๓.๙. Vector Database / Vector Store

๑) สำหรับกรณีใช้งานที่ต้องจัดเก็บและบริหารข้อมูลเชิงเวกเตอร์ (Vector Data) หรือ Embedding ผู้รับจ้างต้องพิจารณาใช้ Vector Database, Vector Store หรือเทคโนโลยีที่มีความสามารถเทียบเท่าและเหมาะสม เพื่อรองรับการสืบค้นข้อมูลด้วย Similarity Search, Semantic Search, Retrieval-Augmented Generation (RAG) หรือกรณีใช้งานด้าน AI อื่นตามขอบเขตของระบบ

๒) ต้องรองรับการจัดเก็บและบริหาร Vector Data, Embedding และ Metadata ที่เกี่ยวข้อง รวมถึงการบริหาร Collection, Index หรือโครงสร้างอื่นที่ใช้สำหรับการสืบค้นตามลักษณะของเทคโนโลยีที่เลือก

๓) ต้องสามารถทำงานร่วมกับแหล่งข้อมูลหรือฐานความรู้ กระบวนการสร้าง Embedding ระบบ RAG, AI Assistant และองค์ประกอบด้านการประมวลผลปัญญาประดิษฐ์ที่เกี่ยวข้องตามสถาปัตยกรรมของระบบ

๔) ต้องรองรับข้อมูลและ Metadata ที่จำเป็นต่อการควบคุม Version และการตรวจสอบย้อนกลับขององค์ประกอบที่เกี่ยวข้องตามความเหมาะสม เช่น Embedding Model Version, Vector Index Version, Document/Knowledge Base Version และ Retrieval Configuration

๕) การควบคุมการสืบค้นและการนำข้อมูลจาก Vector Database หรือ Vector Store ไปใช้ในการประมวลผล ต้องสอดคล้องกับสิทธิการเข้าถึงของผู้ใช้งานและระบบต้นทาง โดยรายละเอียดด้านการสืบค้นข้อมูลตามสิทธิ (Access-aware Retrieval) ให้เป็นไปตามภาคผนวก ง ข้อ ๘.๓

๖) รายละเอียดด้านการใช้งาน Vector Data, Embedding, RAG, AI Processing Pipeline, Versioning และ Traceability ให้เป็นไปตามภาคผนวก ง ข้อ ๒.๘ ข้อ ๓.๔ ข้อ ๕.๗ และ ข้อ ๕.๘ ตามที่เกี่ยวข้อง

๗) ทั้งนี้ สามารถใช้เทคโนโลยีหรือผลิตภัณฑ์ที่มีคุณสมบัติเหมาะสมกับสถาปัตยกรรม ปริมาณข้อมูล ประสิทธิภาพ และลักษณะการใช้งานของระบบ

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

๓.๑๕.๔.๑. การควบคุมเวอร์ชันของฐานข้อมูล

- ๑) ต้องใช้เครื่องมือสำหรับการควบคุมเวอร์ชันของฐานข้อมูล เช่น Liquibase หรือเครื่องมือที่มีคุณสมบัติเทียบเท่า เพื่อจัดการการเปลี่ยนแปลงของโครงสร้างฐานข้อมูลอย่างเป็นระบบ
- ๒) การเปลี่ยนแปลงของฐานข้อมูลต้องถูกบันทึกในรูปแบบของไฟล์ที่สามารถควบคุมเวอร์ชันได้ เช่น SQL, XML, YAML และจัดเก็บร่วมกับซอร์สโค้ดของแอปพลิเคชันในระบบ Version Control เช่น Git
- ๓) ต้องสามารถติดตามประวัติการเปลี่ยนแปลงของฐานข้อมูลได้อย่างชัดเจน และสามารถย้อนกลับ (Rollback) ไปยังสถานะก่อนหน้าได้ในกรณีที่เกิดปัญหา

๓.๑๕.๔.๒. การปฏิบัติตามแนวทางมาตรฐาน

- ๑) การดำเนินการเปลี่ยนแปลงฐานข้อมูลต้องเป็นไปตามแนวทางปฏิบัติที่ยอมรับในระดับสากล เช่น การแยกการเปลี่ยนแปลงโครงสร้างฐานข้อมูลออกจาก การเปลี่ยนแปลงข้อมูล (Data Migration) และการจัดการการเปลี่ยนแปลงอย่างเป็นระบบ (Database Schema Migration Best Practices)
- ๒) การรวมกระบวนการเปลี่ยนแปลงฐานข้อมูลเข้ากับกระบวนการ CI/CD เพื่อให้สามารถปรับใช้การเปลี่ยนแปลงได้อย่างรวดเร็วและปลอดภัย (Continuous Integration/Continuous Deployment (CI/CD))
- ๓) การประสานงานระหว่างทีมพัฒนาและทีมปฏิบัติการเพื่อให้การเปลี่ยนแปลงฐานข้อมูลเป็นไปอย่างราบรื่น (DevSecOps Practices)

๓.๑๕.๔.๓. การใช้งาน Database Middleware เพื่อแยกคำสั่งอ่าน-เขียน (Read-Write Splitting) และจัดการ Replica ผู้รับจ้างต้องออกแบบและติดตั้งระบบฐานข้อมูลให้สามารถทำงานร่วมกับ Database Middleware เพื่อแยกการประมวลผลคำสั่งอ่านและเขียนออกจากกันอย่างมีประสิทธิภาพ รองรับการกระจายโหลด (Load Balancing) และเพิ่มความสามารถในการรองรับการขยายระบบและความทนทานของระบบฐานข้อมูล โดยมีข้อกำหนดอย่างน้อยดังต่อไปนี้

๓.๑๕.๔.๔. คุณสมบัติของ Database Middleware

- ๑) ทำหน้าที่เป็นชั้นกลาง (Middleware Layer) ระหว่างระบบ Application กับระบบฐานข้อมูล (Database Cluster)
 - ๒) รองรับการสื่อสารผ่าน Database Network Protocol ที่เป็นมาตรฐานและสามารถใช้งานร่วมกับโพรโทคอลที่มีอยู่ได้ เช่น TCP-based protocol ของฐานข้อมูลที่ใช้
 - ๓) สามารถตั้งกฎการส่งต่อ (Routing Rules) ตามประเภทคำสั่ง เช่น
- ๑) คำสั่งเขียน (INSERT, UPDATE, DELETE) → ไปยัง Primary Database

ที่เหมาะสม

๒) คำสั่งอ่าน (SELECT) → ไปยัง Replica Database

๔) รองรับการกำหนด Routing อย่างน้อยดังนี้

๑) ประเภทคำสั่ง (Statement Type)

๒) ผู้ใช้งาน (User Context)

๓) ฐานข้อมูลหรือสคีมา (Schema)

๔) ปริมาณโหลด (Connection Load)

๓.๑๕.๔.๕. 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

๓.๑๕.๔.๖. 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

๓.๑๖ OpenAPI

๓.๑๖.๑. การออกแบบและจัดเก็บ OpenAPI Specification

๓.๑๖.๑.๑. ผู้รับจ้างต้องออกแบบ API โดยใช้ OpenAPI Specification เวอร์ชัน 3.0 ขึ้นไป เป็นมาตรฐานหลัก

๓.๑๖.๑.๒. ผู้รับจ้างต้องจัดทำ ไฟล์ .yaml หรือ .json ของ OpenAPI สำหรับทุก API Service

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

- ๓.๑๖.๑.๔. ออกแบบ API ให้สอดคล้องและรองรับสถาปัตยกรรม
Microservices
- ๓.๑๖.๒. ข้อกำหนดด้านการจัดเก็บและบริหารจัดการ API Specification
- ๓.๑๖.๒.๑. จัดเก็บ API Specification ใน Git-based Repository โดยแยกออกจากซอร์สโค้ดหลัก
- ๓.๑๖.๒.๒. ต้องจัดการเวอร์ชันของ Specification ด้วยวิธี Branching Strategy ที่ชัดเจน เช่น Git Flow
- ๓.๑๖.๓. ข้อกำหนดด้านการแก้ไขและปรับปรุง API Specification
- ๓.๑๖.๓.๑. ใช้เครื่องมือ Visual Editor (เช่น Swagger Editor, Stoplight Studio หรือเครื่องมือเทียบเท่า) เพื่ออำนวยความสะดวกในการปรับแก้ Specification
- ๓.๑๖.๓.๒. ต้องมี Workflow ในการจัดการและตรวจสอบคุณภาพ Specification เช่น การใช้ OpenAPI Validator, Spectral หรือเครื่องมือที่เทียบเท่า
- ๓.๑๖.๔. ข้อกำหนดด้านการนำ API Specification ไปใช้ (API Deployment)
- ๓.๑๖.๔.๑. มีแนวทางหรือเครื่องมือในการนำ Specification ไปใช้งานจริงโดยไม่ต้องแก้ไขหรือคอมไพล์ซอร์สโค้ดหลัก เช่น API Gateway, Auto-Generation Framework, Mock Server หรือ Low-Code Platform
- ๓.๑๖.๔.๒. Pipeline CI/CD ต้องรองรับการอ่านและ Deploy Specification ได้แบบอัตโนมัติและรองรับ Zero Downtime Deployment
- ๓.๑๖.๕. ข้อกำหนดการเปลี่ยนแปลง API (Change Management)
- ๓.๑๖.๕.๑. ต้องกำหนดแนวทางการจัดการ Breaking Change อย่างชัดเจน พร้อมการทำ Diff Report ของ API Specification เวอร์ชันใหม่และเก่า
- ๓.๑๖.๕.๒. ระบบต้องรองรับการ Synchronize Specification ใหม่ไปยัง API Gateway และ Mock Server โดยอัตโนมัติ
- ๓.๑๖.๖. ข้อกำหนดด้านเอกสารประกอบและตัวอย่างการใช้งาน API (API Documentation)
- ๓.๑๖.๖.๑. ต้องจัดทำเอกสารในรูปแบบ Swagger UI หรือ ReDoc พร้อมทั้งตัวอย่างโค้ดครบถ้วน (เช่น Curl, Postman Collection, SDK Stub)
- ๓.๑๖.๖.๒. เอกสารต้องระบุขั้นตอนการปรับปรุง Specification อย่างละเอียดและชัดเจน

