AI Meeting Summarizer Project: Final Report

Prepared by: Eric Morse

Date: May 20, 2025

# Project Overview

The AI Meeting Summarizer is a full-stack web application for automatically summarizing meetings, extracting action items, logging decisions, and integrating with calendar systems. It processes meeting audio files or transcripts, applies NLP and LLM models, and presents results in an intuitive dashboard.  
The stack includes React + Vite + Tailwind (frontend) and Flask + Nextcloud + AssemblyAI + OpenAI (backend and integrations).

# Key Accomplishments

Frontend

* - React + Vite application with full test coverage.  
  - Modern, user-friendly UI using Tailwind CSS.  
  - Components for upload, summary, decisions, review, transcript, error boundaries, and Nextcloud integration.  
  - Fully modular, documented codebase.  
  - Vitest and Playwright for unit, integration, and e2e testing.  
  - All critical user flows (upload, analyze, review, schedule) tested.  
  - No reliance on Zoho; Nextcloud is now the calendar backend.  
  - All test files and key components fully documented (with author/date).

Backend

* - Modular Flask backend with separated routes, services, and utilities.  
  - AssemblyAI and OpenAI Whisper support for transcription.  
  - Robust NLP for action/decision extraction.  
  - Nextcloud API integration for calendar scheduling.  
  - High test coverage (96%) with detailed coverage summaries.  
  - All major service and utility files documented.

Security & Maintenance

* - Upgraded vulnerable dependencies (e.g., flask-cors to 6.0.0, axios to 1.9.0).  
  - No known critical vulnerabilities in direct dependencies.  
  - Browserslist added for frontend compatibility.  
  - Continuous integration with test coverage enforcement.

# Test Coverage Summaries

Frontend (as of latest run)

| Metric | All Files | App.jsx | Components (avg) | Hooks | Utils |  
|-------------------|----------|---------|------------------|-------|-------|  
| Statements | 96.8% | 92.2% | 98%+ | 96% | 96% |  
| Branches | 88.7% | 84.6% | 91–97% | 69% | 85% |  
| Functions | 86.5% | 83.3% | 85–100% | 100% | 100% |  
| Lines | 96.8% | 92.2% | 98%+ | 96% | 96% |  
  
\*Uncovered lines\*: Only a handful of edge case lines in App.jsx, some less common error/branch paths in utils/components.

Backend (latest)

| File | Coverage | Notable Missed Lines |  
|-------------------------------|----------|---------------------------------------|  
| All backend code | 96% | Only a few lines in error handling |  
| app/routes/audio\_routes.py | 89% | 138-149, 171-182 |  
| app/services/audio\_processor.py| 93% | 137, 140, 170, 172-175, 180 |  
| app/services/calendar\_api.py | 90% | 99-104 |  
| app/services/nlp\_analysis.py | 98% | 149, 221 |  
| \*All other files\* | 98–100% | - |

# Dependency & Security Review

Frontend  
- All dependencies up to date per latest recommendations.  
- axios bumped to 1.9.0.  
- Modern React and Vite stack.  
- Playwright for end-to-end browser testing.  
  
Backend  
- flask-cors updated to 6.0.0 to fix CORS normalization bug.  
- All core libraries checked for CVEs; no critical or high vulnerabilities.

# Architecture & Documentation

- Every major React component, hook, and utility function is fully documented with author and date.  
- Backend services, routes, and helpers are documented.  
- Object-oriented diagrams created for both frontend and backend.  
- Monthly status and coverage summary reports available.

# Recommendations & Next Steps

1. Maintain Coverage: Keep enforcing >95% coverage on all new features, especially for branch/edge case logic.  
2. End-to-End Testing: Expand Playwright coverage for real browser flows, especially for critical scheduling and error states.  
3. Upgrade Toolchain: Monitor for future updates to Vite, Vitest, and core libraries; upgrade as needed.  
4. Documentation: Continue to fully document new modules and update architectural diagrams as the codebase grows.  
5. Security: Regularly scan dependencies with npm audit and pip-audit.  
6. Performance & UX: Consider implementing frontend lazy loading and backend async for large audio files.  
7. Release & Deploy: Prepare deployment guides for Docker, Heroku, or cloud.

# Appendix

- All project files are up to date as of May 21, 2025.  
- Please refer to attached documentation and test reports for more granular details.

End of Report