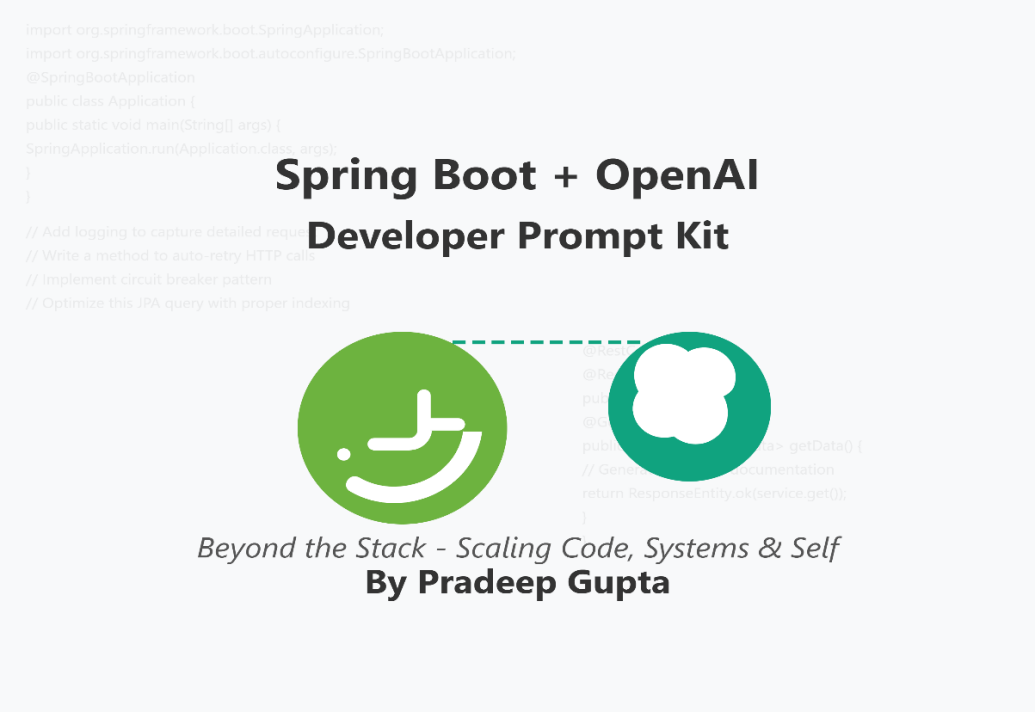
Spring Boot + OpenAI Developer Prompt Kit

Beyond the Stack - Scaling Code, Systems & Self

By Pradeep Gupta



Exclusive Prompts to Debug Faster, Code Smarter, and Observe Better

Developers today need more than just tools - they need smart companions to scale productivity confidently.

🧰 Spring Boot + OpenAI Developer Prompt Kit (Copilot-Optimized Version)

- 🔥 **Debugging Spring Boot Apps (on Cloud)**
- 🧪 **Test Coverage and Test Writing (JUnit5)**
- 🔍 **Code Reviews (Accelerating Development)**
- 📈 **Observability & Investigation (Accelerating Debugging)**
- 🌩️ **Cloud Infrastructure & Deployment**
- 🔒 **Security & Compliance**
- 🚀 **Performance Optimization**
- 📊 **Scalability & Resilience**
- 🧩 **Integration & Interoperability**

✨ Copilot Optimised Prompts:

🔥 Debugging Spring Boot Apps (on Cloud)

// Add logging to capture detailed request and response for cloud debugging

// Write a method to auto-retry HTTP calls with exponential backoff for cloud deployment issues

🧪 Test Coverage and Test Writing (JUnit5)

// Write a JUnit5 unit test for this service method including edge cases

// Generate parameterized tests for input validation in this controller class

🔍 Code Reviews (Accelerating Development)

// Refactor this method for better readability and adherence to clean code principles

// Suggest improvements for error handling and exception flow in this service

📈 Observability & Investigation (Accelerating Debugging)

// Add Micrometer metrics to monitor the performance of this API endpoint

// Create a log summary method to extract key insights from error logs for faster RCA

🌩️ Cloud Infrastructure & Deployment

// Optimize this Dockerfile for smaller image size and faster Spring Boot startup

