UNT's Best

POLISH

TEST PLAN

Date: 10/18/2025

Walid Esmael, Arnav Verma, Mohamed Babiker, Matthew Norman

Table of Contents

Introduction

The Test Plan has been created to communicate the test approach to team members. It includes the objectives, scope, schedule, risks and approach. This document will clearly identify what the test deliverables will be and what is deemed in and out of scope.

1. Objectives

Polish aims to streamline professional document creation by integrating AI directly into the editor. The goal is to eliminate switching between tools and allow users to edit, prompt, and export documents seamlessly.

Key objectives:

- Integrate real-time AI assistance within the editor (R2, R3).
- Support universal file types: DOCX, PDF, and LaTeX (R1, R4).
- Preserve formatting and layout accuracy (R6, NF3).
- Track and manage AI and manual edits (R5, R10).
- Generate and customize templates via prompts (R7, R8).
- Ensure cross-platform performance and reliability (R9, NF2).
- Maintain fast, secure, and stable operations (NF1, NF5, NF6).

Polish combines AI prompting, editing, and exporting into one cohesive workflow for efficiency and precision.

2. Team Members

Resource Name	**Role *(examples are given below)***
Matthew Norman	Developer
Walid Esmael	Project Manager / Tester
Mohamed Babiker	Tester/Developer
Arnav Verma	Designer/Developer

3.

1.3 Scope

The **Polish** project delivers an AI-powered document editor that integrates writing assistance, editing, and exporting in one platform. The system supports multiple formats (DOCX, PDF, LaTeX) and ensures consistent performance across browsers and operating systems.

In Scope:

- Real-time AI assistance and visual feedback (R2, R3)
- Universal file handling and export (R1, R4)
- Change tracking, version control, and formatting preservation (R5, R6, R10)
- Template generation and customization (R7, R8)
- Secure cross-platform functionality via Azure services (NF2, NF6)

Out of Scope:

- Offline document editing
- Mobile app development
- Integration with third-party AI models outside Azure OpenAI

Assumptions / Risks

1. Assumptions

Assumptions:

- Users have stable internet access for cloud-based AI processing.
- All documents are stored and processed securely through Azure services.
- Supported browsers (Chrome, Firefox, Safari, Edge) meet system requirements.
- AI responses remain accurate and contextually relevant for document editing.

2. Risks

- **Performance Delays:** Slow AI response times may affect workflow (NF1).
- **Data Security:** Breach or misconfiguration in cloud services could expose user data (NF6).
- **Formatting Errors:** Inconsistent document rendering across file types (R6, NF3).
- **Dependency on Azure:** System downtime or API limits could interrupt service availability (NF5).
- **User Adoption:** Resistance from users accustomed to traditional editors.

#	Risk	Impact	Trigger	Mitigation Plan
1	Performance Delays	High	AI response exceeds 3 seconds	Use caching, optimize API requests, and keep an eye on performance using Azure Application Insights.
2	Data Security Breach	High	Unauthorized access or misconfigured Azure Key Vault	Implement RBAC, enforce encryption, and conduct frequent security audits.

3	Formatting Errors	Medium	AI alters document layout or export mismatches formatting	Implement format-preservation validation and run regression tests.
4	Azure Service Downtime	Medium	Azure outage or API rate limit exceeded	Use autosave buffers and failover regions to enable redundancy.
5	Low User Adpotion	Low	Users prefer traditional editiors	Gather user feedback, enhance onboarding, and include in-app tutorials.

3. Test Approach

The Polish project uses 3-week sprints along with an Agile testing methodology. Every sprint will include ongoing testing to make sure all non-functional requirements (NF1–NF6) and user stories (R1–R10) are validated before going on to the next stage. A functional build that passes both automatic and manual validation is produced at the end of each sprint

Sprint 1
Focus: Stability of the UI and core functions
Features include AI Prompting (R2), Live Editing (R3), Universal Upload (R1), and Export (R4).
Type of Testing: Basic regression for upload and AI reaction; manual black-box testing
Objective: Verify that the main user flow (upload → edit → export) runs smoothly.

Sprint 2
Reliability of formatting and advanced AI integration
Qualities: Tracking Changes (R5), Maintaining Formats (R6), and Creating Templates (R7)
Types of Testing: Regression, functional, and UI consistency checks
Objective: Assure proper formatting and seamless template customization.

Sprint 3
System scalability and compliance with accessibility
Qualities: Cross-Platform Access (R9), Version Management (R10), and Dynamic Structure Prompts (R8)
Type of Testing: Security, accessibility, and performance testing
Objective: Obtain 100% accessibility pass, quick AI reaction (<3s), and cross-platform compatibility.