๓.๑๖.๗. ข้อกำหนดเกี่ยวกับซอฟต์แวร์และลิขสิทธิ์ (Software Licensing)

๓.๑๖.๗.๑. เครื่องมือและเทคโนโลยีที่นำมาใช้ต้องเป็น Open Source หรือมีลิขสิทธิ์แบบ Perpetual License ยกเว้นได้รับอนุมัติจากผู้ว่าจ้าง

๓.๑๖.๗.๒. ข้อกำหนดด้านการทำงานแบบอัตโนมัติและ DevSecOps (Automation & DevSecOps Requirements)

๓.๑๖.๘. 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

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

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

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

๓.๑๖.๑๒. กระบวนการ Deployment จะต้องรองรับการทำ Zero Downtime Deployment และสามารถ Rollback กลับไปเวอร์ชันก่อนหน้าได้อย่างอัตโนมัติและรวดเร็ว

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

๓.๑๖.๑๔. ข้อกำหนดเพิ่มเติมด้านความปลอดภัยและประสิทธิภาพ (Security and Performance)

๓.๑๖.๑๔.๑. ระบบต้องรองรับ OAuth 2.0, JWT, OWASP API Top 10 และแนวทาง Secure Coding Practices

๓.๑๖.๑๔.๒. มีการวัดประสิทธิภาพ API ด้วย Prometheus หรือ OpenTelemetry

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

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

๓.๑๗.๑. ลักษณะของ Object Storage ที่ต้องรองรับอย่างน้อยดังนี้

๓.๑๗.๑.๑. ต้องจัดเก็บข้อมูลในลักษณะ “Object” โดยแต่ละ Object ต้อง
ประกอบด้วยข้อมูลอย่างน้อยดังนี้

- ๑) Data Payload
- ๒) Metadata (Custom Metadata ได้)
- ๓) Unique Identifier (Object ID / URI)

๓.๑๗.๑.๒. ต้องรองรับโครงสร้างแบบ Flat Structure (ไม่มี Directory
Hierarchy)

๓.๑๗.๑.๓. ต้องรองรับการเข้าถึงผ่าน RESTful API ได้

๓.๑๗.๑.๔. ความเหมาะสมกับงาน Microservices

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

๓.๑๗.๑.๖. ต้องออกแบบให้รองรับการใช้งานร่วมกับระบบ Cloud Native และ
Containerized Services เช่น Kubernetes

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

๓.๑๗.๒. ความปลอดภัยในการเข้าถึง (Authentication & Authorization)

๓.๑๗.๒.๑. การเข้าถึง Object Store ต้องรองรับมาตรฐานความปลอดภัย ได้แก่

- ๑) Token-based Access (เช่น OAuth 2.0, JWT)
- ๒) Pre-signed URL สำหรับการอนุญาตแบบจำกัดเวลา
- ๓) Access Control Policy (ACL) และ Bucket Policy

๓.๑๗.๒.๒. ต้องมีการแยกสิทธิ์ ระหว่าง Service และ User
แต่ละรายอย่างชัดเจน (Least Privilege)

๓.๑๗.๓. มาตรฐานและผลิตภัณฑ์ที่ยอมรับ

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

๓.๑๗.๓.๒. มาตรฐานอ้างอิง

๓.๑๗.๓.๓. NIST 800-209 (Security Guidelines for Storage Infrastructure)

๓.๑๗.๓.๔. CNCF Cloud Native Storage Principles

๓.๑๗.๔. รองรับ Use Cases อย่างน้อยดังนี้

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

๓.๑๗.๔.๒. การเก็บข้อมูลแบบ Data Archiving

๓.๑๗.๔.๓. การทำ Cloud-Native Storage สำหรับระบบ Microservices

๓.๑๗.๔.๔. การเก็บข้อมูลจาก IoT หรือ Sensor (Internet of Things)

๓.๑๗.๔.๕. รองรับระบบ Backup & Recovery

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

๓.๑๗.๔.๗. การจัดเก็บข้อมูลที่เกี่ยวข้องกับกระบวนการตรวจสอบและการประมวลผลด้วยปัญญาประดิษฐ์ เช่น หลักฐานการตรวจสอบ (Audit Evidence) เอกสารหรือไฟล์ต้นฉบับ ผลลัพธ์จากกระบวนการ OCR และข้อมูลที่ใช้เป็นแหล่งข้อมูลสำหรับระบบ Retrieval-Augmented Generation (RAG)

๓.๑๗.๔.๘. ต้องสามารถเชื่อมโยงและตรวจสอบย้อนกลับ (Traceability) ระหว่างไฟล์หรือเอกสารต้นฉบับ หลักฐานการตรวจสอบ ผลลัพธ์จากกระบวนการ OCR ข้อมูลที่ผ่านการจัดเตรียมสำหรับ AI/ML และผลลัพธ์ที่เกิดจากกระบวนการประมวลผลที่เกี่ยวข้องได้ตามความเหมาะสม โดยต้องสามารถระบุแหล่งที่มา (Source) และข้อมูลประกอบ (Metadata) ที่จำเป็นสำหรับการตรวจสอบย้อนหลังได้

๓.๑๗.๔.๙. รองรับการจัดเก็บหลักฐานการตรวจสอบ เอกสารหรือไฟล์ต้นฉบับ ผลลัพธ์ OCR/IDP ข้อมูลหรือเอกสารสำหรับ RAG และข้อมูลหรือผลลัพธ์อื่นที่เกี่ยวข้อง

๓.๑๗.๔.๑๐. ระบบจัดเก็บข้อมูลต้องรองรับ Metadata และข้อมูลอ้างอิงที่จำเป็นเพื่อสนับสนุนการเชื่อมโยงและตรวจสอบย้อนกลับระหว่างข้อมูลหรือเอกสารต้นทาง ข้อมูลที่ผ่านการประมวลผล และผลลัพธ์ที่เกี่ยวข้อง ทั้งนี้ รายละเอียดด้าน Data Lineage และ AI Traceability ให้เป็นไปตามภาคผนวก ง ข้อ ๕.๗ และข้อ ๕.๘

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

๓.๑๘.๑. ดัชนีชี้วัดประสิทธิภาพหลัก (Core Performance Metrics)

๓.๑๘.๑.๑. Load Time

- ๑) ที่ใช้ในการโหลดหน้าเว็บทั้งหมดจนสามารถแสดงผลได้
- ๒) ต้องใช้เครื่องมือวัด เช่น Lighthouse, WebPageTest

หรือเทียบเท่า

๓.๑๘.๑.๒. Time to First Byte (TTFB)

- ๑) ระยะเวลาที่เบราว์เซอร์ได้รับไบนารีข้อมูลแรกจากเซิร์ฟเวอร์
- ๒) ต้องมีค่า < 500ms สำหรับผู้ใช้งานในภูมิภาคเป้าหมาย

๓.๑๘.๑.๓. Request Count

- ๑) จำนวนคำขอ (HTTP Requests) ที่ต้องใช้เพื่อโหลดหน้าเว็บ
- ๒) ควรมีการรวมไฟล์หรือปรับลดเพื่อไม่เกินเกณฑ์ ๗๐-๘๐

requests ต่อหน้า

๓.๑๘.๑.๔. DOMContentLoaded (DCL)

- ๑) เวลาที่ HTML โหลดครบ ไม่รวม CSS/JS/รูปภาพ
- ๒) ใช้สำหรับประเมินความพร้อมใช้งานเบื้องต้นของโครงสร้าง DOM

๓.๑๘.๑.๕. Time to Above-the-Fold Load

- ๑) เวลาที่เนื้อหาบนสุดของเว็บเพจ (ก่อนการ scroll) ปรากฏครบ
- ๒) ควรน้อยกว่า ๒ วินาที

