Software Verification & Validation Assignment 1: Quality Assurance Suite

Group Assignment
Arabic Text Editor Project

February 15, 2026

1 1. Workflow & DevOps Strategy

We implemented a strict professional workflow using GitHub Projects and Kanban methodology as required.

- **Repository:** Public GitHub repository established with branch protection.
- **Kanban Board:** Utilized columns for *Backlog*, *To Do*, *In Progress*, *In Review*, and *Done*.
- **Traceability:** All commits are linked to specific Issue IDs (e.g., `feat: Auto-Save logic #14`).

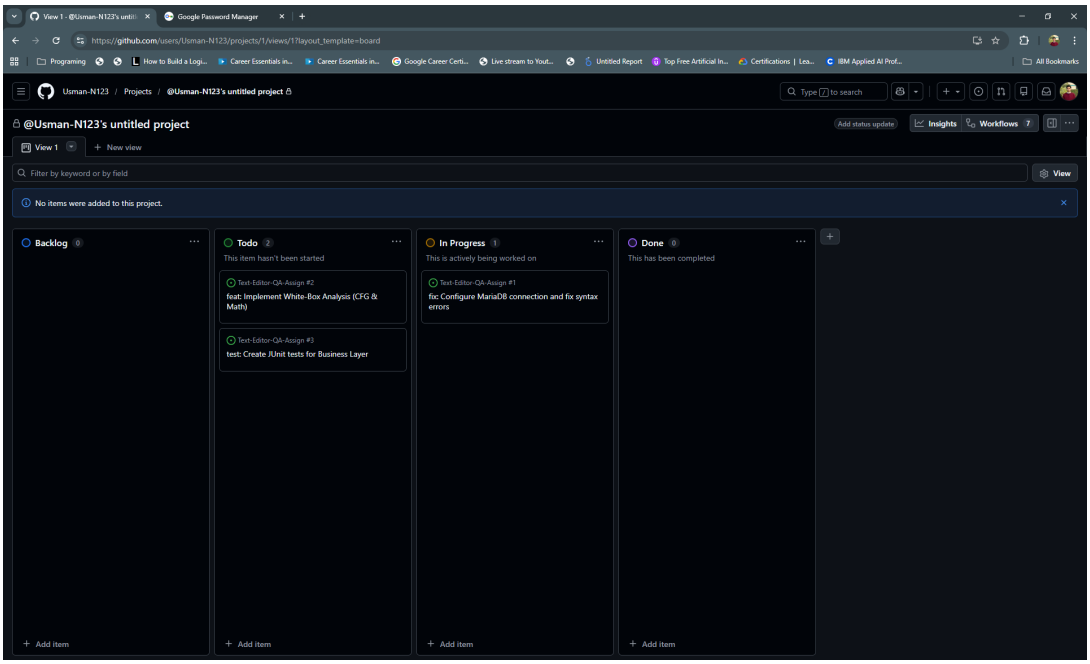

Figure 1: Kanban Board showing issue tracking and workflow status.

2 2. Phase A: White-Box Analysis

Per the requirements, we mathematically analyzed the complexity of two critical business logic features: **Auto-Save Trigger** and **Search & Replace**.

2.1 Feature 1: Auto-Save Trigger Logic

Logic Description: The system checks if the current word count exceeds the threshold (500 words). If true, it triggers the save command and resets the internal timer.

2.1.1 A. Control Flow Graph (CFG)

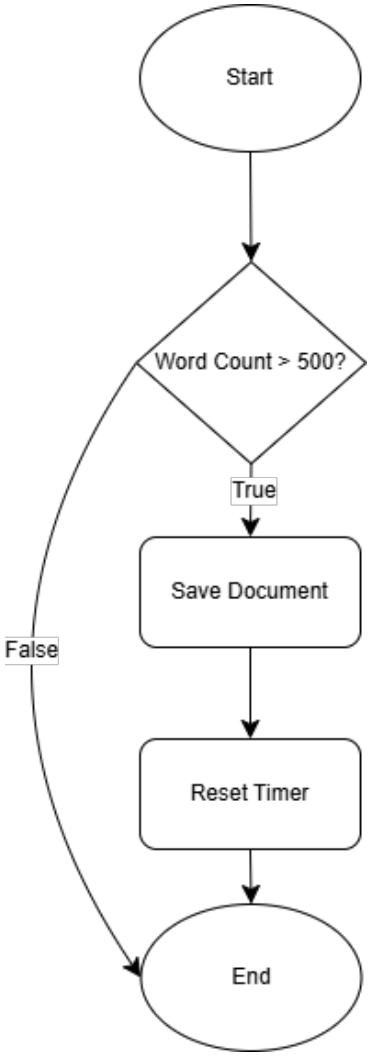

Figure 2: CFG for Auto-Save Trigger

2.1.2 B. Cyclomatic Complexity Calculation

Using McCabe's formula $V(G) = E - N + 2P$:

- **Nodes (N):** 3 (Start Check, Save Action, End)
- **Edges (E):** 3 (Check → Save, Check → End, Save → End)

- **Connected Components (P):** 1

$$V(G) = 3 - 3 + 2(1) = \mathbf{2}$$

2.1.3 C. Independent Path Set

Using Set Notation $P = \{p_1, p_2, ..., p_n\}$:

$$P = \{p_1, p_2\}$$

Where:

- $p_1 = \langle n_{start}, n_{save}, n_{end} \rangle$ (Threshold Met: Save Executed)
- $p_2 = \langle n_{start}, n_{end} \rangle$ (Threshold Not Met: Skipped)

2.2 Feature 2: Search & Replace Word

Logic Description: The algorithm iterates through the document text. It checks for the target string; if found, it replaces the word and increments the change counter before looping back.