3.1. Test Automation

Regression and accessibility testing automation will start in Sprint 2 with Playwright and aXe. Critical user experiences (upload, AI edit, export) will be the focus of automated test suites to minimize manual labor and guarantee consistent outcomes across builds.

3.2 Test Cases (Black Box)

3.2.1 Landing Page

Test Case ID	Description	Requirements Trace	Directions	Expected Output
LP-01	Hero animation timing & sequencing	R3	Open landing page and observe chatbot typing animation → verify headline change occurs 200–500 ms after typing ends.	Chatbot typing completes before headline change; no overlap or stutter.
LP-02	Feature box wave animation sequence	R3	Scroll to feature boxes section and observe animation order.	Boxes animate left-to-right with ~100 ms stagger; smooth motion.
LP-03	Document type modal behavior	R2	Click CTA to open modal → press ESC → click backdrop → tab through elements.	Modal opens; closes on ESC/backdrop click; focus trap works; no scroll bleed.
LP-04	Navigation and footer links	R9	Click all nav and footer links across pages and verify active states.	All links functional with correct active state; footer text visible.

LP-05	Responsive layout stability	NF2	Resize window (320–1920 px) and interact with menus.	Layout stable across breakpoints; menus collapse properly; CLS < 0.1.

3.2.2 Chatbot Animation

Test Case ID	Description	Requirements Trace	Directions	Expected Output
CH-01	User typing animation before AI response	R2, R3	Send prompt → observe user typing then AI typing dots.	User animation completes before AI typing starts.
CH-02	AI response after typing indicator ends	R2	Send multiple prompts → watch AI dots end then text renders.	Response appears only after typing indicator finishes.
CH-03	Model switch animation & context retention	R8	Switch models mid-session and observe.	Smooth animation; no loss of context/history.
CH-04	Message history order & timestamps	R5	Create thread ≥5 messages → check timestamps.	Chronological order preserved; timestamps sequential.
CH-05	No premature text flash before hydration	NF4	Trigger long AI response with markdown.	No token/markdown flashing before render.

3.2.3 Editor Page

Test Case ID	Description	Requirements Trace	Directions	Expected Output
ED-01	Split-screen layout renders correctly	R3	Open Editor at 1440 px and 768 px breakpoints.	Left pane = preview; right pane = chat; layout stable.
ED-02	AI chat send/receive integration	R2, NF1	Type 'Summarize this paragraph.' and send.	/api/chat returns 200; response renders properly.
ED-03	Export dialog open/close behavior	R4	Click Export → press ESC → check focus.	Dialog opens; ESC closes; focus returns to button.
ED-04	Export functionality (PDF)	R4	Export current doc as PDF → open file.	Valid PDF with correct structure and margins.

3.2.4 API Routes

Test Case ID	Description	Requirements Trace	Directions	Expected Output
API-01	/api/chat request handling	R2, NF1	Send multiple chat requests; measure latency.	HTTP 200; valid JSON; avg latency < 2.5 s.

API-02	/api/export PDF generation	R4	Call export endpoint for sample doc.	Response headers correct; PDF opens successfully.
API-03	Error handling for failed requests	NF5, NF6	Send invalid payload or simulate failure.	Toast error shown; no crash.

Test Case ID	Description	Requirements Trace	Directions	Expected Output
A11Y-01	Color contrast ratios	NF3 / WCAG 2.1	Use Chrome DevTools contrast checker on text.	Contrast ≥4.5:1; focus outlines visible.
A11Y-02	Keyboard navigation	NF4	Use Tab/Shift+Tab through UI and modals.	Logical tab order; no traps; focus contained.
A11Y-03	Screen reader compatibility (NVDA)	NF4	Navigate with NVDA → check landmarks, headings, chat updates.	NVDA reads content correctly and announces live updates.

3.2.5 Accessibility

4. Test Environment

The Polish application will be tested in a regulated online setting that replicates the Microsoft Azure production deployment. To guarantee constant performance and compatibility, the environment facilitates testing across a variety of devices, operating systems, and browsers.

Test Schedule

Task Name ***(sample is below, focus on spring 1 to start)***	**Start**	**Finish**	**Effort**	**Comments**
Test Planning				
Review Requirements documents				
Create initial test estimates				
Learn new test resources				
First deploy to QA test environment				
Functional testing – Sprint 1				
Iteration 2 deploy to QA test environment				
Functional testing – Sprint 2				
System testing				
Regression testing				
Usability Testing				
Resolution of final defects and final build testing				
Deploy to Staging environment				
Performance testing				
Release to Production				