๓.๑๘.๑.๖. First Contentful Paint (FCP)

- ๑) เวลาแรกที่เบราว์เซอร์แสดงเนื้อหา เช่น ข้อความหรือรูปภาพ
- ๒) เป้าหมาย < ๑.๘ วินาที

๓.๑๘.๑.๗. Page Size

- ๑) ขนาดรวมของ HTML, CSS, JS, ภาพ และ assets ทั้งหมด
- ๒) ควรไม่เกิน ๒.๕ MB ต่อหน้า หรือมีเทคนิค Lazy Loading

๓.๑๘.๒. ข้อกำหนดเพิ่มเติม

๓.๑๘.๒.๑. ต้องแสดงผลลัพธ์จากเครื่องมือ Performance Audit อย่างน้อย ๒

เครื่องมือ พร้อมค่าเปรียบเทียบก่อน/หลังการปรับปรุง

๓.๑๘.๒.๒. จัดทำรายงานการวัดผลพร้อมคำอธิบายแนวทางการปรับปรุง

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

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

๓.๒๐ ผู้รับจ้างต้องจัดทำและส่งมอบบัญชีรายการส่วนประกอบซอฟต์แวร์ (Software Bill of Materials: SBOM) ของระบบและองค์ประกอบที่เกี่ยวข้อง โดยครอบคลุมอย่างน้อยซอฟต์แวร์ ไลบรารี, โมเดลปัญญาประดิษฐ์ (AI Model), Package, Dependency, Container Image, องค์ประกอบของบุคคลภายนอกหรือซอฟต์แวร์โอเพนซอร์ส (Third-party/Open-source Components) และองค์ประกอบอื่นที่เกี่ยวข้องกับการให้บริการของระบบ พร้อมข้อมูลชื่อหรือรหัสส่วนประกอบ เวอร์ชัน แหล่งที่มา และสิทธิการใช้งาน (License) ตามความเหมาะสม เพื่อให้สามารถตรวจสอบ ติดตามช่องโหว่ (Vulnerability) และตรวจสอบการปฏิบัติตามเงื่อนไขสิทธิการใช้งานซอฟต์แวร์ได้

การจัดทำและปรับปรุง SBOM ต้องใช้เครื่องมือ Software Composition Analysis (SCA) หรือเครื่องมือหรือวิธีการอื่นที่มีคุณสมบัติเทียบเท่า และต้องปรับปรุงข้อมูลให้เป็นปัจจุบันเมื่อมีการเปลี่ยนแปลงส่วนประกอบที่เกี่ยวข้อง ทั้งนี้ ให้สอดคล้องกับข้อกำหนดด้าน Software License Compliance, Vulnerability Management และ DevSecOps ที่กำหนดไว้ในภาคผนวกนี้ และต้องนำเสนอ SBOM ให้สำนักงานการตรวจเงินแผ่นดินให้ความเห็นชอบ

สำหรับแบบจำลองปัญญาประดิษฐ์และองค์ประกอบด้านปัญญาประดิษฐ์ที่นำมาใช้ในระบบ ต้องสามารถเชื่อมโยงข้อมูลส่วนประกอบที่เกี่ยวข้องกับทะเบียนระบบปัญญาประดิษฐ์ (AI System Inventory) และทะเบียนแบบจำลอง (Model Registry) ตามภาคผนวก ง ข้อ ๕.๒ และข้อ ๕.๓ ตามที่เกี่ยวข้อง และในกรณีที่แบบจำลอง บริการ หรือองค์ประกอบด้านปัญญาประดิษฐ์ของบุคคลภายนอก ให้เป็นไปตามข้อกำหนดด้านการบริหารความเสี่ยงจากองค์ประกอบ AI ของบุคคลภายนอกตามภาคผนวก ง ข้อ ๙.๕

๓.๒๑ การออกแบบตามข้อ ๓.๑ ถึงข้อ ๓.๒๐ 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

๔.๗ ทดสอบแนวคิดด้านคุณภาพ ความถูกต้อง และความน่าเชื่อถือขององค์ประกอบปัญญาประดิษฐ์

๔.๗.๑. สาธิตการทำงานขององค์ประกอบปัญญาประดิษฐ์ด้วยชุดข้อมูลหรือชุดกรณีทดสอบที่สำนักงานการตรวจเงินแผ่นดินกำหนดหรือให้ความเห็นชอบ โดยอย่างน้อยต้องครอบคลุม OCR/IDP สำหรับการอ่านและสกัดข้อมูลจากเอกสาร Rule Engine สำหรับการตรวจสอบเงื่อนไขหรือการกระหายอดข้อมูล Anomaly Detection สำหรับการตรวจจับความผิดปกติของข้อมูล และ Risk Scoring สำหรับการประเมินและจัดลำดับความเสี่ยงตามกรณีใช้งานที่กำหนด

๔.๗.๒. เกณฑ์ผ่าน: ผลการประเมินของแต่ละองค์ประกอบต้องเป็นไปตามตัวชี้วัด วิธีการประเมิน และให้เป็นไปตามภาคผนวก ง ข้อ ๔.๑.๒ ข้อ ๔.๑.๓ ข้อ ๔.๑.๔ และข้อ ๔.๑.๗ ตามลำดับ

๔.๘ RAG, Generative AI และ AI Assistant

๔.๘.๑. สาธิตการสืบค้นข้อมูล เอกสาร หรือหลักฐานจากชุดเอกสารหรือฐานความรู้ที่สำนักงานการตรวจเงินแผ่นดินกำหนด การตอบคำถามโดยอ้างอิงข้อมูลหรือหลักฐานต้นทาง และการใช้งาน AI Assistant เพื่อสนับสนุนการวิเคราะห์ การสรุป หรือการจัดทำร่างเนื้อหาตามกรณีใช้งานที่กำหนด

๔.๘.๒. เกณฑ์ผ่าน: ระบบต้องสามารถสืบค้นข้อมูลหรือหลักฐานที่เกี่ยวข้อง เชื่อมโยงคำตอบหรือผลลัพธ์กับข้อมูลหรือหลักฐานต้นทาง แสดงแหล่งอ้างอิง และจัดการกรณีที่ไม่พบข้อมูลหรือหลักฐานเพียงพอได้ตามเกณฑ์การประเมินที่กำหนดไว้ในภาคผนวก ง ข้อ ๔.๑.๕

๔.๙ ทดสอบแนวคิดด้านมาตรการควบคุมและความมั่นคงปลอดภัยของระบบปัญญาประดิษฐ์ (AI Guardrails and AI Security)

๔.๙.๑. สาธิตและทดสอบมาตรการควบคุมของระบบปัญญาประดิษฐ์ โดยใช้ชุดกรณีทดสอบที่สำนักงานการตรวจเงินแผ่นดินกำหนดหรือให้ความเห็นชอบ และให้ครอบคลุมความเสี่ยงตามลักษณะขององค์ประกอบที่นำมาใช้ เช่น Prompt Injection, Jailbreak, Data Leakage, Cross-permission Retrieval, Unauthorized Tool Use และ Indirect Prompt Injection ตามความเหมาะสม

๔.๙.๒. เกณฑ์ผ่าน: ระบบต้องสามารถบังคับใช้มาตรการควบคุมตามชุดกรณีทดสอบที่กำหนดหรือได้รับความเห็นชอบ และต้องไม่สามารถเข้าถึง ประมวลผล หรือเปิดเผยข้อมูลที่อยู่นอกเหนือสิทธิของผู้ใช้งานในกรณีทดสอบดังกล่าวได้ ทั้งนี้ ให้เป็นไปตามภาคผนวก ง ข้อ ๘.๑ ข้อ ๘.๓ และข้อ ๘.๔ ตามที่เกี่ยวข้อง

๔.๑๐ ทดสอบแนวคิดด้านการกำกับดูแลโดยมนุษย์ (Human Oversight / Human-in-the-loop)

๔.๑๐.๑. สถิติการแสดงผลสถานะหรือข้อความกำกับผลลัพธ์ที่สร้างหรือได้รับการสนับสนุนจากระบบปัญญาประดิษฐ์ การกำหนดจุดตรวจทานโดยมนุษย์ (Human Review Point) กระบวนการตรวจทาน แก้ไข ยืนยัน รับรอง หรืออนุมัติโดยผู้มีสิทธิ และการบันทึกประวัติการดำเนินการเพื่อให้สามารถตรวจสอบย้อนหลังได้

๔.๑๐.๒. เกณฑ์ผ่าน: ผลลัพธ์ที่สำนักงานการตรวจเงินแผ่นดินกำหนดให้ต้องผ่านการตรวจทานหรืออนุมัติ ต้องไม่สามารถข้ามกระบวนการดังกล่าวหรือนำไปใช้เป็นผลลัพธ์ที่ได้รับการรับรองก่อนผ่านการตรวจทานตามสิทธิและลำดับขั้นที่กำหนด และต้องสามารถตรวจสอบย้อนหลังผู้ดำเนินการสถานะ ผลการพิจารณา และวันเวลาได้ ทั้งนี้ให้เป็นไปตามภาคผนวก ง ข้อ ๖.๒ ข้อ ๖.๓ ข้อ ๖.๔ และข้อ ๖.๕ ตามที่เกี่ยวข้อง

