Functional Programming with Scala Project

Requirements and Specification Document

Said BOUDJELDA

Date : Jun, 25, 2025
Due Date: July 25, 2025
Academic Year: 2024-2025

1 Project Overview

This project aims to develop a comprehensive understanding of Scala 3's advanced functional programming features through the implementation of a data processing and analysis system. Students will create a modular, type-safe application that demonstrates proficiency in modern Scala programming paradigms, including algebraic data types, higher-order functions, and concurrent programming patterns.

1.1 Learning Objectives

Upon completion of this project, students will be able to:

- Utilize Scala 3's new syntax features including significant indentation and improved type inference
- Implement complex data structures using case classes, sealed traits, and enums
- Apply functional programming principles including immutability, pure functions, and referential transparency
- Demonstrate proficiency with higher-order functions, monads, and functional error handling
- Implement concurrent and parallel processing using Scala's Future and parallel collections
- Create comprehensive unit tests using ScalaTest framework
- Document code effectively using ScalaDoc conventions

2 Project Requirements

2.1 Core Functionality Requirements

2.1.1 Data Model (25 points)

Implement a comprehensive data model for a library management system that includes:

- **Book Entity**: Create a case class representing books with fields for ISBN, title, authors (as a list), publication year, genre, and availability status
- **User Entity**: Implement a sealed trait hierarchy for different user types (Student, Faculty, Librarian) with appropriate fields and methods
- **Transaction Entity**: Design a transaction system to track book loans, returns, and reservations with timestamps and user associations
- **Library Catalog**: Implement a main catalog class that manages collections of books and users using immutable data structures

2.1.2 Functional Operations (30 points)

Implement the following functional operations using Scala 3 features:

- **Search and Filter**: Create higher-order functions to search books by various criteria (title, author, genre, availability) using function composition and partial application
- **Data Transformation**: Implement map, filter, and reduce operations on book collections to generate reports and statistics
- **Validation System**: Use Either or custom ADTs to handle validation of book data, user information, and transaction requests
- **Recommendation Engine**: Create a simple recommendation system using functional composition to suggest books based on user reading history

2.1.3 Advanced Scala 3 Features (25 points)

Demonstrate proficiency with Scala 3 specific features:

- **Union Types**: Use union types for flexible API design where appropriate
- **Opaque Types**: Implement opaque type aliases for domain-specific types (ISBN, UserID)
- **Extension Methods**: Create extension methods to enhance existing types with domain-specific functionality
- **Given/Using**: Implement type class patterns using given instances and using clauses
- **Enums**: Replace sealed trait hierarchies with appropriate enum definitions where suitable

2.1.4 Error Handling and IO (10 points)

- Implement robust error handling using functional approaches (Try, Either, Option)
- Create file I/O operations to persist and load library data using JSON format
- Handle edge cases and provide meaningful error messages

2.1.5 Testing and Documentation (10 points)

- Write comprehensive unit tests covering all major functionality using ScalaTest
- Achieve minimum 80% code coverage
- Document all public APIs using ScalaDoc with examples
- Include property-based tests using ScalaCheck for at least two core functions

3 Technical Specifications

3.1 Project Structure

The project must follow standard Scala project structure:

```
project-root/
          build.sbt
          project/
                build.properties
                plugins.sbt
          src/
                main/
                      scala/
                          models/
                          services/
                          utils/
                          Main.scala
                test/
                    scala/
                        models/
                        services/
                        utils/
          data/
          docs/
          README.md
```

3.2 Dependencies and Build Configuration

Your `build.sbt` must include:

```scala
ThisBuild / scalaVersion := "3.3.1"
ThisBuild / version := "0.1.0-SNAPSHOT"
ThisBuild / organization := "edu.efrei"

lazy val root = (project in file("."))
  .settings(
    name := "library-management-system",
    libraryDependencies ++= Seq(
      "org.scalatest" %% "scalatest" % "3.2.17" % Test,
      "org.scalacheck" %% "scalacheck" % "1.17.0" % Test,
      "org.typelevel" %% "cats-core" % "2.10.0",
      "io.circe" %% "circe-core" % "0.14.6",
      "io.circe" %% "circe-generic" % "0.14.6",
      "io.circe" %% "circe-parser" % "0.14.6"
    )
  )
```

3.3 Code Quality Standards

- Follow Scala style guidelines and naming conventions
- Use meaningful variable and function names
- Implement proper error handling without throwing exceptions in business logic
- Ensure all functions are pure where possible

- Use immutable data structures throughout
- Apply appropriate access modifiers (private, protected, etc.)

4 Deliverables

4.1 Source Code

- Complete Scala 3 project with all required functionality
- Properly structured codebase following conventions
- All code must compile without warnings using Scala 3.7.1

4.2 Documentation

- Comprehensive README.md with setup and usage instructions
- ScalaDoc documentation for all public APIs
- Design document explaining architectural decisions (1-2 pages)
- User manual with examples of system usage

4.3 Testing

- Complete test suite with minimum 80% coverage
- Property-based tests for core algorithms
- Integration tests for file I/O operations
- Performance benchmarks for search operations

5 Evaluation Criteria

Criterion	**Points**	**Description**
Functionality	40	Correctness and completeness of implementation
Code Quality	25	Style, organization, and best practices
Scala 3 Features	20	Proper use of advanced language features
Testing	10	Coverage and quality of test suite
Documentation	5	Clarity and completeness of documentation
Total	**100**	

Table 1: Project Evaluation Rubric

6 Submission Guidelines

6.1 Submission Format

- Submit as a compressed archive (.zip or .tar.gz) and far better as Github repository
- Include all source code, tests, and documentation

- Ensure project builds successfully with `sbt compile`
- Include a brief video demonstration (5-10 minutes) showing key features

6.2 Late Submission Policy

- 10% penalty per day late
- No submissions accepted after one week past due date
- Extensions granted only for documented emergencies

7 Resources and References

7.1 Required Reading

- Scala 3 Official Documentation: `https://docs.scala-lang.org/scala3/`
- ”Programming in Scala” by Odersky, Spoon, and Venners (4th Edition)
- Course lecture materials and slides

7.2 Recommended Resources

- Scala 3 Migration Guide: `https://docs.scala-lang.org/scala3/guides/migration/`
- Cats Library Documentation: `https://typelevel.org/cats/`
- ScalaTest User Guide: `https://www.scalatest.org/user_guide`
- Functional Programming in Scala by Chiusano and Bjarnason

8 Academic Integrity

This project must represent your original work. While you may discuss general concepts with classmates, all code implementation must be your own. Use of AI assistants is permitted for documentation and debugging but not for core algorithm implementation. Cite any external sources or inspirations in your documentation.

Plagiarism will result in automatic failure of the assignment and may lead to course failure. When in doubt, consult with the instructor before submission.

9 Getting Help

- Hours: Any time is welcome, I will answer evening generally
- Course discussion forum: Moodle, Teams
- Email: mohamed-said.boudjelda@intervenants.efrei.net
- Study groups encouraged for concept discussion

This document is subject to revision. Students will be notified of any changes via the course management system.