ST Assignment 1: Arabic Text Editor

Muhammad Saad (22F-3411) & Muhammad Saood (22F-0504)

February 2026

1 Phase A: Structural Analysis (White-Box Testing)

1.1 Target Method 1: `PaginationDAO.paginate`

The first structural analysis was performed on the `paginate` method within the `dal.PaginationDAO` class to verify logical flow and branch coverage. This method splits a long document into multiple pages based on word or character limits.

1.1.1 Control Flow Graph (CFG)

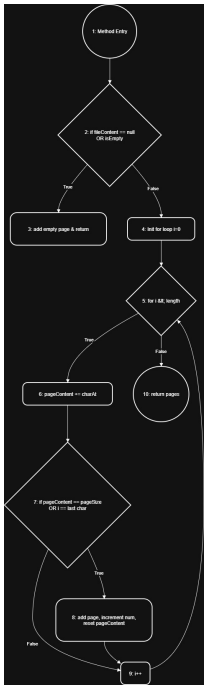

Figure 1: Control Flow Graph for Pagination Logic (`PaginationDAO.paginate`)

1.1.2 Cyclomatic Complexity Calculation

Cyclomatic Complexity is computed as $V(G) = E - N + 2P$:

$$N = 10, \quad E = 12, \quad P = 1$$
$$V(G) = 12 - 10 + 2(1) = 4$$

The complexity of 4 implies that at least 4 linearly independent paths are required for full branch coverage.

1.2 Target Method 2: `SearchWord.searchKeyword`

The second analysis was performed on the search logic within the `bll.SearchWord` class. This method contains nested iterations over documents and pages with multiple conditional branches.

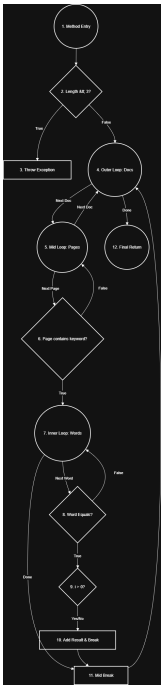

Figure 2: Control Flow Graph for Search Logic (`SearchWord.searchKeyword`)

1.2.1 Control Flow Graph (CFG)

1.2.2 Complexity Calculation

$$N = 12, \quad E = 16, \quad P = 1$$
$$V(G) = 16 - 12 + 2(1) = 6$$

The result implies 6 independent test paths are required to exercise all branches.

2 Phase B: Modular JUnit Testing

2.1 Business Layer: Logic and Commands

The business layer testing focused on high-level algorithms and the Command pattern.

2.1.1 TF-IDF Algorithm and Transliteration

The `TFIDFCalculator` was tested to verify mathematical accuracy against a manual baseline within a tolerance of ± 0.01. Additionally, the Transliteration command was validated to ensure Arabic input maps correctly to its phonetic Latin representation (e.g., "Bsm").

Figure 3: JUnit: TF-IDF Results.

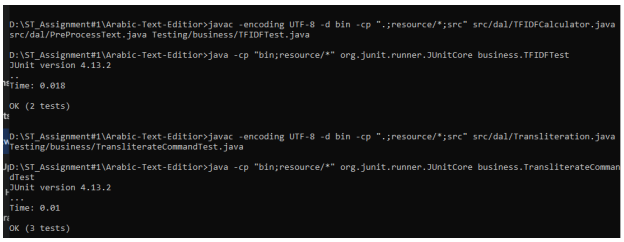

Figure 4: JUnit: Transliteration Results.

2.2 Data Persistence Layer (Mocking & Integrity)

In this layer, we focused on the architectural integrity of the database connection and the security of document metadata.

2.2.1 Hashing Integrity (MD5/SHA1)

We implemented `HashingIntegrityTest` to verify data persistence rules:

- **Verification**: Editing a file's content must generate a new "Session Hash" for the current editor state.
- **Persistence**: The original "Import Hash" stored in the database metadata must remain unchanged, acting as a permanent fingerprint.

2.2.2 Singleton Pattern Testing

The `DatabaseConnection` class was tested to ensure it strictly follows the Singleton pattern:

- **Instance Check**: Verified that `getInstance()` always returns the same object reference.
- **Resource Management**: Confirmed that the JDBC connection is not re-initialized across multiple requests.

