Answer the following questions in 1–2 sentences each:

1. What is the purpose of using express.json() in an Express server? Why do we need it before handling POST requests?

express.json() เป็น middleware ที่ใช้สำหรับ parse หรือแปลง JSON payload ที่มาพร้อมกับ request ให้กลายเป็น JavaScript object เพื่อให้เราสามารถเข้าถึงข้อมูลเหล่านั้นได้ผ่าน req.body ก่อนที่จะนำไปใช้ใน POST request

2. In the context of REST APIs, what is the difference between GET /player/:id and POST /player/score in terms of intent and use cases?

GET /player/:id ใช้สำหรับดึงข้อมูลของผู้เล่นที่มี id เฉพาะเจาะจงตามที่ระบุใน URL ซึ่งเป็น method ที่ไม่เปลี่ยนแปลงข้อมูลบนเซิร์ฟเวอร์ ส่วน POST /player/score ใช้สำหรับสร้างหรือส่งข้อมูลใหม่ไปยังเซิร์ฟเวอร์ เช่นการอัปเดตคะแนนของผู้เล่น เป็น method ที่ส่งข้อมูลไปเปลี่ยนแปลงสถานะหรือสร้างข้อมูลใหม่บนเซิร์ฟเวอร์

3. Why should we separate routing logic (routes/player.js) from controller logic  
(controllers/playerController.js) in an Express application?

การแยก routing logic ออกจาก controller logic ช่วยให้โค้ดมีระเบียบและอ่านง่ายขึ้น โดย routing จะทำหน้าที่กำหนดเส้นทางและเรียกใช้ controller ที่เหมาะสม ส่วน controller จะทำหน้าที่จัดการกับ request และ response โดยเฉพาะ ทำให้แต่ละส่วนมีหน้าที่ชัดเจนและลดการพึ่งพาซึ่งกันและกัน

4. How does Express’s Router help in organizing large game backend systems with multiple endpoints?

Express Router ช่วยจัดระเบียบ Endpoint ของระบบหลังบ้านที่มีขนาดใหญ่ โดยอนุญาตให้เราสร้าง router ที่แยกย่อยสำหรับแต่ละส่วนของระบบ (เช่น router สำหรับ player, router สำหรับ inventory) และนำมารวมกันในไฟล์หลัก ทำให้โค้ดมีความเป็นโมดูลและง่ายต่อการจัดการ

5. In Task 4, we created a View using HTML and JavaScript. What role does this View play in the MVC architecture? How does it communicate with the backend?

View ในสถาปัตยกรรม MVC ทำหน้าที่ในการแสดงผลข้อมูลหรือ user interface ให้กับผู้ใช้ โดย View จะสื่อสารกับ Backend ผ่านการส่ง HTTP requests (เช่น GET, POST, PUT, DELETE) ไปยัง API endpoint ที่กำหนดไว้ เพื่อขอข้อมูลหรือส่งข้อมูลกลับไป

6. Imagine you're developing an online RPG game. The game has modules such as player stats, inventory, quests, and guild management. How would you design your Express server's folder structure and routing strategy to keep the codebase maintainable and scalable? Please give an example.

ในการออกแบบโครงสร้าง Folder สำหรับระบบเกม RPG ควรมีการแยก Module ตามฟังก์ชันการทำงานหลักๆ เช่น routes, controllers, models, และ middlewares ภายในโฟลเดอร์ routes จะมีการสร้างไฟล์แยกตาม Module (เช่น player.js, inventory.js, quests.js, guilds.js) เพื่อให้การจัดการแต่ละส่วนเป็นอิสระต่อกัน ตัวอย่างเช่น:

* routes/player.js จะมี Endpoint ที่เกี่ยวข้องกับ player ทั้งหมด
* controllers/playerController.js จะมี Logic ในการจัดการข้อมูล player
* app.js (ไฟล์หลัก) จะมีการนำ Router ย่อยแต่ละตัวมาใช้งาน เช่น app.use('/player', playerRouter);