๔.๑๑ ทดสอบแนวคิดด้านการปกป้องข้อมูลและการควบคุมการใช้ AI ภายในและภายนอก

๔.๑๑.๑. สถิติการควบคุมการเลือกใช้ Private AI/Local LLM และ External AI/External LLM ตามประเภทข้อมูล ระดับชั้นความลับ สิทธิของผู้ใช้งาน และกรณีใช้งาน รวมถึงการปกปิดหรือลดทอนข้อมูล (Data Masking, Anonymization หรือวิธีการอื่นที่เหมาะสม) และการควบคุมการสืบค้นข้อมูลตามสิทธิ (Access-aware Retrieval)

๔.๑๑.๒. เกณฑ์ผ่าน: ระบบต้องสามารถควบคุมการเลือกใช้ Private AI/Local LLM และ External AI/External LLM ตามประเภทข้อมูล ระดับชั้นความลับ สิทธิ และนโยบายที่กำหนด โดยข้อมูลที่ไม่ได้รับอนุญาตให้ส่งออกไปประมวลผลภายนอกต้องไม่สามารถถูกส่งออกได้ และการสืบค้นหรือนำข้อมูลไปใช้ต้องไม่เกินสิทธิของผู้ใช้งาน ทั้งนี้ให้เป็นไปตามภาคผนวก ง ข้อ ๘.๒ และข้อ ๘.๓

๔.๑๒ ทดสอบแนวคิดด้านสมรรถนะการประมวลผลปัญญาประดิษฐ์แบบ On-Premise

๔.๑๒.๑. ทดสอบสมรรถนะของ AI Workload ที่เกี่ยวข้องกับขอบเขตโครงการ เช่น OCR Throughput, Embedding Generation, LLM Inference Latency และ Vector Search โดยใช้ปริมาณข้อมูล จำนวนคำขอพร้อมกัน และสภาพแวดล้อมการประมวลผลที่กำหนดหรือได้รับความเห็นชอบ

๔.๑๒.๒. เกณฑ์ผ่าน: ผลการทดสอบต้องเป็นไปตามค่าเป้าหมายและเกณฑ์ผ่านของ Non-Functional Requirements และ Performance Test Plan ที่ผู้ว่าจ้างให้ความเห็นชอบ โดยต้องแสดง Test Environment, Hardware/Resource Configuration, Workload, Concurrent Requests, Throughput, Latency และ Resource Utilization ที่ใช้ในการทดสอบ ทั้งนี้ให้สอดคล้องกับข้อ ๖.๕.๓ ถึงข้อ ๖.๕.๕ และข้อ ๖.๕.๙ ตามที่เกี่ยวข้อง

๔.๑๓ ทดสอบแนวคิดด้าน MLOps และการบริหารวงจรชีวิตแบบจำลอง

๔.๑๓.๑. สถิติการบริหารจัดการแบบจำลองด้วย Model Registry การควบคุมเวอร์ชันของแบบจำลอง (Model Versioning) การนำแบบจำลองขึ้นให้บริการ (Model Deployment) การย้อนกลับไปใช้แบบจำลองเวอร์ชันก่อนหน้า (Rollback) และการติดตาม Data Drift หรือการเปลี่ยนแปลงของประสิทธิภาพแบบจำลองภายหลังจากนำไปใช้งานจริงตามความเหมาะสมกับประเภทของแบบจำลอง

๔.๑๓.๒. เกณฑ์ผ่าน: ระบบต้องสามารถลงทะเบียนและบริหารแบบจำลอง ควบคุมและตรวจสอบเวอร์ชัน นำแบบจำลองขึ้นให้บริการ ย้อนกลับไปยังเวอร์ชันก่อนหน้าที่กำหนด และติดตามสถานะหรือการเปลี่ยนแปลงที่เกี่ยวข้องกับข้อมูลและประสิทธิภาพของแบบจำลองหลังจากนำไปใช้งานจริงได้ ทั้งนี้ ให้เป็นไปตามภาคผนวก ง ข้อ ๒.๖ ข้อ ๕.๓ ข้อ ๕.๘ และข้อ ๙.๑

๕. การพัฒนา (Implementation)

๕.๑ ผู้รับจ้างพัฒนาระบบให้เป็นไปตามรายละเอียดคุณลักษณะเฉพาะตามภาคผนวก ก รายละเอียดคุณลักษณะเฉพาะระบบสารสนเทศของโครงการ สำหรับองค์ประกอบด้าน AI ให้เป็นไปตามภาคผนวก ง การพัฒนาระบบปัญญาประดิษฐ์และการกำกับดูแลการใช้ปัญญาประดิษฐ์

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 และกระจายทรัพยากรฟิสิกส์เข้าได้อย่างมีประสิทธิภาพ โดยรองรับผู้ใช้งานพร้อมกันไม่น้อยกว่า ๔,๐๐๐ ราย พร้อมมีการตรวจสอบสถานะของ เซิร์ฟเวอร์ (Health Check) และสามารถขยายขนาดระบบได้อัตโนมัติเมื่อตรวจพบปริมาณงานที่เพิ่มขึ้น

๕.๘ ผู้รับจ้างต้องพัฒนาระบบให้สามารถในการบันทึกและวิเคราะห์ข้อมูล Log การทำงานของระบบ แบบเรียลไทม์ 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 ทั้งนี้เพื่อป้องกันช่องโหว่ที่อาจส่งผลกระทบต่อความมั่นคงปลอดภัยของระบบ โดยผู้รับจ้างจะต้องติดตามและปรับปรุงให้สอดคล้องกับแนวทางที่อัปเดตในอนาคตอย่างเหมาะสม

๕.๑๑ การทดสอบความถูกต้องในระดับหน่วยย่อย (Unit Testing) ผู้รับจ้างต้องดำเนินการทดสอบความถูกต้องของซอร์สโค้ดในระดับหน่วยย่อย (Unit Test) สำหรับระบบที่พัฒนาขึ้น โดยมีเกณฑ์ชี้วัดและมาตรฐานการตรวจรับ ดังนี้

๑) เกณฑ์ความครอบคลุมของโค้ด (Code Coverage): ต้องมีสัดส่วนความครอบคลุมของการทดสอบซอร์สโค้ดไม่น้อยกว่าร้อยละ 80 (80%) โดยคำนวณจากเฉพาะส่วนของโค้ดที่มีการเพิ่มขึ้นใหม่หรือมีการแก้ไขเปลี่ยนแปลง (New/Changed Code) เท่านั้น

๒) การครอบคลุมฟังก์ชันสำคัญ (Critical Function Coverage): นอกเหนือจากเกณฑ์เฉลี่ยร้อยละ 80 แล้ว การทดสอบต้องครอบคลุมฟังก์ชันการทำงานหลักที่เป็นส่วนวิกฤตของระบบ (Critical Functions) ทุกฟังก์ชันอย่างครบถ้วน 100% (เช่น ฟังก์ชันที่เกี่ยวข้องกับความปลอดภัย ธุรกรรมประมวลผลข้อมูลหลัก หรือการเชื่อมต่อฐานข้อมูลสำคัญ)

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 จะทำหน้าที่
โหลดข้อมูลจากฐานข้อมูลให้เองเมื่อเกิด cache miss

๒) เหมาะกับการใช้งานแบบอ่านบ่อย (read-heavy)

๓) กลยุทธ์ฝั่ง Write (Write Caching Strategies)

๕.๑๓.๑.๓. Write Around

๑) เขียนข้อมูลลงฐานข้อมูลโดยตรง โดยไม่อัปเดต Cache

๒) เหมาะกับข้อมูลที่ไม่ค่อยถูกอ่านหลังจากเขียน

๕.๑๓.๑.๔. Write Through

๑) เขียนข้อมูลลงทั้ง Cache และฐานข้อมูลพร้อมกัน

๒) เหมาะกับข้อมูลที่ต้องการความสอดคล้องสูงทันทีหลังการเขียน

๕.๑๓.๑.๕. Write Back

๑) เขียนข้อมูลลง Cache ก่อน แล้วค่อยเขียนลงฐานข้อมูลภายหลัง

เป็นรอบ (batch/interval)

๒) เหมาะกับระบบที่เน้นประสิทธิภาพการเขียนและสามารถยอมรับ

ความหน่วงเวลาในการ Persist ข้อมูลได้

๕.๑๓.๒. ข้อกำหนดทั่วไป โดยมีข้อกำหนดอย่างน้อยดังต่อไปนี้

๕.๑๓.๒.๑. กำหนดค่าการจัดการอายุของข้อมูลในแคช (Time to Live - TTL)
โดยผู้รับจ้างต้องสามารถกำหนดช่วงเวลาที่มีอายุและหมดอายุอัตโนมัติได้
เพื่อป้องกันการใช้ข้อมูลที่ล้าสมัย

