CODESCORE by Arjun Kumbakkara outsource.arjun@gmail.com Sign Out

Welcome back, outsource.arjun!

Submit your code or SQL queries for AI-powered review and analysis

Code Review SQL Review

Submit Code for Scoring

Programming Language

Java

File Upload

Choose File

Code Content

```
               * allowedIps.setCpMasterID(cpDeatils.getCpId());
               *
               * ips.add(allowedIps); }
               *
               * cpDeatils.setAllowedIps(new HashSet<>(ips));
               *
               * // Adding TPS deatils to VIL_DAG DB
               *
               * JsonNode text = jsonNode.get("apis").get("tpsDetails"); if
(text.isArray()) {
               * tpsList = new ArrayList<>(); for (JsonNode jsonNode1 : text)
{ tpsDetailss =
               * new TPSDetails();
```

Score Code

Code Score Complete

7/23/2025, 1:27:26 PM PDF Share

Language: java

Here's a comprehensive code review of the provided Java controller:
### 1. Overall Code Score
Score: 5/10
Summary:
The code is functional but suffers from several architectural and quality issues. It's a large monolithic controller handling too many responsibilities, with inconsistent error handling, poor separation of concerns, and numerous code smells. The Redis/JPA interactions are generally well-implemented, but the overall structure needs significant refactoring.
### 2. Issues Found
Critical Issues:
1. **Massive God Class**: The controller handles registration, modification, deletion, and querying of accounts - violating Single Responsibility Principle
2. **Inconsistent Error Handling**: Some methods return ResponseEntity with HTTP status, others return plain strings
3. **Security Vulnerabilities**:
- No input validation/sanitization for Redis operations
- Potential SQL injection in raw SQL queries
- Hardcoded credentials/keys in some response messages
4. **Memory Leaks**: Redis/Jedis connections not always properly closed
5. **Thread Safety Issues**: Shared SimpleDateFormat instance without synchronization
Other Problems:
- Duplicate code blocks (especially in error responses)
- Mixed concerns (business logic mixed with presentation)
- Inconsistent logging levels
- Dead/commented code
- Magic numbers/strings
- Inconsistent response formats
### 3. Code Quality
Readability:
- Poor method/variable naming (e.g., `jc`, `jc1`, `jc2` for Jedis instances)
- Excessive method length (some methods > 200 lines)
- Inconsistent indentation and formatting
- Overuse of comments for disabled code
Maintainability:
- Tight coupling between Redis, JPA and business logic
- No clear separation between API contracts and implementation
- Difficult to test due to many dependencies
- High cyclomatic complexity in several methods

Best Practices:

- Violates SOLID principles

- Missing proper DTOs for requests/responses

- No proper service layer abstraction

- Exception handling is inconsistent

### 4. Performance Concerns

1. **Redis Operations**:

- Multiple unnecessary hgetAll operations

- No pipelining for batch Redis operations

- Potential slot calculation overhead in cluster mode

2. **Database**:

- N+1 query problems (fetching related entities separately)

- No pagination in some queries

- No caching of frequently accessed data

3. **Memory**:

- Large object creation in hot paths

- No object pooling for expensive objects

### 5. Recommendations

Architectural:

1. Split into multiple focused controllers (Registration, Query, Management)

2. Introduce proper service layer

3. Use DTOs for all API contracts

4. Implement proper validation layer

Code Quality:

1. Extract helper methods for:

- Response formatting

- Error handling

- Redis operations

2. Remove dead code

3. Standardize logging

4. Use constants for magic strings/numbers

Performance:

1. Implement Redis pipelining

2. Add caching layer

3. Optimize database queries

4. Use connection pooling properly

Security:

1. Add input validation

2. Use parameterized SQL queries

3. Implement proper authentication
4. Sanitize all outputs

### 6. Security Observations

1. **Input Validation**:

- Missing validation for accountId, TPS values
- No rate limiting on endpoints

2. **Data Protection**:

- Potential sensitive data leakage in logs
- No encryption for stored credentials

3. **API Security**:

- No proper authorization checks
- Insecure direct object references
- No CSRF protection

4. **Redis Security**:

