AI Post-Processing Implementation Plan

Project: Taglish Meeting Transcriber - Phase 4
Status: Awaiting User Review & Approval

Executive Summary

This plan adds AI-powered intelligence layers on top of the raw transcription, transforming the app from a basic transcriber into a comprehensive meeting documentation platform. Inspired by industry leaders (Otter.ai, Fireflies.ai, Descript, Rev.ai), the system will offer one-click document generation tailored to meeting types.

Key Value Proposition: "Upload once, generate unlimited meeting artifacts"

---

1. Market Research Findings

Leading Transcription Services (2024-2026)

Otter.ai

- AI-generated meeting summaries with key topics
- Action item extraction with assignees
- Automated follow-up email generation
- Speaker identification and labeling
- Search across all transcripts

Fireflies.ai

- Custom summary templates by meeting type (Sales, Interview, Board Meeting)
- CRM integration (auto-log action items)
- Sentiment analysis on customer calls
- Time-stamped highlights

Descript

- Text-based audio editing (edit transcript = edit audio)
- Filler word removal
- Chapter markers generation

Rev.ai

- Human-in-the-loop review (hybrid AI + human QA)
- Custom vocabulary training
- Multi-language detection

Gap Analysis: What We Should Adopt

Feature	Otter	Fireflies	Descript	**Priority for Us**
Context-Aware Summaries				**High**
Action Item Extraction				**High**
Meeting Type Templates	Basic	Strong		**High**
Speaker Diarization				**Medium**
Search & Replace				**High**
Audio Editing				**Low**
Human Review Queue			Partial	**Medium**

---

2. Proposed Features (Phased Approach)

Phase 4A: Post-Processing Core (Est. 3-5 days)

Feature 1: One-Click Summary Generation **UI Location**: After transcription completes, show dropdown menu:

```
 Generate Document

   Executive Summary
   Action Items & Owners
   Formal Meeting Minutes
   Sermon/Bible Study Notes
   Board Meeting Report
   Key Decisions & Next Steps
```

Technical Architecture: - **LLM**: GPT-4 Turbo or Gemini 1.5 Pro (15K context window) - **Input**: Full corrected transcript + user-selected template - **Output**: Formatted markdown document - **Storage**: Save to Firestore as separate document linked to transcript

Prompt Engineering Strategy:

```
PROMPT_TEMPLATES = {
    "executive_summary": """
        Analyze this meeting transcript and create a 1-page executive summary.
        Format:
        - Meeting Duration & Date
        - Key Topics Discussed (max 5 bullet points)
        - Major Decisions Made
        - Open Questions
    Transcript: {transcript}
""",

"action_items": """
    Extract ALL action items from this meeting transcript.
    For each item, identify:
    1. The task description
    2. The person responsible (if mentioned)
    3. The deadline (if mentioned)
    4. Dependencies or blockers

    Format as a checklist with   for incomplete items.

    Transcript: {transcript}
""",

"meeting_minutes": """
    Create formal meeting minutes following this structure:

    **Meeting Information**
    - Date: [Extract from context]
    - Attendees: [Extract names mentioned]
    - Duration: [Calculate from timestamps]

    **Agenda Items**
    [Group discussion by topic]

    **Decisions Made**
    [List all decisions with rationale]

    **Action Items**
    [Table format: Task | Owner | Deadline]

    Transcript: {transcript}
""",

"sermon_notes": """
    This is a church sermon/Bible study transcript. Extract:

    **Scripture References**
    [List all Bible verses cited with [timestamp]]

    **Main Points**
    1. [Key theological point with timestamp]
    2. [Key theological point with timestamp]
```

```
        **Application Questions**
        [Life application points for small group discussion]

        **Prayer Focus Areas**
        [Topics mentioned for prayer]

        Transcript: {transcript}
    """
}
```

Feature 2: Smart Search & Replace **UI**: Add toolbar above transcript editor:

```
Find: [Pasta John    ]
Replace: [Pastor John  ]
[Replace All] [Replace Next]
```