// Generate Kubernetes deployment YAML with resource limits and health checks for this Spring Boot app

// Create a GitHub Actions workflow with Maven caching and parallel test execution

🔒 Security & Compliance

// Add CSRF protection and secure headers to this Spring Security configuration

// Implement field-level encryption for PII data in this entity class

// Generate input validation for this REST controller to prevent injection attacks

🚀 Performance Optimization

// Optimize this JPA query with proper indexing and fetch strategy

// Implement Redis caching for this frequently-called service method

// Add connection pooling configuration with optimal settings for high throughput

📊 Scalability & Resilience

// Implement circuit breaker pattern with Resilience4j for this external API call

// Convert this stateful service to be stateless for horizontal scaling

// Add distributed tracing with Spring Cloud Sleuth for this request flow

🧩 Integration & Interoperability

// Generate OpenAPI documentation with examples for this REST controller

// Create a service adapter to integrate with this legacy SOAP API

// Implement webhook handler with validation for this event type

✨How to Use:

1. Place these comments above or inside your code.
2. Start typing the comment, and Copilot will predict the code you want!
3. Modify slightly depending on your service class, controller, or repo context.

📦 Spring Boot + OpenAI Developer Prompt Kit

- ✅ Debugging Spring Boot Applications on Cloud
- ✅ Test Coverage and Test Writing (JUnit 5)
- ✅ Code Reviews (Accelerating PR Reviews)
- ✅ Observability & Debugging (Accelerated Investigation)
- ✅ Cloud Infrastructure & Deployment
- ✅ Security & Compliance
- ✅ Performance Optimization
- ✅ Scalability & Resilience
- ✅ Integration & Interoperability
- ✅ Architecture & Design
- ✅ Documentation & Knowledge Transfer

GPT-Based Prompt

☑ Debugging Spring Boot Applications on Cloud

1. Cloud Debugging Prompt: Analyze the following Spring Boot stack trace and suggest root causes based on common cloud deployment issues (e.g., missing environment variables, network/firewall policies, configuration differences): {paste stack trace}

2. Dependency/Config Debugging Prompt: Given this application.properties or YAML configuration, check if there are any missing or misconfigured entries that could cause issues in a cloud deployment. Suggest improvements or fixes: {paste configuration file}

☑ Test Coverage and Test Writing (JUnit 5)

1. Test Coverage Gap Prompt: Analyze this Spring Boot Controller/Service class and identify 5 edge cases or failure scenarios that should be covered by unit or integration tests: {paste Java class}

2. Test Case Generator Prompt: Generate JUnit5 test methods for the following Spring Boot REST Controller, focusing on happy paths, bad requests, and server error simulations. Use Mockito for mocking dependencies: {paste controller code}

☑ Code Reviews (Accelerating PR Reviews)

1. PR Review Summary Prompt: Review this Java pull request diff for a Spring Boot application. Summarize any potential issues around security, performance, readability, and consistency. Suggest 3 actionable improvements: {paste PR diff or code changes}

2. API Contract Validation Prompt: Check if this modified Spring Boot REST endpoint maintains backward compatibility and follows good API design principles (e.g., consistent status codes, request/response validation): {paste updated endpoint signature and body}

☑ Observability & Debugging (Accelerated Investigation)

1. Log Analysis Prompt: Analyze the following application logs collected from a Spring Boot service during an incident. Summarize key error patterns, frequency of failures, and likely root causes: {paste logs}

2. Metrics/Tracing Insight Prompt: Given these application metrics and traces (latency, error rates, throughput) from a Spring Boot microservice, identify bottlenecks, performance regressions, or error hotspots that should be investigated further: {paste metric or trace data}

☑ Cloud Infrastructure & Deployment

1. Infrastructure as Code Review Prompt: Analyze this Terraform/CloudFormation/Kubernetes configuration for my Spring Boot application and identify security best practices, optimization opportunities, and potential scaling issues: {paste IaC code}

2. Containerization Optimization Prompt: Review this Dockerfile for a Spring Boot application and suggest improvements for smaller image size, faster builds, and better security practices: {paste Dockerfile}

