CODE REVIEW

TRAINEE: RICHMOND KWAME NYARKO
REVIEWER: PATRICK APPIAH

✅ What Went Well

- **Solid Microservice Design:**
  Clear separation of concerns across services, which aligns well with microservice best practices.

- **Consistent API Responses:**
  Uniform response structure makes the API predictable and easier for clients to consume.

- **Proper Use of HTTP Status Codes:**
  Correct mapping of business outcomes to HTTP status codes enhances clarity and interoperability.

- **Good Use of Interfaces:**
  Leveraging interfaces improves testability and supports clean architecture principles.

- **Effective Use of DTOs and Mappers:**

  DTOs are well-defined and decouple internal models from external API contracts. Use of mappers (manual or libraries like MapStruct) ensures clean transformations and keeps controllers slim.

- **Strong Test Coverage:**
  Excellent effort in writing unit and integration tests. This greatly increases confidence in the system's reliability.

⚠ Areas for Improvement

- **Remove Commented-Out Code:**
  Avoid leaving commented code in the repository. It clutters the codebase and

version control systems already provide history.

- **Improve Folder Structure:**
  Some packages could be better organised to reflect bounded contexts or feature modules, which improves maintainability as the app grows.

- **Global Exception Handling:**
  Extend the global exception handlers to catch more cases and return meaningful error responses, so clients have consistent error messages.

- **Pagination for Large Lists:**
  Endpoints like `getAllRestaurants` and `getAllOrders` should implement pagination. This prevents performance bottlenecks and makes the API scalable.

💡 Suggestions for Future Work

- **Base Test Class:**
  Create an abstract base test class for unit and integration tests. This can centralise shared setups like test containers, reducing duplication and easing maintenance.

- **Shared Libraries / Modules:**
  Consider extracting common code (DTOs, exceptions, utility classes, API response wrappers) into shared libraries. This ensures consistency across microservices and reduces duplication.

✅ Overall, excellent work, showing a solid grasp of microservices and clean coding principles. The above points will take it to an even higher standard.

-