2.2.3 Test Execution Results: Data Layer

The following results confirm the successful execution of the Data Layer test suite, ensuring both the Singleton property and the hashing integrity of the system are functional.

```
EBUG StatusConsoleListener Registering MBean org.apache.logging.log4j2:type=2a139a55
EBUG StatusConsoleListener Registering MBean org.apache.logging.log4j2:type=2a139a55,component=StatusLogger
EBUG StatusConsoleListener Registering MBean org.apache.logging.log4j2:type=2a139a55,component=ContextSelector
EBUG StatusConsoleListener Registering MBean org.apache.logging.log4j2:type=2a139a55,component=Loggers,name=
EBUG StatusConsoleListener Registering MBean org.apache.logging.log4j2:type=2a139a55,component=Appenders,name=File
AACE StatusConsoleListener Using default SystemClock for timestamps.
EBUG StatusConsoleListener org.apache.logging.log4j.core.util.SystemClock supports precise timestamps.
AACE StatusConsoleListener Using DummyNanoClock for nanosecond timestamps.
EBUG StatusConsoleListener Reconfiguration complete for context[name=2a139a55] at URI D:\ST_Assignment#1\Arabic-Text-
itor\bin\log4j2.xml (org.apache.logging.log4j.core.LoggerContext@4b29d1d2) with optional ClassLoader: null
EBUG StatusConsoleListener Shutdown hook enabled. Registering a new one.
EBUG StatusConsoleListener LoggerContext[name=2a139a55, org.apache.logging.log4j.core.LoggerContext@4b29d1d2] started

..
me: 0.844

( (5 tests)

EBUG StatusConsoleListener Stopping LoggerContext[name=2a139a55, org.apache.logging.log4j.core.LoggerContext@4b29d1d2
EBUG StatusConsoleListener Stopping LoggerContext[name=2a139a55, org.apache.logging.log4j.core.LoggerContext@4b29d1d2

AACE StatusConsoleListener Unregistering 1 MBeans: [org.apache.logging.log4j2:type=2a139a55]
AACE StatusConsoleListener Unregistering 1 MBeans: [org.apache.logging.log4j2:type=2a139a55,component=StatusLogger]
AACE StatusConsoleListener Unregistering 1 MBeans: [org.apache.logging.log4j2:type=2a139a55,component=ContextSelector
AACE StatusConsoleListener Unregistering 1 MBeans: [org.apache.logging.log4j2:type=2a139a55,component=Loggers,name=]
AACE StatusConsoleListener Unregistering 1 MBeans: [org.apache.logging.log4j2:type=2a139a55,component=Appenders,name=
s]
AACE StatusConsoleListener Unregistering but no MBeans found matching 'org.apache.logging.log4j2:type=2a139a55,compon
*,AsyncAppenders,name=*'
```

Figure 5: JUnit Results: Data Persistence Layer (Singleton Verification and Hashing Integrity).

3 DevOps and Collaboration (GitHub)

To maintain a professional development workflow, we utilized GitHub as our central Version Control System (VCS). All project source code, test suites, and collaboration history are hosted at the following repository:

`https://github.com/SaoodQazi/Arabic-Text-Editior`

This approach integrated several DevOps principles to ensure code quality and collaboration efficiency:

- **Version Control**: All source code and test suites were managed via Git, allowing for history tracking.
- **Collaborative Branching**: We utilized a branch-per-feature strategy where the Business and Data Layer tests were developed on separate branches before being merged.
- **Traceability**: Each commit was documented with descriptive messages, providing an audit trail for the work performed by both collaborators.

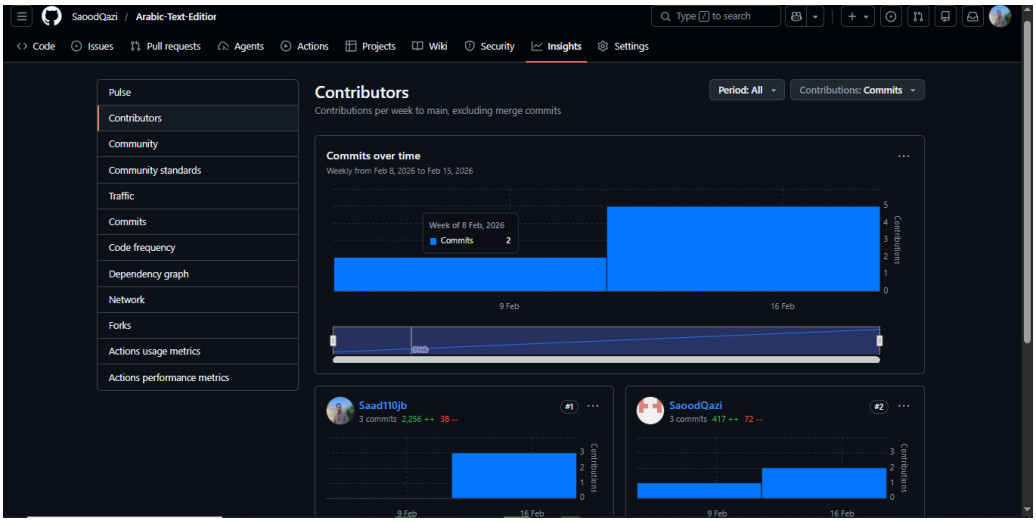

Figure 6: GitHub Network Graph demonstrating active collaboration and branch management.

4 Conclusion

In this project, we successfully performed white-box structural analysis on core methods, calculating Cyclomatic Complexities to ensure comprehensive path coverage. We established a robust modular JUnit infrastructure that separates Business logic from Data persistence. By verifying the Singleton pattern and Hashing integrity in the Data Layer, we ensured the application is resource-efficient and maintains strict data traceability, fulfilling professional Quality Assurance standards.