3. CI/CD Pipeline Enhancement Prompt: Evaluate this GitHub Actions/Jenkins pipeline for a Spring Boot application and recommend improvements for faster builds, better test coverage, and more reliable deployments: {paste pipeline code}

☑ Security & Compliance

1. Security Vulnerability Assessment Prompt: Review this Spring Security configuration and identify potential security vulnerabilities, missing headers, or authentication/authorization weaknesses: {paste security config}

2. GDPR/Compliance Code Review Prompt: Analyze this controller/service handling user data and suggest improvements to ensure GDPR compliance, proper data anonymization, and secure data handling: {paste data handling code}

3. API Security Checklist Prompt: Generate a security checklist for this Spring Boot REST API to ensure it follows OWASP top 10 best practices and includes proper rate limiting, input validation, and output encoding: {paste API endpoint}

☑ Performance Optimization

1. Database Query Optimization Prompt: Review this Spring Data JPA repository/query method and suggest optimizations for better performance, proper indexing, and reduced database load: {paste repository code}

2. Memory Management Prompt: Analyze this service class for potential memory leaks, inefficient object creation, or thread safety issues. Suggest improvements for better memory utilization: {paste service code}

3. Caching Strategy Prompt: Recommend an appropriate caching strategy (Redis, Caffeine, etc.) for this Spring Boot service method based on data access patterns, update frequency, and performance requirements: {paste service method}

☑ Scalability & Resilience

1. Circuit Breaker Implementation Prompt: Suggest how to implement resilience patterns (circuit breaker, retry, bulkhead) using Resilience4j for this Spring Boot service that makes external API calls: {paste service code}

2. Horizontal Scaling Readiness Prompt: Review this Spring Boot application component and identify potential issues preventing horizontal scaling (state management, session handling, etc.): {paste application component}

3. Event-Driven Architecture Transition Prompt: Transform this synchronous REST-based interaction between services into an event-driven approach using Spring Cloud Stream or Kafka: {paste synchronous code}

☑ Integration & Interoperability

1. API Contract Generation Prompt: Generate an OpenAPI/Swagger specification for this Spring Boot controller to ensure proper API documentation and client code generation: {paste controller code}

2. Legacy System Integration Prompt: Design an adapter pattern implementation to integrate this legacy system API with our modern Spring Boot microservice architecture: {paste legacy system details}

3. Multi-Protocol Support Prompt: Extend this REST controller to also support GraphQL and gRPC clients while maintaining business logic reusability: {paste controller code}

☑ Architecture & Design

1. Domain-Driven Design Refactoring Prompt: Refactor this anemic domain model into a rich domain model following DDD principles with proper aggregates, value objects, and domain events: {paste domain model}

2. Design Pattern Application Prompt: Suggest which design patterns would improve the flexibility and maintainability of this Spring Boot component, and provide sample implementation: {paste component code}

3. Microservice Boundary Analysis Prompt: Evaluate if this service has appropriate boundaries based on domain contexts, and suggest potential decomposition or consolidation: {paste service details}

☑️ Documentation & Knowledge Transfer

1. Technical Documentation Generation Prompt: Generate comprehensive technical documentation for this Spring Boot component, including purpose, dependencies, configuration options, and usage examples: {paste component code}

2. Architecture Decision Record Prompt: Create an Architecture Decision Record (ADR) template for implementing this feature/pattern in our Spring Boot application: {paste feature requirements}

3. Developer Onboarding Guide Prompt: Generate a developer onboarding guide section explaining how this critical component of our Spring Boot application works, its dependencies, and common troubleshooting steps: {paste component code}

How to Use

✨ Copy-paste these prompts directly into your favorite AI assistant and accelerate your Spring Boot development workflow like never before! ✨

Subscribe to Beyond the Stack (Newsletter on LinkedIn for FREE)

Beyond the Stack

Link: https://www.linkedin.com/newsletters/beyond-the-stack-7318612377875161089/

Practical insights and real-world takeaways on Java, Spring Boot, Cloud Engineering, and AI for curious developers

Subscribe to continue receiving exclusive prompts, technical deep dives, and developer productivity tips from **Pradeep Gupta**.