AI Enhancement: Auto-suggest corrections based on common Taglish/Filipino name misspellings: - "Pasta" → "Pastor" - "Gawain" (task) vs "Gawain" (name)

Feature 3: Export Formats **Current**: Text (.txt)
New Options: - Microsoft Word (.docx) - using `python-docx` - PDF with formatting - using `reportlab` - Google Docs export - using Google Drive API

Phase 4B: Interactive Transcript (Est. 3-4 days)

Feature 4: Review Station UI **Goal**: Allow users to validate and correct the AI transcript with audio sync.

Layout Design:

```
[ ] [ Play] [ Pause] [ ]   Speed: [1.0x  ]


 [0:00:12] Good morning everyone. Let's start...
  Play this segment

 [0:00:45] First item on the agenda is...
  Play this segment    Low confidence
   Suggested: "First item"  [Apply]

 [0:01:30] Pastor John mentioned budget...
  Play this segment
```

Technical Requirements: - **Audio Player**: Integrate Streamlit audio widget with timestamp seek - **Editable Segments**: Each sentence as `st.text_area()` with save button - **Confidence Highlighting**: - Green border: High confidence (>0.9) - Yellow border: Medium confidence (0.5-0.9) - Red border: Low confidence (<0.5) + show alternatives

Implementation:

```
# Store Whisper confidence scores during transcription
segment_data = {
    "timestamp": "[0:00:12]",
    "text": "Good morning everyone",
    "confidence": 0.95,
    "start_ms": 12000,
    "end_ms": 15000,
    "alternatives": []  # From Whisper's n_best if available
}
```

Phase 4C: Speaker Identification (Est. 5-7 days)

Feature 5: Speaker Diarization Tool: Pyannote.audio (already in PROJECT_DEFINITION.md)

Output Format:

```
[0:00:12] **Speaker A**: Good morning everyone. Let's start...

[0:00:45] **Speaker B**: Thanks for joining. First item is...

[0:01:30] **Speaker A**: Pastor John mentioned the budget...
```

User Workflow: 1. After transcription, system shows: "3 speakers detected" 2. User labels speakers: - Speaker A → "Pastor John" - Speaker B → "Elder Maria" - Speaker C → "Youth Leader Carlos" 3. System updates all instances automatically

Challenge: Taglish accent handling
Solution: Train custom speaker embedding model on Filipino voice samples

---

Phase 4D: Advanced AI Features (Est. 4-6 days)

Feature 6: Context-Aware Templates **Logic**: Auto-detect meeting type based on transcript content

Detection Prompts:

```python
def detect_meeting_type(transcript_sample):
    """
    Analyze first 500 words to classify meeting type.
    Returns: 'church_service', 'board_meeting', 'staff_meeting', 'bible_study'
    """

    prompt = f"""
    Classify this meeting based on the first 500 words:
    Options: church_service, board_meeting, staff_meeting, bible_study, other

    Sample: {transcript_sample[:2000]}

    Return only the classification label.
    """

    classification = llm.predict(prompt)
    return classification
```

Auto-Apply Templates: - Church Service → Sermon Notes template - Board Meeting → Minutes + Action Items - Bible Study → Discussion Questions + Scripture Index

Feature 7: Multi-Transcript Search **UI**: Global search bar in sidebar

```
Search All Meetings

Query: [budget discussion]

Results:
Jan 15 Board Meeting [0:45:12]
Jan 8 Staff Meeting [1:02:00]
Dec 20 Strategic Planning
```

Implementation: Full-text search using Firestore or Algolia

---

3. Technical Architecture

System Flow

```
Raw
Transcript
```

```
 User Correction (Review Station)




 AI Post-Processor       Template Selection
 (GPT-4 / Gemini)


          Executive Summary
          Action Items
          Meeting Minutes
          Custom Format




 Export Engine
 (.txt/.docx/.pdf)
```

