MSSE672 – Component Based Software

Week 2 Submission Guide

Student: John Michael Kreski

Instructor: Mohammad Abu Matar

Assignment: Activity 2 – Incorporate Collections, Lambdas, and Streams

Date: 07/13/2025

## Summary

This week’s assignment focused on incorporating Java Collections, lambda expressions, the Stream API, and functional interfaces into the geometryApp backend.

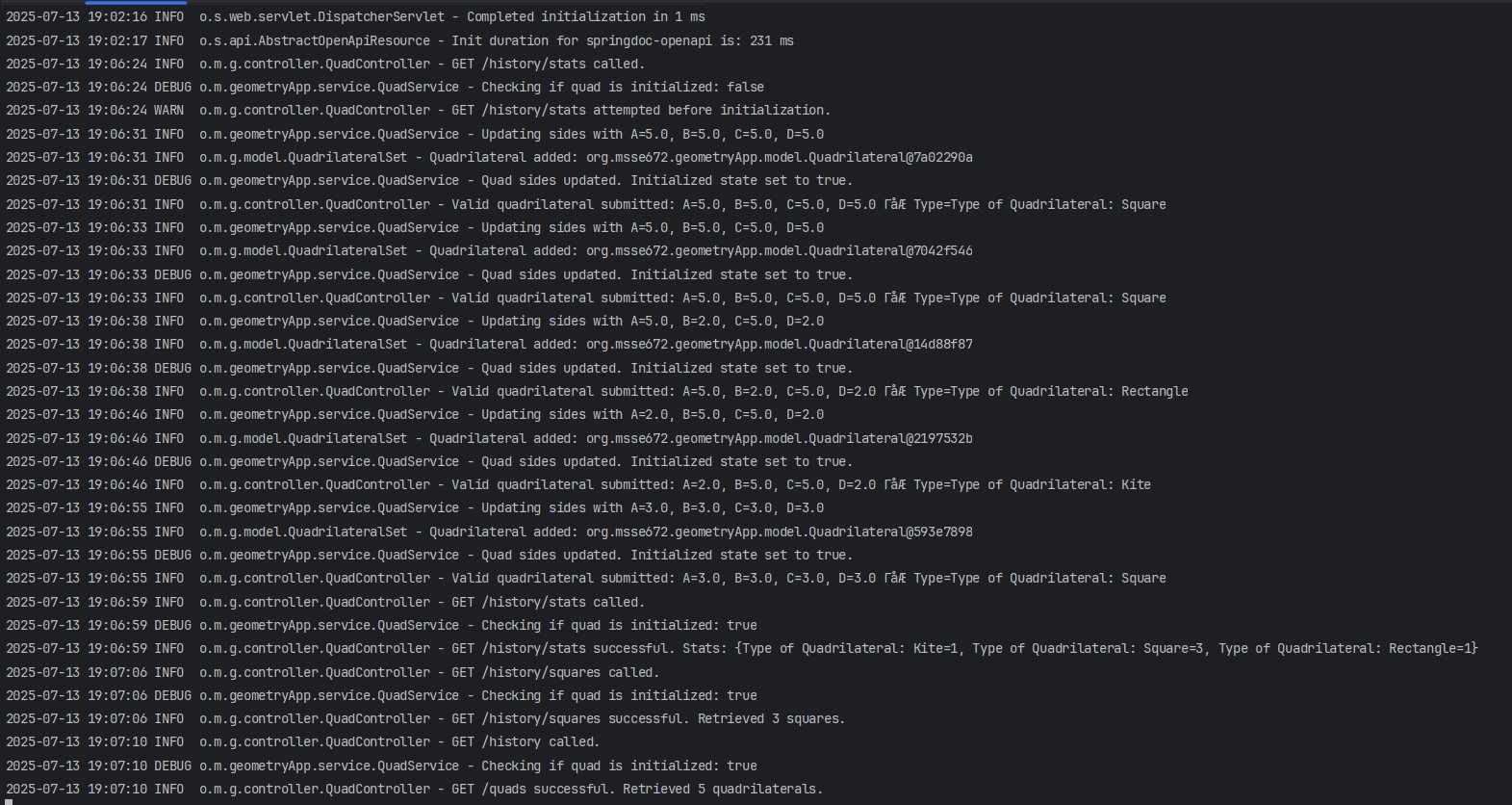
To meet these goals, I expanded the application to track user-submitted quadrilaterals and introduced service methods that process and return filtered lists and statistics:

* **Collections Used:** A List<Quadrilateral> is now maintained in the QuadService to store all submitted quads.
* **Lambdas & Streams:** The getOnlySquares() and getQuadStats() methods use Java Stream operations such as filter(), collect(), and map().
* **Functional Logic:** Basic statistics like total count and shape distribution are derived using stream-based grouping and reduction.
* **Refactoring:** Code changes are limited to QuadService, QuadController, and the domain Quadrilateral class.

