**Static Code Analysis Rubric**

1. **Architecture & Layering**

|  |  |  |
| --- | --- | --- |
| **Assessment Criteria** | **Questions to Ask** | **Comment** |
| **MVC Separation** | Are Controllers, Services, and data access layers clearly separated?  Is business logic kept out of controllers? |  |
| **Controller Design** | Are controllers mapped correctly using @RequestMapping, @GetMapping, etc.?  Are they lightweight, delegating tasks to services? |  |
| **Service Layer** | Are services adhering to the Single Responsibility Principle (SRP)?  Is complex logic encapsulated in services? |  |
| **Data Access Layer** | Is SQL or JDBC usage confined to a dedicated layer?  Are there clear boundaries between data access and business logic? |  |

1. **Code Quality & Style**

|  |  |  |
| --- | --- | --- |
| **Assessment Criteria** | **Questions to Ask** | **Comment** |
| **Naming & Formatting** | Are naming conventions consistent and meaningful?  Is the code properly formatted (indentation, spacing, etc.)? |  |
| **Code Smells & Redundancy** | Are there overly long methods, duplicated code, or dead code?  Is commented-out code present that should be removed? |  |
| **Readability** | Is the code self-explanatory with adequate inline comments where necessary?  Are magic numbers or hardcoded values avoided? |  |

1. **Security & Session Management**

|  |  |  |
| --- | --- | --- |
| **Assessment Criteria** | **Questions to Ask** | **Comment** |
| **Session Handling** | Are HttpSessions managed correctly without storing sensitive information?  Are session timeouts configured to reduce the risk of hijacking? |  |
| **Input Validation** | Are all user inputs validated on the server side? |  |
| **SQL Injection Prevention** | Are raw SQL queries parameterized (e.g., using PreparedStatement) rather than built via string concatenation? |  |

1. **Credential Storage & Management**

|  |  |  |
| --- | --- | --- |
| **Assessment Criteria** | **Questions to Ask** | **Comment** |
| **Hard-coded Credentials** | Are database credentials (usernames, passwords, connection strings) directly embedded in the source code?  Are there any hard-coded secrets in classes or methods? |  |
| **Environment Variables** | Are credentials loaded securely from environment variables or a secure secrets management system?  Is the code properly abstracted to read credentials from secure sources at runtime? |  |

1. **Deployment & Runtime Environment**

|  |  |  |
| --- | --- | --- |
| **Assessment Criteria** | **Questions to Ask** | **Comment** |
| **Secrets Injection** | Are deployment pipelines and environments set up to inject credentials securely (via environment variables, secure config servers, etc.)? |  |

1. **Database Interaction & Performance**

|  |  |  |
| --- | --- | --- |
| **Assessment Criteria** | **Questions to Ask** | **Comment** |
| **Query Optimization** | Are SQL queries efficient and optimized for performance? |  |
| **Connection Management** | Are database connections closed properly to prevent resource leaks? |  |
| **Transaction Handling** | For multi-step operations, are transactions managed properly with commit/rollback logic? |  |

1. **Exception Handling & Logging**

|  |  |  |
| --- | --- | --- |
| **Assessment Criteria** | **Questions to Ask** | **Comment** |
| **Error Handling** | Are exceptions caught specifically rather than using overly broad catches?  Is there a consistent strategy for handling exceptions in Controllers, Services, etc? |  |
| **Logging Practices** | Is logging implemented using a proper framework (e.g., SLF4J/Logback) rather than System.out calls?  Are sensitive data and detailed error messages omitted from logs? |  |

1. **Maintainability & Modularity**

|  |  |  |
| --- | --- | --- |
| **Assessment Criteria** | **Questions to Ask** | **Comment** |
| **Modularization** | Is functionality broken into reusable, small modules or classes?  Is dependency injection used appropriately to facilitate testability? |  |
| **Documentation & Comments** | Is the codebase adequately documented (e.g., README files, inline comments)?  Are complex business logic or decisions explained? |  |
| **Testability** | Is the code structured to allow for unit and integration testing?  Are interfaces or abstractions provided for easier testing of individual components? |  |

1. **Thymeleaf & Front-end Quality**

|  |  |  |
| --- | --- | --- |
| **Assessment Criteria** | **Questions to Ask** | **Comment** |
| **Template Organization** | Are Thymeleaf templates organized using fragments or layouts for reuse?  Is view logic kept minimal in templates, with business logic handled by controllers/services? |  |
| **HTML/CSS Best Practices** | Is the HTML well-structured and semantic?  Are CSS styles organized, with no inline styles scattered throughout templates? |  |