๕.๑๓.๒.๒. กำหนดกลยุทธ์การล้างข้อมูลในแคช (Invalidation Strategy) และ
การ Refresh ข้อมูล โดยระบบต้องสามารถล้างหรือทำให้ข้อมูลในแคชไม่ถูกต้องเมื่อมีการอัปเดตจาก
ฐานข้อมูลต้นทาง และสามารถโหลดข้อมูลใหม่เข้ามาทดแทนโดยอัตโนมัติ

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

๕.๑๓.๒.๔. ปัญหาข้อมูลไม่ตรงกันระหว่างแคชกับฐานข้อมูล (Cache
Inconsistency) โดยผู้รับจ้างต้องออกแบบกลไกให้สามารถตรวจจับและแก้ไขข้อมูลแคชที่ไม่ตรงกับ
ฐานข้อมูลได้ เพื่อความถูกต้องของข้อมูลที่นำมาแสดง

๕.๑๓.๒.๕. สามารถดำเนินการกู้คืนข้อมูลในกรณีที่เกิดปัญหา (Recovery) ระบบ
ต้องสามารถฟื้นฟูข้อมูลจากแคช หรือเรียกข้อมูลจากแหล่งต้นทางในกรณีที่ระบบแคชล่มหรือสูญหาย

๕.๑๓.๒.๖. สามารถเลือกใช้กลยุทธ์การแคชแบบผสมผสานตามความเหมาะสม เช่น Cache Aside และ Write Through

๕.๑๓.๒.๗. Cache Aside: ดึงข้อมูลจากฐานข้อมูลเมื่อไม่มีในแคช แล้วบันทึกเข้าแคชภายหลัง

๑) Write Through: บันทึกข้อมูลเข้าสู่แคชและฐานข้อมูลพร้อมกัน ทุกครั้งที่มีการอัปเดต

๕.๑๓.๒.๘. หากใช้เครื่องมือจัดการแคช เช่น Redis หรือเทียบเท่า ต้องมีคุณสมบัติอย่างน้อยดังต่อไปนี้

๑) รองรับการทำงานแบบ Distributed Cache เพื่อกระจายข้อมูล และโหลดการใช้งาน

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 จากโมดูลจัดการผู้ใช้ ไปยังระบบแจ้งเตือน และต่อเนื่องไปยังการส่งอีเมล เพื่อให้มั่นใจว่าข้อมูลสามารถส่งต่อกันได้ถูกต้องและเป็นไปตามลำดับการใช้งานจริง และจัดทำรายงานผลการทดสอบประกอบการส่งมอบระบบ

๖.๑.๒. ระบบจะต้องถูกตรวจสอบตามรายละเอียดคุณลักษณะเฉพาะตามภาคผนวก ก รายละเอียดคุณลักษณะของระบบสารสนเทศของโครงการ สำหรับองค์ประกอบ AI ให้ทดสอบตามเกณฑ์ในภาคผนวก ง ข้อ ๔

๖.๒ System Testing

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

๖.๒.๒. ระบบจะต้องถูกตรวจสอบตามรายละเอียดคุณลักษณะเฉพาะตามภาคผนวก ก รายละเอียดคุณลักษณะของระบบสารสนเทศของโครงการ สำหรับองค์ประกอบ AI ให้ทดสอบตามเกณฑ์ในภาคผนวก ง ข้อ ๔

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 และต้องจัดทำรายงานข้อเสนอแนะจากผู้ใช้งานปลายทาง

๖.๕ Performance Testing / Load Testing

๖.๕.๑) ระบบต้องสามารถรองรับผู้ใช้งานพร้อมกันได้ไม่น้อยกว่า ๔๐๐๐ คน โดย Response Time ต้องไม่เกิน 5 วินาทีต่อคำขอ โดยต้องแนบผลการทดสอบ Performance ด้วยเครื่องมือที่ได้มาตรฐาน (เช่น JMeter หรือเทียบเท่า)

๖.๕.๒) ผู้รับจ้างต้องจัดส่งผลการทดสอบประสิทธิภาพ (Performance Test) และเอกสารแผนการประมวลผลของคำสั่งค้นหาข้อมูล (Query Execution Plan) สำหรับ Query หลัก ประกอบการส่งมอบงาน โดยมีข้อกำหนดอย่างน้อยดังต่อไปนี้

๖.๕.๒.๑) "Query หลัก" หมายถึง Query ที่มีความถี่ในการเรียกใช้งานสูง Query ที่เกี่ยวข้อง กับตารางที่มีปริมาณข้อมูลจำนวนมาก หรือ Query ที่อยู่ในกระบวนการทางธุรกิจที่สำคัญ (Critical Business Function) ทั้งนี้ ผู้รับจ้างต้องจัดทำรายการ Query หลัก (Query Inventory) พร้อมเกณฑ์การคัดเลือก และนำเสนอให้ผู้ว่าจ้างพิจารณา เห็นชอบก่อนเริ่มดำเนินการทดสอบ

๖.๕.๒.๒) การทดสอบต้องดำเนินการบนสภาพแวดล้อมและชุดข้อมูลที่ใกล้เคียงกับการใช้งานจริง (Production-like Environment) โดยต้องระบุปริมาณข้อมูลที่ใช้ทดสอบ (Data Volume) จำนวนผู้ใช้งานพร้อมกัน (Concurrent Users) และเงื่อนไขการทดสอบ (Test Conditions) ไว้ในรายงานอย่างชัดเจน

๖.๕.๒.๓) ผลการทดสอบต้องแสดงเวลาตอบสนอง (Response Time) ของแต่ละ Query อย่างน้อย ในรูปค่าเฉลี่ย (Average) ค่าเปอร์เซ็นไทล์ที่ ๙๕ (P95) และค่าสูงสุด (Maximum) โดยเกณฑ์เวลาตอบสนองต้องสอดคล้องกับข้อกำหนดด้าน Performance Testing / Load Testing ของโครงการ

๖.๕.๒.๔) เอกสาร Execution Plan ต้องได้มาจากเครื่องมือหรือคำสั่งมาตรฐานของระบบ ฐานข้อมูลที่ใช้ เช่น EXPLAIN, EXPLAIN ANALYZE หรือเครื่องมือเทียบเท่า พร้อมคำอธิบายผลการวิเคราะห์ของแต่ละ Query

๖.๕.๒.๕) ผลการทดสอบต้องไม่พบการสแกนตารางทั้งหมด (Full Table Scan) โดยไม่มีเหตุผล อันสมควร กรณีที่มีความจำเป็นทางเทคนิค เช่น ตารางขนาดเล็ก หรือ Query เชิงวิเคราะห์ที่ต้องอ่านข้อมูลทั้งตาราง ผู้รับจ้างต้องจัดทำเอกสารชี้แจงเหตุผล ทางเทคนิค (Technical Justification) เป็นรายกรณี และต้องได้รับความเห็นชอบ จากผู้ว่าจ้างก่อนถือว่าผ่านเกณฑ์การตรวจรับ

๖.๕.๒.๖) เอกสารทั้งหมดต้องจัดส่งในรูปแบบไฟล์ PDF และไฟล์ที่แก้ไขได้ พร้อมแนบผลลัพธ์ดิบ (Raw Output) ของ Execution Plan และผลการทดสอบ เช่น Log, CSV หรือ JSON เพื่อให้สามารถตรวจสอบย้อนหลัง (Audit Trail) ได้

๖.๕.๒.๗) กรณีผลการทดสอบไม่ผ่านเกณฑ์ที่กำหนด ผู้รับจ้างต้องดำเนินการปรับปรุง ประสิทธิภาพ เช่น การปรับแต่ง Query (Query Tuning) หรือการปรับปรุงดัชนี (Index Optimization) และดำเนินการทดสอบซ้ำจนกว่าจะผ่านเกณฑ์ โดยไม่คิดค่าใช้จ่ายเพิ่มเติม

๖.๕.๓) การทดสอบสมรรถนะของโครงสร้างพื้นฐานและระบบที่รองรับงานปัญญาประดิษฐ์ (AI Infrastructure Performance Testing)

๖.๕.๓.๑) ผู้รับจ้างต้องทดสอบสมรรถนะของโครงสร้างพื้นฐานที่ใช้รองรับ ระบบปัญญาประดิษฐ์ โดยครอบคลุมอย่างน้อย CPU, GPU (ถ้ามี), Memory, Storage และ Network พร้อมจัดทำผลการทดสอบและแสดงค่าการใช้ทรัพยากรในระหว่างการทดสอบ

๖.๕.๓.๒) การทดสอบ Storage ต้องครอบคลุมค่า IOPS (Input/Output Operations Per Second), Throughput และ Storage Latency ทั้งในกรณีการอ่านและการเขียนข้อมูล เพื่อประเมินความเหมาะสมของโครงสร้างพื้นฐานสำหรับงาน AI/ML, OCR, Vector Database และการประมวลผลข้อมูลขนาดใหญ่