2.2.1 A. Control Flow Graph (CFG)

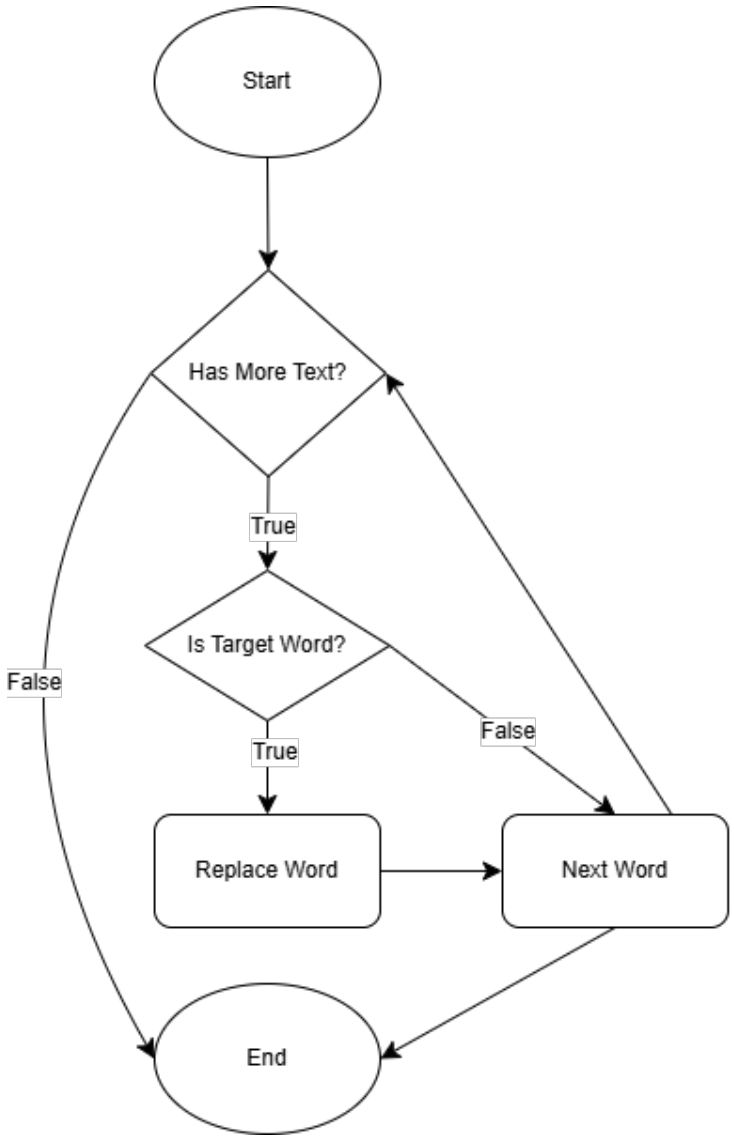

Figure 3: CFG for Search & Replace Loop

2.2.2 B. Cyclomatic Complexity Calculation

Using McCabe's formula $V(G) = E - N + 2P$:

- **Nodes (N):** 4 (Start/While, If Found, Replace Action, End)
- **Edges (E):** 5
    1. Start $\rightarrow$ If Found (Enter Loop)

2. Start → End (Loop Terminate)
3. If Found → Replace (Match True)
4. If Found → Start (Match False - Next)
5. Replace → Start (Action Complete - Next)

- **Connected Components (P):** 1

$$V(G) = 5 - 4 + 2(1) = \mathbf{3}$$

2.2.3 C. Independent Path Set

$$P = \{p_1, p_2, p_3\}$$

Where:

- $p_1 = \langle n_{start}, n_{end} \rangle$ (Empty Document/End of Text)
- $p_2 = \langle n_{start}, n_{check}, n_{start} \rangle$ (No Match Found - Continue)
- $p_3 = \langle n_{start}, n_{check}, n_{replace}, n_{start} \rangle$ (Match Found - Replace & Continue)

3 3. Test Coverage Strategy (Phase B)

We have established a `Testing` package at the project root to ensure modularity.

- **Business Layer:** JUnit tests implemented for `TFIDF.java` (Positive/Negative cases) and Command Pattern execution.
- **Data Layer:** Mocking strategy used to verify MD5 hashing integrity during file edits.