Database Schema Updates

Firestore Collections:

```
// Current
/transcripts/{job_id}
  - filename
  - status
  - transcript (raw text)
  - upload_date

// New
/transcripts/{job_id}
  - filename
  - status
  - transcript (raw text)
  - segments (array)          // NEW: Per-sentence data
  - speakers (array)          // NEW: Speaker labels
  - upload_date
  - meeting_type              // NEW: Auto-detected
  - corrected_transcript      // NEW: User-edited version

/summaries/{job_id}_{type}    // NEW COLLECTION
```

```
- source_transcript_id
- summary_type (e.g., "action_items")
- generated_date
- content (markdown)
- prompt_used
```

API Integrations

Required: - OpenAI API (GPT-4 Turbo) - $0.01/1K tokens - OR Google Vertex AI (Gemini 1.5 Pro) - $0.00025/1K tokens ← **Cheaper**

Optional: - Pyannote.audio (Self-hosted, no API cost) - Google Drive API (Free tier: 15GB storage)

---

4. Cost Estimation

Per-Meeting Costs (3-hour meeting = ~15,000 words)

Service	Current	With AI Post-Processing
Whisper Transcription	$0.36	$0.36 (no change)
Summary Generation (GPT-4)	-	$0.15
Action Items (GPT-4)	-	$0.10
Meeting Minutes (GPT-4)	-	$0.15
Total	**$0.36**	**$0.76** (+$0.40)

Alternative: Use Gemini 1.5 Pro - Summary/Minutes/Actions: ~$0.04 total - **New Total**: $0.40 per meeting (saves 50%)

Monthly Cost (20 meetings/month)

- **With GPT-4**: $15.20/month
- **With Gemini**: $8.00/month

---

5. Implementation Timeline

Week 1: Core Summarization

- ☐ Day 1-2: Implement prompt templates (6 types)
- ☐ Day 3: Build UI dropdown + generation flow
- ☐ Day 4: Add export to .docx and PDF
- ☐ Day 5: Testing with real church meetings

Week 2: Review Station

- ☐ Day 1-2: Build segment-based UI with audio sync
- ☐ Day 3: Implement confidence scoring + highlighting
- ☐ Day 4-5: Add smart search/replace tool

Week 3: Speaker Diarization (Optional)

- ☐ Day 1-2: Setup Pyannote.audio pipeline
- ☐ Day 3: Build speaker labeling UI
- ☐ Day 4-5: Fine-tune for Filipino accents

---

6. User Review Required

Decision Points

1. **LLM Choice**: GPT-4 ($0.40/meeting) vs Gemini ($0.04/meeting)?
   - Recommendation: **Gemini 1.5 Pro** (8x cheaper, comparable quality)
2. **Phase Priority**: Which features to build first?
   - Recommendation: **Phase 4A only** (summaries + exports)
   - Defer speaker diarization (complex, lower ROI)
3. **Export Formats**: Do you need Google Docs integration?
   - Recommendation: Start with .docx and PDF, add Drive later
4. **Meeting Types**: Which templates are most valuable?
   - Current list: Executive Summary, Action Items, Minutes, Sermon Notes, Board Report
   - Add custom types?

---

7. Success Metrics

Before AI Features: - User downloads raw .txt file - Manual effort to create summary: ~30 minutes

After AI Features: - User clicks "Generate Action Items" - Receives formatted document: ~**30 seconds** - Time savings: **98%**

Measurement: - Track which summary types are most generated - Collect user feedback on accuracy - Monitor API costs per meeting

---

Next Steps

1. **User Review**: Approve/modify this plan
2. **LLM Selection**: Confirm GPT-4 vs Gemini choice

3. **Phase Scope**: Prioritize Phase 4A features
4. **Budget Approval**: $8-15/month for AI processing
5. **Start Development**: Begin Week 1 implementation

Estimated Total Development Time: 2-3 weeks (depending on phases selected)