๖.๕.๓.๓) การทดสอบ Network ต้องแสดงค่า Network Throughput, Network Latency และ Packet Loss ในกรณีที่มีการรับส่งข้อมูลระหว่าง Application, AI/ML Services, Database, Vector Database และระบบจัดเก็บข้อมูล

๖.๕.๓.๔) ผู้รับจ้างต้องทดสอบความสามารถในการรองรับภาระงานของ AI/ML Service โดยแสดงค่าอย่างน้อย Inference Throughput, Inference Latency, Concurrent Inference Requests, Average Response Time, P95 Response Time และ Maximum Response Time ตามประเภทของ Model และบริการที่นำมาใช้งาน

๖.๕.๔) การทดสอบสมรรถนะระบบ OCR และ Document AI

๖.๕.๔.๑) ผู้รับจ้างต้องทดสอบความสามารถของระบบ OCR ในการประมวลผลเอกสาร โดยต้องแสดงค่า OCR Throughput เช่น จำนวนหน้าเอกสารที่สามารถประมวลผลได้ต่อหน่วยเวลา (Pages per Minute / Pages per Hour) และ Response Time ต่อเอกสาร

๖.๕.๔.๒) การทดสอบต้องครอบคลุมเอกสารที่มีลักษณะแตกต่างกันตามขอบเขตการใช้งานของโครงการ เช่น เอกสารภาษาไทย ภาษาอังกฤษ เอกสารที่มีตาราง เอกสารหลายรูปแบบ และเอกสารที่มีความซับซ้อนด้าน Layout

๖.๕.๔.๓) ผู้รับจ้างต้องรายงานการใช้ทรัพยากรของระบบในระหว่างการประมวลผล OCR เช่น CPU, GPU, Memory, VRAM และ Storage รวมถึงแสดง Throughput และ Latency ของแต่ละขั้นตอนที่สำคัญของ OCR Pipeline

๖.๕.๕) การทดสอบสมรรถนะระบบ RAG และ Generative AI

๖.๕.๕.๑) ผู้รับจ้างต้องทดสอบสมรรถนะของระบบ Retrieval-Augmented Generation (RAG) โดยแสดงค่าอย่างน้อย Retrieval Latency, Generation Latency, End-to-End Response Time, P95 Response Time และ Throughput

๖.๕.๕.๒) การทดสอบต้องครอบคลุมกระบวนการตั้งแต่การรับคำถาม (Query) การสืบค้นข้อมูลหรือหลักฐานจากแหล่งข้อมูลหรือกลไกการสืบค้นที่ระบบใช้จริง การจัดเตรียม Context การประมวลผลด้วย AI Model จนถึงการส่งผลลัพธ์กลับไปยังผู้ใช้งาน

๖.๕.๕.๓) ผู้รับจ้างต้องทดสอบความสามารถในการรองรับคำขอพร้อมกัน (Concurrent Requests) ของระบบ RAG และต้องรายงานผลกระทบต่อ Response Time และ Throughput เมื่อจำนวนคำขอเพิ่มขึ้น

๖.๕.๕.๔) กรณีที่ระบบมีการใช้งาน Large Language Model (LLM) ผู้รับจ้างต้องรายงานค่า Token Throughput เช่น Tokens per Second (tokens/sec) หรือค่าที่เหมาะสมกับ Model ที่เลือกใช้ พร้อมระบุ Model, Model Version, Hardware และ Configuration ที่ใช้ในการทดสอบ

๖.๕.๖) การทดสอบสมรรถนะ Dashboard และ AI Analytics

๖.๕.๖.๑) ผู้รับจ้างต้องทดสอบ Response Time ของ Dashboard และระบบแสดงผลข้อมูล โดยครอบคลุมการเรียกดูข้อมูลทั่วไป การค้นหา การกรองข้อมูล (Filter) การแสดงผลข้อมูลจำนวนมาก และการแสดงผลข้อมูลที่เกิดจากกระบวนการ AI/ML

๖.๕.๖.๒) ผลการทดสอบต้องแสดงอย่างน้อย Average Response Time, P95 Response Time และ Maximum Response Time รวมถึงจำนวน Concurrent Users ที่ใช้ในการทดสอบ

๖.๕.๖.๓) กรณี Dashboard มีการเรียกข้อมูลจาก AI/ML Service, Data Warehouse, Data Lakehouse หรือระบบวิเคราะห์ข้อมูล ผู้รับจ้างต้องแสดงระยะเวลาการประมวลผลของแต่ละองค์ประกอบ เพื่อให้สามารถวิเคราะห์คอขวดของระบบ (Performance Bottleneck) ได้

๖.๕.๗) การทดสอบความสามารถในการขยายระบบและการจัดสรรทรัพยากร (Scalability Testing)

๖.๕.๗.๑) ผู้รับจ้างต้องทดสอบความสามารถในการเพิ่มทรัพยากรของระบบทั้งแบบ Scale Up และ Scale Out ตามสถาปัตยกรรมที่ออกแบบ

๖.๕.๗.๒) กรณีมีการออกแบบ Auto Scaling ผู้รับจ้างต้องทดสอบการเพิ่มและลดจำนวน Instance/Node ตามภาระงาน และต้องแสดงผลก่อน ระหว่าง และหลังการ Scale

๖.๕.๗.๓) การทดสอบต้องครอบคลุม AI/ML Services, OCR Services, RAG Services และ Application Services ตามองค์ประกอบที่มีอยู่จริงในระบบ

๖.๕.๘) การทดสอบความทนทานและความต่อเนื่องของระบบ (Resilience, High Availability and Failover Testing)

๖.๕.๘.๑) ผู้รับจ้างต้องทดสอบกรณี Node Failure โดยจำลองการหยุดทำงานของ Node ที่ให้บริการ และต้องแสดงให้เห็นว่าระบบสามารถให้บริการต่อเนื่องหรือสามารถกู้คืนบริการได้ตามสถาปัตยกรรมที่ออกแบบ

๖.๕.๘.๒) ผู้รับจ้างต้องทดสอบกรณี Controller Failure หรือองค์ประกอบควบคุมระบบที่มีความสำคัญเกิดความเสียหาย และต้องแสดงความสามารถในการทำงานต่อเนื่องหรือการกู้คืนระบบ

๖.๕.๘.๓) ผู้รับจ้างต้องทดสอบกระบวนการ Failover ขององค์ประกอบที่สำคัญ เช่น Application, Database, AI/ML Service, Storage, Load Balancer หรือองค์ประกอบอื่นตามสถาปัตยกรรมของระบบ

๖.๕.๘.๔) ผลการทดสอบต้องระบุระยะเวลาในการตรวจพบความผิดพลาด (Failure Detection Time) ระยะเวลาในการเปลี่ยนไปยังระบบสำรอง (Failover Time) และระยะเวลาในการกลับมาให้บริการ (Recovery Time) ตามความเหมาะสม

๖.๕.๘.๕) การทดสอบต้องไม่ทำให้ข้อมูลสูญหายเกินกว่าค่า RPO ที่กำหนด หรือทำให้ข้อมูลเสียหาย และต้องมีการบันทึก Log และผลการทดสอบเพื่อใช้เป็นหลักฐานในการตรวจรับ

๖.๕.๙) การจัดทำรายงาน Performance Test สำหรับ AI และ Infrastructure

ผู้รับจ้างต้องจัดทำรายงานผลการทดสอบที่สามารถตรวจสอบย้อนหลังได้ โดยอย่างน้อยต้องประกอบด้วย

๖.๕.๙.๑) Test Scenario และ Test Case

๖.๕.๙.๒) Test Environment และ Hardware Specification

๖.๕.๙.๓) Model และ Model Version ที่ใช้ทดสอบ

๖.๕.๙.๔) Dataset และ Data Volume ที่ใช้ทดสอบ

๖.๕.๙.๕) Concurrent Users / Concurrent Requests

๖.๕.๙.๖) CPU, GPU, Memory และ VRAM Utilization

๖.๕.๙.๗) IOPS, Throughput และ Latency

๖.๕.๙.๘) OCR Throughput และ OCR Response Time

๖.๕.๙.๙) RAG Retrieval Latency และ End-to-End Response Time

๖.๕.๙.๑๐) AI/ML Inference Throughput และ Inference Latency

๖.๕.๙.๑๑) Dashboard Response Time

๖.๕.๙.๑๒) Average, P95 และ Maximum Response Time

๖.๕.๙.๑๓) ผลการทดสอบ Scalability และ Auto Scaling

๖.๕.๙.๑๔) ผลการทดสอบ Node Failure, Controller Failure และ Failover

๖.๕.๙.๑๕) Raw Output, Log, CSV, JSON หรือ ข้อมูลผลการทดสอบที่เกี่ยวข้อง เพื่อให้สามารถตรวจสอบและทำ Audit ย้อนหลังได้

