**Required Testing Framework Components**

**1. API Testing (MANDATORY)**

**Postman Collection Testing**

* **Complete API test suite** covering all endpoints
* Authentication flow testing
* Project upload/analysis/migration workflows
* Error handling and edge case validation
* Request/response validation for all APIs

**OpenAPI Contract Testing**

* Schema validation for all endpoints
* Response time validation (< 2 seconds for most operations)
* Authentication requirements testing
* Async operation testing
* **OpenAPI 3.0 specification** with 100% endpoint coverage

**2. Unit Testing**

* **Minimum 85% code coverage** (target from documentation)
* **Framework**: JUnit 5 for Java or Jest for Node.js
* **Mocking**: Mockito for Java dependencies
* **Test Categories**:
* Parser testing with various Java code patterns
* Transformation rule validation
* API endpoint testing
* CLI command testing

**3. Integration Testing**

* **Real project testing** with actual open-source Java projects
* **End-to-end workflows**: Complete migration scenarios
* **API integration**: Full API workflow testing
* **Performance testing**: Large project migration benchmarks

**4. Test Projects Required**

According to the docs, test with:

1. Simple Spring Boot 2.x application (baseline)
2. Legacy Java 8 enterprise application (complex scenario)
3. Multi-module Maven project (dependency complexity)
4. Spring Security application (framework-specific challenges)

**5. Load & Performance Testing**

* Performance testing scenarios
* Concurrent user simulation
* Response time benchmarking
* Error rate monitoring
* Tools: K6, JMeter, or similar

**6. Key Testing Areas**

Authentication & Authorization

├── Login/logout flows

├── Token refresh mechanisms

├── Permission validation

└── Rate limiting

Project Management

├── Project upload (valid/invalid files)

├── Project listing with pagination

├── Project status tracking

└── Configuration validation (source/target folders)

Analysis Module

├── Analysis initiation

├── Progress tracking

├── Analysis results validation

└── Error scenarios

Migration Planning

├── Plan generation

├── Risk assessment validation

├── Plan customization

└── Phase breakdown testing

Migration Execution

├── Execution workflows

├── Progress monitoring

├── Backup/rollback mechanisms

└── Error handling

Error Scenarios

├── All HTTP error codes (400, 401, 403, 404, 409, 422, 429, 500, 503)

├── Validation errors

├── Business logic errors

└── Server errors

**7. Testing Infrastructure Requirements**

**Mock Server Setup**

* Development testing environment
* API contract validation
* Frontend development support

**CI/CD Integration**

* Automated test execution on commits
* Code coverage reporting
* Performance regression detection

**Documentation**

* Swagger UI at /api/v1/docs
* OpenAPI spec at /api/v1/openapi.json
* Redoc documentation at /api/v1/redoc

**8. Success Metrics from Documentation**

* **100% API endpoint functionality** with automated testing
* **All APIs documented and testable** via Postman collections
* **95%+ compilation success rate** after migration
* **90%+ test pass rate** after migration
* **API response times under 2 seconds** for most operations
* **Automated API contract testing with 95%+ pass rate**

Would you like me to help you:

1. **Create a sample Postman collection** for the key API endpoints?
2. **Set up a unit testing framework** structure?
3. **Create integration test scenarios**?
4. **Generate an OpenAPI specification** for the APIs?
5. **Design a comprehensive test plan document**?