Howazit Senior Developer/Architect Assessment

Overview

This assignment evaluates your architectural thinking, problem-solving approach, and coding skills for Howazit's customer experience platform. You may use AI assistance throughout this assignment. Please document your decisions and reasoning clearly.

Time Allocation

4-6 hours total (AI assistance expected)

- Architecture: 1-2 hours
- Coding: 3-4 hours

Section 1: Architecture Design (50% weight)

Estimated Time: 1-2 hours with AI assistance

Scenario

Howazit is expanding globally and needs to redesign the platform to handle:

- 10x current traffic (from 100K to 1M survey responses/day)
- Multi-tenancy with enterprise clients requiring data isolation
- Real-time analytics and dashboard updates
- AI-powered insights generation that may take 30+ seconds to compute
- Integration with 50+ external CRM/support systems

Task

Design a scalable, resilient architecture for this expansion. Provide:

1. **High-level system architecture diagram** showing major components and data flow
2. **Technology choices justification** (within .NET ecosystem + our current stack)
3. **Data architecture strategy** addressing:
   - Survey response ingestion and storage
   - Multi-tenant data isolation
   - Analytics data modeling
   - Real-time vs batch processing decisions
4. **Scaling strategy** covering:
   - Horizontal scaling approach
   - Database partitioning/sharding strategy
   - Caching layers
   - CDN strategy for global deployment
5. **Integration architecture** for external systems

Deliverables

- Architecture document (3-4 pages)
- System diagram
- Database schema design for key entities
- API contract examples for critical endpoints

Section 2: Coding Challenge (50% weight)

Estimated Time: 3-4 hours with AI assistance

Task: Survey Response Processing Service

Build a .NET service that handles survey response ingestion with these requirements:

Core Requirements

1. **REST API** to receive survey responses with this payload:

```
{
 "surveyId": "string",
 "clientId": "string",
 "responseId": "string",
 "responses": {
  "nps_score": 8,
  "satisfaction": "satisfied",
  "custom_fields": {...}
 },
 "metadata": {
  "timestamp": "2024-01-01T10:00:00Z",
  "user_agent": "string",
  "ip_address": "string"
 }
}
```

2. **Data validation** and sanitization
3. **Async processing** using SQS-like pattern (can simulate with in-memory queue)
4. **Dual storage**: Fast storage (simulate DynamoDB) + Relational storage (SQL Server/SQLite)
5. **Real-time metrics** endpoint returning aggregated NPS scores by client
6. **Error handling** and retry logic
7. **Unit tests** for critical components

Advanced Features (Choose 2)

- Multi-tenant data isolation
- Rate limiting per client
- Data encryption for sensitive fields
- Event sourcing pattern implementation
- Circuit breaker for external dependencies
- Background job for computing insights

Technical Constraints

- Use .NET 6+ with minimal APIs or controllers
- Entity Framework Core for SQL operations
- Implement repository pattern
- Use dependency injection
- Include comprehensive logging
- Docker containerization

Time Breakdown (With AI Assistance)

- **Initial setup & research**: 30-45 minutes
- **Core API implementation**: 1.5-2 hours
- **Dual storage & async processing**: 1-1.5 hours
- **Testing & documentation**: 45-60 minutes
- **Advanced features** (if chosen): 30-45 minutes

Deliverables

- Complete working solution with source code
- README with setup instructions
- API documentation (Swagger/OpenAPI)
- Test coverage report
- Brief explanation of architectural decisions (500 words)

Bonus Points

- Implement health checks
- Add OpenTelemetry/monitoring
- Performance benchmarks
- Load testing setup

Submission Guidelines

Format

- Archive all deliverables in a single ZIP file and upload them to a shared google drive folder that was shared with you
- Include a main README explaining your approach
- Organize folders clearly: /architecture, /code

Evaluation Criteria

Architecture (50%)

- System thinking and scalability considerations
- Technology choices alignment with requirements
- Documentation clarity and completeness
- Innovation and best practices

Coding (50%)

- Code quality and organization
- Test coverage and quality
- Error handling robustness
- Performance considerations
- Documentation and comments

Notes

- AI assistance is explicitly allowed and encouraged
- Document when and how you used AI tools
- Focus on demonstrating your thinking process
- Ask clarifying questions if needed (shows good judgment)
- Quality over quantity - depth is more important than breadth

Questions?

Please reach out to DEV.Test@howazit.com with any clarifications needed.
We're evaluating your ability to identify and ask the right questions as much as your technical skills.

Good luck! We're excited to see your approach to these challenges.