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 5.10

๖.๖.๓. การทดสอบการเจาะระบบโดยบุคคลที่สาม (Third-Party Penetration Testing): ก่อนการนำระบบขึ้นใช้งานจริง (Go-Live) ผู้รับจ้างต้องจัดให้มีการทดสอบการเจาะระบบ (Penetration Testing) โดยบุคคลหรือหน่วยงานที่สามที่เป็นอิสระจากทีมพัฒนา และต้องมีบุคลากรผู้ทำการทดสอบที่ได้รับการรับรองมาตรฐานสากลที่เป็นที่ยอมรับ เช่น OSCP, CREST, GPEN หรือเทียบเท่า โดยขอบเขตการทดสอบต้องครอบคลุมการโจมตีและการตรวจพบช่องโหว่อย่างน้อยตามมาตรฐาน OWASP Top 10 และ OWASP API Security Top 10 ทั้งนี้ หากรายงานผลการทดสอบตรวจพบช่องโหว่ด้านความปลอดภัยในระดับวิกฤต (Critical) และระดับสูง (High) ผู้รับจ้างต้องดำเนินการแก้ไขช่องโหว่ทั้งหมดที่พบให้เสร็จสิ้น และต้องจัดให้มีการทดสอบซ้ำ (Re-test) โดยบุคคลหรือหน่วยงานที่สามรายเดิม เพื่อยืนยันว่าช่องโหว่ระดับดังกล่าวได้รับการแก้ไขโดยสมบูรณ์และไม่พบช่องโหว่ค้างก่อนการส่งมอบและตรวจรับพัสดุ โดยผู้รับจ้างต้องเป็นผู้รับผิดชอบค่าใช้จ่ายในการทดสอบ การแก้ไข และการทดสอบซ้ำทั้งหมด โดยไม่มีการคิดค่าใช้จ่ายเพิ่มเติมแก่ผู้ว่าจ้าง

๖.๖.๔. การทดสอบเชิงรุกด้านความมั่นคงปลอดภัยของระบบปัญญาประดิษฐ์ (AI Red Teaming)

๖.๖.๔.๑ ผู้รับจ้างต้องจัดทำแผนและดำเนินการทดสอบ AI Red Teaming สำหรับองค์ประกอบด้านปัญญาประดิษฐ์ที่อยู่ในขอบเขต โดยขอบเขตและประเภทความเสี่ยงที่ต้องทดสอบให้เป็นไปตามภาคผนวก ง ข้อ ๘.๔

๖.๖.๔.๒ การทดสอบต้องดำเนินการในสภาพแวดล้อมที่เหมาะสมและต้องไม่ส่งผลกระทบต่อข้อมูลหรือการให้บริการจริง เว้นแต่ได้รับความเห็นชอบจากผู้ว่าจ้าง โดยต้องบันทึก Test Scenario, Input/Prompt, องค์ประกอบและเวอร์ชันที่ใช้ ผลลัพธ์ ประเด็นหรือพฤติกรรมที่ตรวจพบ ระดับความรุนแรง และหลักฐานประกอบให้สามารถตรวจสอบย้อนหลังได้

๖.๖.๔.๓ กรณีพบช่องโหว่หรือพฤติกรรมที่ไม่ผ่านเกณฑ์ ผู้รับจ้างต้องดำเนินการแก้ไขหรือจัดให้มีมาตรการควบคุม และดำเนินการทดสอบซ้ำ (Retest) ในส่วนที่ได้รับผลกระทบ โดยก่อนนำระบบขึ้นใช้งานจริงต้องไม่มีประเด็นระดับ Critical หรือ High ที่ยังไม่ได้รับการแก้ไขหรือมีมาตรการจัดการที่ผู้ว่าจ้างให้ความเห็นชอบ

๖.๖.๔.๔ เมื่อมีการเปลี่ยนแปลงที่มีนัยสำคัญต่อองค์ประกอบตามภาคผนวก ง ข้อ ๘.๔.๓ ผู้รับจ้างต้องพิจารณาดำเนินการทดสอบซ้ำในส่วนที่ได้รับผลกระทบตามความเหมาะสม

๖.๗ Backup & Recovery Testing

๖.๗.๑. ผู้รับจ้างต้องทำการทดสอบการ Backup และ Restore เพื่อยืนยันว่าระบบสามารถกู้คืนข้อมูลได้ โดยการสูญหายของข้อมูลต้องไม่เกินกว่าค่า RPO ที่กำหนด

๖.๗.๒. การทดสอบการสำรองและการกู้คืนระบบ (Backup & Recovery Testing):

๖.๗.๒.๑. ผู้รับจ้างต้องจัดทำแผนและดำเนินการทดสอบการสำรองข้อมูล (Backup) และการกู้คืนข้อมูลและระบบ (Restore/Recovery) เพื่อยืนยันว่าข้อมูลและองค์ประกอบของระบบที่อยู่ในขอบเขตการสำรองสามารถกู้คืนและกลับมาใช้งานได้ตามค่าเป้าหมายที่กำหนด โดยต้องจัดทำรายงานและหลักฐานผลการทดสอบเพื่อประกอบการตรวจรับ

๖.๗.๒.๒. การทดสอบต้องครอบคลุมข้อมูล ฐานข้อมูล ไฟล์ เอกสาร การตั้งค่า (Configuration) และองค์ประกอบอื่นที่จำเป็นต่อการกู้คืนและการให้บริการของระบบตามสถาปัตยกรรมที่ออกแบบ รวมถึงองค์ประกอบด้านปัญญาประดิษฐ์ที่จำเป็นต่อการกู้คืนระบบตามขอบเขตที่นำมาใช้งานจริง ทั้งนี้ องค์ประกอบด้านปัญญาประดิษฐ์ที่กู้คืนต้องสามารถตรวจสอบเวอร์ชันและความสอดคล้องกับ AI System Inventory หรือ Registry ขององค์ประกอบประเภทนั้น และต้องสอดคล้องกับสถานะหรือเวอร์ชันที่กำหนดไว้สำหรับจุดกู้คืน โดยให้เป็นไปตามภาคผนวก ง ข้อ ๕.๒ ถึงข้อ ๕.๕ และข้อ ๕.๘ ตามที่เกี่ยวข้อง

๖.๗.๒.๓. การทดสอบต้องพิสูจน์ว่าระบบสามารถกู้คืนข้อมูลและกลับมาให้บริการได้ภายในค่าเป้าหมายอย่างน้อยดังต่อไปนี้

๑) Recovery Point Objective (RPO): ระยะเวลาสูงสุดของข้อมูลที่อาจสูญหายจากเหตุขัดข้องต้องไม่เกิน 24 ชั่วโมง โดยผู้รับจ้างต้องแสดงให้เห็นว่ากลไกและรอบการสำรองข้อมูลสามารถรองรับค่า RPO ดังกล่าวได้

๒) Recovery Time Objective (RTO): ระยะเวลาตั้งแต่เริ่มกระบวนการกู้คืนจนระบบและข้อมูลที่อยู่ในขอบเขตสามารถกลับมาพร้อมให้บริการได้ตามเงื่อนไขที่กำหนดต้องไม่เกิน 4 ชั่วโมง

๖.๗.๒.๔. ผู้รับจ้างต้องทดสอบการกู้คืนจากข้อมูลสำรองจริงอย่างน้อยตาม Test Scenario ที่สำนักงานการตรวจเงินแผ่นดินกำหนดหรือให้ความเห็นชอบ โดยต้องตรวจสอบความครบถ้วนและความถูกต้องของข้อมูลหลังการกู้คืน รวมถึงตรวจสอบว่าระบบหรือบริการที่เกี่ยวข้องสามารถกลับมาทำงานได้ตามที่กำหนด

๖.๗.๒.๕. รายงานผลการทดสอบต้องระบุอย่างน้อย Test Scenario วันที่และเวลาของข้อมูลสำรองที่ใช้ในการกู้คืน เวลาที่เริ่มและสิ้นสุดการกู้คืน ผลการตรวจสอบข้อมูล ค่า RPO และ RTO ที่วัดได้จริง ปัญหาที่พบ และผลการแก้ไข พร้อม Log หรือหลักฐานอื่นที่สามารถตรวจสอบย้อนหลังได้

๖.๗.๒.๖. ในกรณีที่ผลการทดสอบไม่เป็นไปตามค่า RPO หรือ RTO ที่กำหนด ผู้รับจ้างต้องวิเคราะห์สาเหตุ ดำเนินการปรับปรุง และทดสอบซ้ำจนเป็นไปตามเกณฑ์ก่อนการส่งมอบและตรวจรับ

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

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

๖.๙.๑. Configuration Testing ต้องทดสอบการตั้งค่าหรือกำหนดค่าต่าง ๆ ที่ปรับแต่งให้ เหมาะกับการใช้งานในหน่วยงาน เช่น สิทธิ์ผู้ใช้, ภาษา, ฟังก์ชันเฉพาะ ฯลฯ เพื่อยืนยันว่าการกำหนดค่า นั้นถูกต้องและไม่ทำให้ระบบทำงานผิดพลาด

