**API Virtualization Service - Technical Specification Document**

**1. System Overview**

This document outlines the complete technical implementation of our API virtualization service, focusing on open-source technologies and Windows Server deployment.

**2. Technology Stack**

**Core Platform**

* Windows Server 2019/2022 with Containers feature
* WSL2 (Windows Subsystem for Linux 2)
* Spring Boot 2.7.x LTS
* WireMock 2.x
* MongoDB 6.0 Community Edition
* Docker Desktop Enterprise
* Minikube/K3s for Kubernetes

**Development Tools**

* Git for version control
* Maven/Gradle for build management
* Jenkins for CI/CD
* Swagger/OpenAPI for documentation

**3. Implementation Details**

**Windows Server Setup**

* Enable Windows Container feature
* Install WSL2
* Configure Docker Desktop Enterprise
* Setup Minikube for Kubernetes orchestration

**Spring Boot Application**

* REST endpoints for virtual service management
* Request/Response mapping engine
* Scenario management
* Template processing
* Security filters
* MongoDB integration
* Swagger documentation

**Database Design (MongoDB)**

Collections:

* virtual\_services: Stores service definitions
* scenarios: Contains scenario configurations
* responses: Holds response templates
* audit\_logs: Tracks system usage
* users: Manages user access

**Docker Configuration**

* Multi-stage build process
* Optimized layer caching
* Resource limiting
* Security scanning integration

**Kubernetes Deployment**

* Namespace isolation
* Resource quotas
* Auto-scaling policies
* Service discovery
* Load balancing
* Config management

**4. API Management**

**Core APIs**

1. Virtual Service Management
   * Create/Update/Delete services
   * Manage response templates
   * Configure scenarios
   * Status monitoring
2. User Management
   * Authentication
   * Authorization
   * Role management
   * Access control
3. System Administration
   * Health monitoring
   * Performance metrics
   * Backup management
   * Log access

**Security Implementation**

* JWT authentication
* API key validation
* Role-based access control
* Audit logging
* SSL/TLS encryption

**5. Operational Features**

**Monitoring**

* Service health checks
* Performance metrics
* Error tracking
* Resource utilization
* Alert management

**Backup Strategy**

* Automated daily backups
* Point-in-time recovery
* Configuration versioning
* Disaster recovery plans

**Scaling Capabilities**

* Horizontal scaling
* Load balancing
* Cache management
* Performance optimization

**6. User Interaction**

**API Access**

* RESTful endpoints
* Swagger UI documentation
* Postman collections
* Example implementations

**Response Customization**

* Template-based responses
* Dynamic data generation
* Conditional logic
* Error simulation

**7. Maintenance Procedures**

**Updates and Patches**

* Rolling updates
* Version control
* Rollback procedures
* Change management

**Performance Tuning**

* Database optimization
* Cache configuration
* Resource allocation
* Network optimization

**8. Development Workflow**

**CI/CD Pipeline**

* Automated testing
* Code quality checks
* Security scanning
* Deployment automation

**Version Control**

* Feature branching
* Code review process
* Release management
* Documentation updates

**9. Support and Troubleshooting**

**Logging**

* Centralized log management
* Error tracking
* Audit trails
* Performance monitoring

**Documentation**

* API documentation
* Setup guides
* Troubleshooting guides
* Best practices

**10. Future Enhancements**

* AI-powered response generation
* Advanced scenario management
* Extended protocol support
* Enhanced security features