Objective

Enhance your existing Smart Device Management backend with advanced features that demonstrate production-ready skills in system design, security, and performance optimization.

Core Requirements

1. API Performance & Caching (30 points)

Implement Redis caching layer:

- Cache device listings and user data with appropriate TTL (15-30 minutes)
- Cache expensive analytics queries for 5 minutes
- Implement cache invalidation on device updates
- Add response time logging to identify slow endpoints

Performance requirements:

- Device listing API should respond within 100ms for cached results
- Analytics endpoints should handle 1000+ concurrent requests

2. Advanced Authentication & Security (25 points)

Implement refresh token mechanism:

- Short-lived access tokens (15 minutes)
- Long-lived refresh tokens (7 days)
- Token rotation on refresh
- Blacklist mechanism for revoked tokens

Add API security headers:

- CORS configuration
- Rate limiting per endpoint (different limits for auth vs device operations)
- Request logging with IP tracking

3. Real-time Device Status (25 points)

WebSocket integration:

- Real-time device status updates to connected clients
- Broadcast device heartbeat events to organization users

- Connection authentication via JWT
- Handle connection drops gracefully

Alternative if WebSocket is complex:

- Server-Sent Events (SSE) for real-time updates
- Polling optimization with ETag/Last-Modified headers

4. Data Export & Reporting (20 points)

Implement data export APIs:

- Export device logs as CSV/JSON for date ranges
- Generate usage reports with charts data (JSON format)
- Async job processing for large exports (return job ID, check status)
- Email notification when export is ready (simulate with logs)

Bonus Features (Extra 10 points each)

A. Database Optimization

- Add database indexes for frequently queried fields
- Implement database connection pooling
- Add query performance monitoring

B. Error Handling & Monitoring

- Structured error responses with error codes
- Health check endpoint with dependency status
- Basic metrics endpoint (request count, response times)

Technical Requirements

Stack Constraints:

- Must use: Node.js/Express, MongoDB, Redis
- Optional: TypeScript, Docker, any testing framework

Code Quality:

- Modular architecture with clear separation of concerns
- Environment-based configuration

- Input validation on all new endpoints
- Error handling with appropriate HTTP status codes

Deliverables

1. Working Application

- Complete source code on GitHub
- All features functional and tested
- Docker Compose setup for easy local testing

2. Documentation

- Updated README with new features
- API documentation for all new endpoints
- Performance benchmarks for caching improvements
- Architecture diagram (simple sketch/flowchart is fine)

3. Testing Evidence

- Unit tests for core business logic
- Integration tests for new API endpoints
- Performance test results or screenshots
- Postman collection with example requests

Evaluation Criteria

Criteria	**Weight**	**What We're Looking For**
Feature Completeness	40%	All core requirements working correctly
Code Quality	25%	Clean, maintainable, well-structured code
Performance	20%	Proper caching, optimization, response times
Documentation	10%	Clear setup instructions and API docs

Testing	5%	Evidence of testing and quality assurance

Submission Guidelines

What to Submit:

1. GitHub Repository URL (public or provide access)
2. Live Demo URL (optional but preferred - can use Heroku, Railway, etc.)
3. Performance Screenshots showing cache hit/miss, response times
4. Brief Architecture Explanation (2-3 paragraphs in README)

How to Submit:

- Email repository link by Friday 6 PM IST
- Include any special setup instructions
- Mention which bonus features you implemented

Success Tips

Time Management:

- Day 1: Focus on caching and security features (core functionality)
- Day 2: Real-time features and data export + documentation

Prioritization:

- Complete core requirements first
- Add bonus features only after core is solid
- Spend adequate time on documentation - it matters!

Common Pitfalls to Avoid:

- Don't over-engineer - focus on working features
- Test your Docker setup before submission
- Ensure your Redis setup works in containerized environment

Questions?

For technical clarifications, email us within the first 24 hours. We'll respond within 4 hours during business hours.

Remember: This assignment tests real-world backend development skills. Focus on building something you'd be proud to deploy in production.

Good luck! We're excited to see your implementation.