๖.๙.๒. Integration Testing ต้องทดสอบการเชื่อมโยงกับระบบภายนอก เช่น ระบบ บัญชี, ระบบบุคลากร, Email Gateway, LDAP หรือ SSO โดยต้องยืนยันว่าข้อมูลส่งต่อกันได้ถูกต้องและ ครบถ้วน

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

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

๖.๙.๕. Security & Access Control Testing ทดสอบสิทธิ์ การเข้าถึงของผู้ใช้งาน ประเภทต่าง ๆ ตามที่กำหนด (Role-based Access Control) เพื่อยืนยันว่าไม่มีการเข้าถึงข้อมูลโดยไม่ได้รับอนุญาตหากระบบมีช่องโหว่สื่อสารข้อมูลหรือรั่วข้อมูล ให้ทดสอบความเสี่ยงพื้นฐาน เช่น การป้อน ข้อมูลผิดรูปแบบ, การเปลี่ยน URL

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

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

๖.๑๐ ผู้รับจ้างต้องดำเนินการในส่วนที่เกี่ยวข้องกับการทดสอบ (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. SIT (System Integration Test) สำหรับการทดสอบ

๗.๑.๓. UAT (User Acceptance Test Environment) สำหรับผู้ใช้งานปลายทาง/เจ้าของระบบ ทำการตรวจสอบความถูกต้องของระบบ

๗.๑.๔. 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, SIT, 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 โดยต้องไม่มีผลกระทบต่อการใช้งานจริง และไม่มีช่วงเวลาหยุดทำงาน (Zero Downtime Deployment)

๗.๘ ผู้รับจ้างต้องดำเนินการในส่วนที่เกี่ยวข้องกับการปรับใช้ (Deployment) ตามข้อ ๓ การออกแบบ

๗.๙ สำหรับองค์ประกอบด้านปัญญาประดิษฐ์ การนำ AI Model, Prompt, Configuration, AI Processing Pipeline หรือองค์ประกอบ AI อื่นระหว่างสภาพแวดล้อม Dev, SIT, UAT และ Production ต้องเป็นไปตามภาคผนวก ง ข้อ ๒.๙ และการควบคุมเวอร์ชันและการเปลี่ยนแปลงตามข้อ ๕.๘

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 20000: มาตรฐานระบบบริหารจัดการบริการ IT ที่กำหนดให้มีการควบคุมการเปลี่ยนแปลงอย่างเป็นระบบ

๘.๔ สำหรับการเปลี่ยนแปลงองค์ประกอบด้านปัญญาประดิษฐ์ที่มีผลต่อพฤติกรรมหรือผลลัพธ์ของระบบ ให้เป็นไปตามภาคผนวก ง ข้อ ๕.๘ และในกรณีที่เป็นการปรับปรุง ผิดสอน หรือเปลี่ยนแปลง องค์ประกอบ AI จากผลการติดตามหรือข้อเสนอแนะของผู้ใช้งาน ให้เป็นไปตามข้อ ๙.๓ ของภาคผนวก ง

๙. การซ่อมบำรุง (Maintenance)

๙.๑ ผู้รับจ้างจะต้องดำเนินการซ่อมบำรุงระบบ (Maintenance) เพื่อให้ระบบสามารถทำงานได้อย่างต่อเนื่อง มีเสถียรภาพ และสอดคล้องกับความต้องการของสำนักงานการตรวจเงินแผ่นดิน ภายหลังจากการส่งมอบระบบ โดยขอบเขตงานซ่อมบำรุงต้องครอบคลุมรายละเอียด ดังนี้

- ๙.๑.๑. ผู้รับจ้างต้องมีแผนการตรวจสอบและบำรุงรักษาระบบเป็นระยะ เช่น แผนและรายงานการใช้ทรัพยากร การเคลียร์ log ตรวจสอบฐานข้อมูล ตรวจสอบสถานะของระบบที่สำคัญ การประเมินความเสี่ยงจากภัยคุกคามและอัปเดตความปลอดภัย (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 โครงการจัดจ้างระบบคอมพิวเตอร์ ปัญญาประดิษฐ์เพื่อเพิ่มประสิทธิภาพการตรวจเงินแผ่นดิน หัวข้อ 12 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. การเปลี่ยนแปลงระบบในระหว่างการทำงานต้องเป็นไปตามกระบวนการควบคุมการเปลี่ยนแปลงตามข้อ ๘ และสำหรับการเปลี่ยนแปลงองค์ประกอบด้านปัญญาประดิษฐ์ให้เป็นไปตามข้อ ๘.๔ ตามที่เกี่ยวข้อง

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
- ๕) สถานะ GPU กรณีมีการใช้งาน

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) เมื่อผู้รับจ้างตรวจพบเหตุการณ์ผิดปกติหรือได้รับสัญญาณแจ้งเตือน (Alert) ที่อาจส่งผลกระทบต่อความพร้อมใช้งาน ประสิทธิภาพ ความมั่นคงปลอดภัย ความถูกต้องของข้อมูล หรือการให้บริการของระบบ เช่น ความล้มเหลวของบริการ (Service Failure) การใช้ทรัพยากรเกินเกณฑ์ (Resource Utilization Overload) ประสิทธิภาพของระบบลดลง (Performance Degradation) หรือเหตุการณ์ด้านความมั่นคงปลอดภัย (Security Incident) ผู้รับจ้างต้องดำเนินการตามกระบวนการ Incident Management ที่กำหนด เพื่อให้สามารถตรวจพบ บันทึก วิเคราะห์ แจ้งเตือน ตอบสนอง แก้ไข ฟื้นฟู และติดตามเหตุการณ์ได้อย่างเป็นระบบ

สำหรับเหตุการณ์ที่เกี่ยวข้องกับ องค์ประกอบด้านปัญญาประดิษฐ์หรือ บริการ AI ผู้รับจ้างต้องให้กระบวนการ Monitoring และ Incident Management ของระบบโดยรวมสามารถทำงานร่วมกับกระบวนการติดตามและจัดการเหตุการณ์เฉพาะด้านปัญญาประดิษฐ์ เพื่อสนับสนุนการวิเคราะห์สาเหตุ การจำกัดผลกระทบ การแก้ไข และการฟื้นฟูระบบตามความเหมาะสม

ทั้งนี้ รายละเอียดด้านการติดตามประสิทธิภาพ คุณภาพ สถานะการทำงาน และความผิดปกติขององค์ประกอบด้านปัญญาประดิษฐ์ การกำหนด Baseline, Threshold และระดับความรุนแรงของเหตุการณ์ การจัดการเหตุการณ์ที่เกี่ยวข้องกับระบบปัญญาประดิษฐ์ และการดำเนินการเมื่อบริการ AI ไม่พร้อมใช้งาน ให้เป็นไปตาม ภาคผนวก ง การพัฒนาระบบปัญญาประดิษฐ์และการกำกับดูแลการใช้ปัญญาประดิษฐ์ ข้อ ๙.๑ ข้อ ๙.๒ และข้อ ๙.๖ ตามที่เกี่ยวข้อง และในกรณีที่การแก้ไขเหตุการณ์นำไปสู่การปรับปรุงหรือเปลี่ยนแปลงองค์ประกอบด้านปัญญาประดิษฐ์ ให้ดำเนินการตาม ข้อ ๙.๓ ของภาคผนวกดังกล่าว ผู้รับจ้างต้องดำเนินการอย่างน้อยดังต่อไปนี้

๙.๑.๑๖.๑. ดำเนินการตอบสนองและควบคุมเหตุการณ์เบื้องต้น (Immediate Response and Containment) ตามระดับความรุนแรงและขอบเขตที่สำนักงานการตรวจเงินแผ่นดินกำหนด หรืออนุมัติไว้ล่วงหน้า เพื่อจำกัดผลกระทบและป้องกันความเสียหายขยายวงกว้าง

๙.๑.๑๖.๒. แจ้งเหตุการณ์ต่อสำนักงานการตรวจเงินแผ่นดินตามช่องทาง ระดับความรุนแรง และระยะเวลาที่กำหนดในกระบวนการ Incident Management หรือข้อตกลงระดับการให้บริการที่เกี่ยวข้อง

๙.๑.๑๖.๓. วิเคราะห์สาเหตุและผลกระทบของเหตุการณ์ตามความเหมาะสม และจัดทำรายงานสรุปเหตุการณ์ (Incident Report) โดยอย่างน้อยต้องประกอบด้วย วันและเวลาที่เกิดหรือตรวจพบเหตุการณ์ รายละเอียดและระดับความรุนแรงของเหตุการณ์ ระบบหรือบริการที่ได้รับผลกระทบ การดำเนินการตอบสนองและแก้ไข สาเหตุของเหตุการณ์ในกรณีที่สามารถระบุได้ ผลการฟื้นฟูหรือสถานะล่าสุด และข้อเสนอแนะหรือมาตรการป้องกันการเกิดซ้ำ พร้อมหลักฐานที่เกี่ยวข้องเพื่อให้สามารถตรวจสอบย้อนหลังได้

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