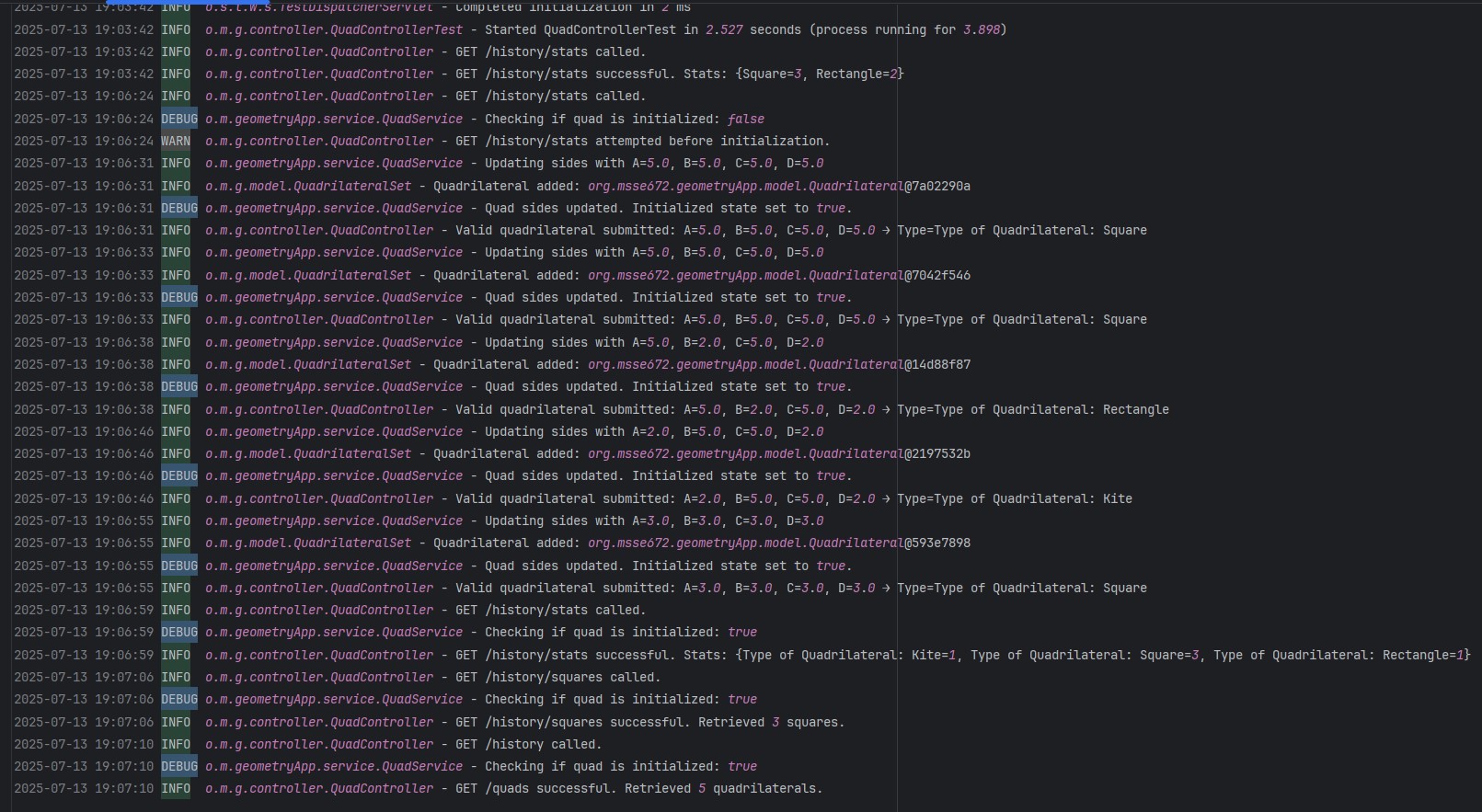
All updates were tested using JUnit and verified through functional testing in the browser and console logs.

## Logs

## Screenshot 1 – Console Logs

*This screenshot displays runtime console output showing log messages at various levels (INFO, DEBUG, etc.), confirming that logging is active and reflecting application flow. It also captures successful POST and PUT requests used to populate the quadrilateral collection.*