- No ACLs on Redis keys
- Potential injection via key names

### 7. Final Score Breakdown

Category	Score (1-10)	Justification
Architecture	3	Monolithic structure, poor separation of concerns
Code Quality	4	Long methods, poor naming, high complexity
Error Handling	5	Inconsistent, some good practices but mixed with anti-patterns
Performance	6	Generally acceptable but with clear optimization opportunities
Security	4	Several vulnerabilities, lacks basic protections
Maintainability	4	Difficult to modify due to tight coupling and size
Documentation	5	Some Swagger docs but missing critical details
Overall	**5**	Works but needs significant refactoring for production quality

Critical Action Items:

1. Break down the monolithic controller
2. Implement consistent error handling
3. Add proper input validation
4. Fix security vulnerabilities
5. Optimize Redis/database interactions

The code shows potential but requires substantial refactoring to meet production-grade standards for enterprise applications.

Review History

8 reviews

Search Reviews

Search by code content, filename, or review...

Filter by Language

All Languages

java Code
7/23/2025 • Java

javascript Code
7/22/2025 • Javascript

sql Code
7/22/2025 • Sql

sql Code
7/22/2025 • Sql

java Code
7/22/2025 • Java

sql Code
7/22/2025 • Sql

sql Code
7/22/2025 • Sql

sql Code
7/22/2025 • Sql

Admin Panel

Cleanup Old Reviews (7+ days)

Approval Requests (4)   Code Reviews (7)

Approval Requests

8 total requests

srilakshmi.mohan@6dtech.co.in Pending

Access request

Requested: 7/23/2025, 12:31:58 PM

shyam.reghu@6dtech.co.in Pending

:)

Requested: 7/23/2025, 10:33:22 AM

saurabh.mishra@6dtech.co.in Pending

For core and SQL query review.

Requested: 7/22/2025, 3:01:27 PM

natbarlal.sharma@6dtech.co.in Approved

Please access

Requested: 7/22/2025, 2:53:03 PM • Processed: 7/22/2025, 2:53:24 PM

testoftheindia@gmail.com Pending

Plase ccept

Requested: 7/22/2025, 1:39:17 PM

arjun.k@6dtech.co.in Approved

Please allow access

Requested: 7/22/2025, 2:48:15 AM • Processed: 7/22/2025, 2:48:32 AM

steffirose.rose@gmail.com Approved

SSAD

Requested: 7/22/2025, 2:32:26 AM • Processed: 7/22/2025, 2:32:59 AM

testofthecountry@gmail.com Approved

dsadasd

Requested: 7/22/2025, 2:23:18 AM • Processed: 7/22/2025, 2:23:40 AM

⚠️ Code Safety & Privacy Notice

Important information about your code security and data handling

What We Protect

- Your code remains **proprietary** and confidential
- No explicit logging or permanent storage beyond user accounts
- Secure transmission using HTTPS encryption
- Row-level security on all database operations

Data Processing

- Code is processed by **AI models** for analysis only
- Analysis results stored in your secure user account
- Only you can access your code reviews and history
- Shared reports are generated on-demand with your consent

Important Disclaimers

Third-Party AI Processing: Your code is processed by third-party AI models for analysis. While we don't explicitly log or save your code beyond your user account, the underlying LLM engine may process your code according to their own privacy policies.

Sensitive Code Warning: Avoid submitting highly sensitive, classified, or production-critical code that contains secrets, API keys, passwords, or proprietary algorithms.

Sharing Responsibility: When you generate shareable reports, you are responsible for ensuring the shared content doesn't violate your organization\'s security policies.

By using CodeScore, you acknowledge that you understand these privacy considerations and agree to use the service responsibly with appropriate code that doesn't compromise security.

CODESCORE

About CodeScore

CodeScore is an AI-powered code review platform that helps developers improve their code quality through comprehensive analysis and scoring. Built with cutting-edge AI technology to provide instant, detailed feedback on your Java, JavaScript, and Python code.

Instant Analysis

Get comprehensive code reviews in seconds with AI-powered insights

Quality Scoring

Receive detailed scores and recommendations for code improvement

Multi-Language

Support for Java, JavaScript, Python, and more programming languages

Join the Community

CodeScore is designed to help developers at all levels improve their coding skills. Get started today and see how AI can enhance your development workflow.

✨ AI-Powered Reviews 📊 Detailed Scoring 📱 Mobile Friendly 🔒 Secure & Private

Arjun Kumbakkara
Creator & Developer
AI & Software Engineering Enthusiast

GitHub Medium