TechnoReady In-Mexico

Challenge 7 – Java and JavaScript. Programming Procedures

Iván Kaleb Ramírez Torres

Nao ID: 3357

November 10th, 2025

Tracking Tables

Table 1 – Requirements list

Sprint	Requirements
Sprint 1: **JUnit Tests for Reservations (Java)** Focus: Environment setup, creation of comprehensive test suites, ≥ 90 % coverage with JaCoCo.	• Configure Maven or Gradle to include JUnit 5, Mockito and JaCoCo. • Verify IDE setup and ensure proper integration with IntelliJ. • Create test classes for ReservationService, ReservationRepository and ReservationController. • Write positive and negative tests for creating, editing and canceling reservations. • Validate exception handling (e.g., invalid dates, overlapping reservations, past dates). • Achieve at least 90 % coverage using JaCoCo; generate HTML reports. • Document results and issues in test-log.md. • Add screenshots of coverage and successful test runs to /evidence/sprint1/.
Sprint 2: **Jest Tests for Graph Visualization (JavaScript)** Focus: Testing the city-graph module, ensuring reliability and 90 % coverage via Jest.	• Initialize a Node project; install Jest and configure coverage threshold to 90 %. • Create unit tests validating graph rendering of nearby cities and distance links. • Include edge-case tests (empty inputs, duplicate cities, corrupted data). • Mock data sources and simulate user interactions. • Generate Jest coverage reports and ensure all tests pass. • Take screenshots of coverage and test runs for documentation. • Record technical issues and solutions in tech-notes.md. • Optional: include a short peer review file (peer-review.md) with feedback and improvements.
Sprint 3: **Documentation and Diagrams** Focus: Project documentation, code comments, and architectural diagrams for Digital NAO review.	• Write a comprehensive README.md covering project purpose, installation, and testing steps. • Add examples of test execution outputs for Java and JavaScript.

	• Integrate Javadoc and JsDoc comments across modules. • Create system architecture and flow diagrams explaining module interactions. • Review and ensure all documentation is clear, consistent and complete. • Confirm repository permissions for the Digital NAO team. • Store all visual assets in /docs/diagrams/ and screenshots in /evidence/sprint3/.
Final Project: **Document Analysis & Results for the whole project**	• Make a video presentation explaning Analysis & Result of the Challenge 6.

Table 2: Prioritize list

Requirements	Stages (Steps)	Time Estimation	Deliverables
JUnit & Maven Setup	1) Add JUnit 5, Mockito, and JaCoCo 2) Configure coverage threshold (90%) 3) Verify IDE integration and build success	3h	pom.xml, jacoco config, verified environment
Reservation Test Data Setup	1) Create mock objects for Reservation, Room, and Customer 2) Implement TestDataBuilder utilities 3) Define reusable constants	2h	ReservationTestData.java, TestUtils.java
Create Reservation Tests	1) Test valid reservation creation 2) Validate	3h	ReservationServiceTest#create_ok()

	repository persistence 3) Assert generated IDs and timestamps		
Validation & Negative Tests	1) Test invalid inputs (past date, overlapping, null fields) 2) Assert thrown exceptions 3) Check business rule enforcement	4h	ReservationServiceTest#validation_*()
Edit Reservation Tests	1) Simulate reservation modification 2) Validate updated data 3) Ensure restricted edits on canceled reservations	3h	ReservationServiceTest#edit_*()
Cancel Reservation Tests	1) Simulate valid cancellation flow 2) Test late cancellation penalties 3) Verify idempotent operations	3h	ReservationServiceTest#cancel_*()
Exception & Edge Case Handling	1) Mock repository errors 2) Simulate unhandled exceptions 3) Confirm proper wrapping	2h	ReservationExceptionTest.java
Coverage Validation & Fixes	1) Run JaCoCo coverage report 2) Identify missing branches 3) Add tests or refactor code	2h	Coverage ≥ 90%, updated report
Evidence & Documentation	1) Capture screenshots 2) Document issues and resolutions	1h	evidence/, test-log.md

	3) Commit logs and reports		
Jest Environment Setup	1) Initialize Node project and install Jest 2) Set coverage threshold (90%) 3) Create npm scripts for testing	2h	package.json, jest.config.js
Graph Rendering Tests	1) Test node and edge creation 2) Validate distances and relationships 3) Assert correct rendering	3h	graph.spec.js
Graph Error Handling	1) Handle empty/invalid data 2) Verify resilience to missing inputs 3) Confirm error messages	3h	graph-errors.spec.js
Mock Data & Isolation Tests	1) Mock data source and fetch requests 2) Use deterministic mocks 3) Simulate UI rendering without DOM	2h	mocks/graphDataMock.js, Jest tests
Graph Coverage & Optimization	1) Run Jest coverage report 2) Add missing test cases 3) Optimize test speed	2h	Jest coverage ≥ 90%, updated report.
Technical Notes & Peer Review	1) Summarize difficulties and solutions 2) Include feedback from teammate 3) Integrate final adjustments	1h	tech-notes.md, peer-review.md
Documentation Development	1) Write final README.md 2) Include test	2h	README.md

	instructions for Java/JS 3) Add examples of successful outputs		
Code Documentation	1) Add Javadoc/JsDoc 2) Document logic and structure 3) Review for completeness	2h	decision-log.md, inline docs
Diagram Creation	1) Create class and flow diagrams 2) Export to /docs/diagrams/ 3) Integrate into README	2h	docs/diagrams/*.png
Final Review & Access Config	1) Ensure Digital NAO access 2) Validate organization and consistency 3) Perform final project check	1.5h	evidence/README.md, repo ready

As the User Stories was an exercise already made in Challenge 1, All this backlog was made according to Challenge 7 requirements for All 3 Sprints and Final Project.