## Screenshot 2 – App Logs

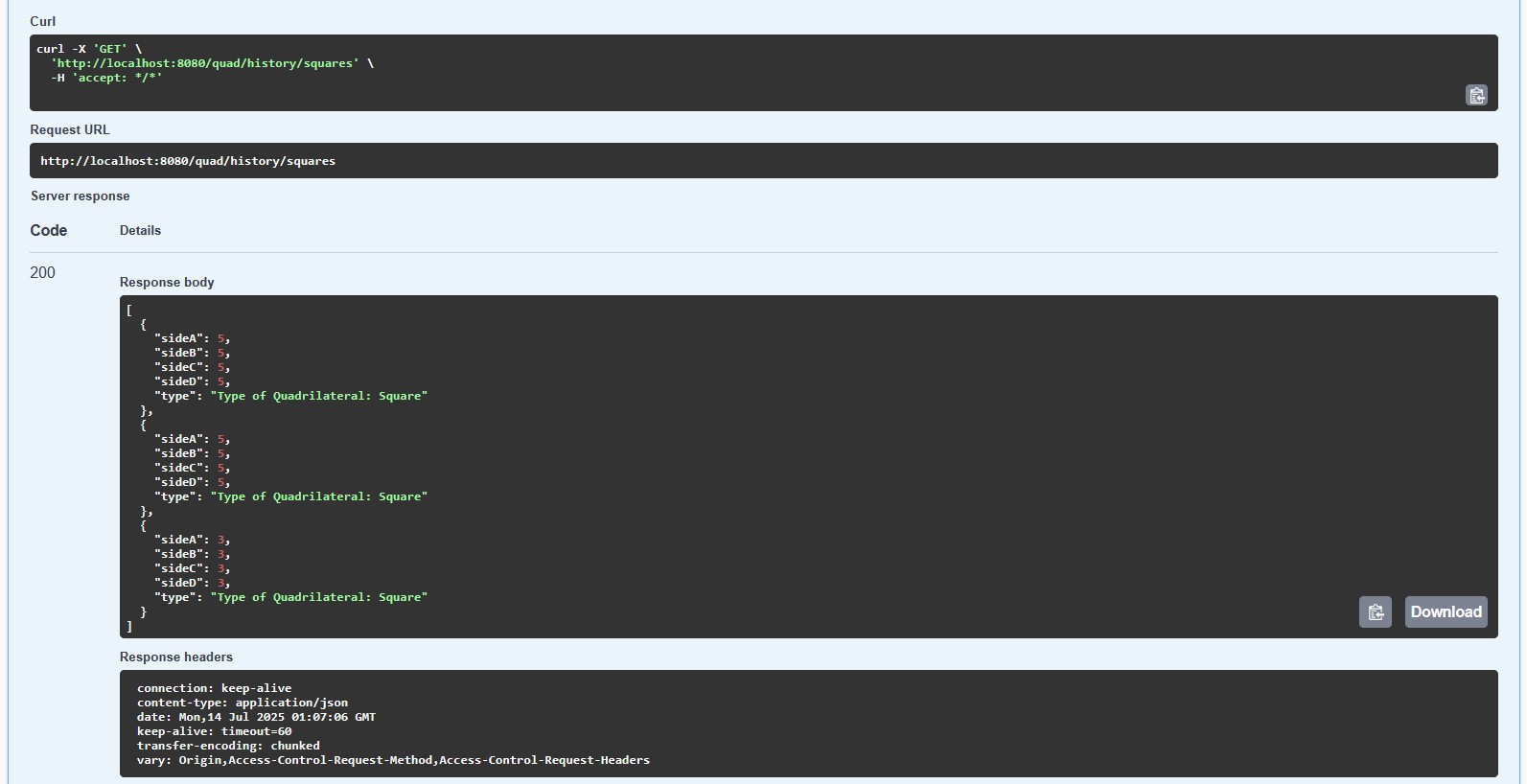
*This screenshot shows the contents of the triangle-app.log file, demonstrating persistent logging of user input and application state. It includes recorded POST and PUT interactions that populate and modify the in-memory collection of quadrilaterals.*

## Spring UI

## Screenshot 1 – /quad/history/stats

*This* *screenshot demonstrates the statistics endpoint for all submitted quadrilaterals, processed using a collection with Stream and Lambda APIs in the service layer.*

## Screenshot 2 – /quad/history/squares

***This screenshot shows the filtered list of quadrilaterals that are classified as squares, returned from a service-layer collection filtered using stream operations.*

## Screenshot 3 – /quad/history/allQuads

*A screenshot of a computer

AI-generated content may be incorrect.This screenshot shows all quadrilaterals stored in memory, confirming that a collection is maintained and accessible via a dedicated service method.*

## Refactored Files

* Added in-memory List<Quadrilateral> to QuadService
* Implemented getAllQuads(), getOnlySquares(), getQuadStats() using lambda + streams
* Modified QuadController to expose new endpoints for filtered and aggregated data
* Added new JUnit tests for both service and controller layers

## Notes

* This project will be maintained in a public GitHub repository for version control and learning practice.
* A `.zip` file containing only `.java` files and the `/Docs` folder is submitted as per course instructions.
  + Docs/week2
* Screenshots are provided in this single document for clarity, as per the professor’s recent guidance.
* All screenshots referenced in this document are included in the project ZIP file under the directory: Docs/Week2/assets/.