Google-Ready Order Management System
Brief Explanation, Examples & Code

1. Project Overview

This project is a scalable backend application built using Spring Boot and AWS to manage orders securely and efficiently.

2. Example Use Case

A user logs in, creates an order, tracks order status, and retrieves order details securely using JWT authentication.

3. Key Technologies Explained (Brief)

• **Spring Boot:** Simplifies REST API development.
• **JWT:** Secure token-based authentication.
• **Redis:** Improves performance by caching data.
• **AWS:** Ensures scalability and reliability.

4. Sample REST Controller Code

```
@RestController
@RequestMapping("/orders")
public class OrderController {

    @Autowired
    private OrderService orderService;

    @PostMapping
    public Order createOrder(@RequestBody Order order) {
        return orderService.saveOrder(order);
    }

    @GetMapping("/{id}")
    public Order getOrder(@PathVariable Long id) {
        return orderService.getOrderById(id);
    }
}
```

5. JWT Authentication Example

```
public String generateToken(String username) {
    return Jwts.builder()
        .setSubject(username)
        .setIssuedAt(new Date())
        .setExpiration(new Date(System.currentTimeMillis() + 1000 * 60 * 60))
        .signWith(SignatureAlgorithm.HS256, "secretKey")
        .compact();
}
```

6. How Redis Improves Performance

Frequently accessed order data is stored in Redis. Cache hits return data instantly, reducing database calls.

7. Deployment Summary

The application is deployed on AWS EC2 with database on AWS RDS and monitored using CloudWatch.

8. Interview Explanation (Example)

“I built a scalable backend using Spring Boot, implemented JWT authentication, optimized performance with Redis, and deployed the application on AWS achieving 99.9% uptime.”

9. Conclusion

This project demonstrates backend engineering, security, cloud deployment, and scalability skills aligned with Google Software Engineer expectations.