MedCall QA Documentation

# 1. Test Plan

\*\*Project Name:\*\* MedCall (AI Hospital Platform)  
\*\*Prepared By:\*\* QA Team  
\*\*Date:\*\* May 30, 2025  
  
\*\*Objective:\*\* Ensure end-to-end functionality, reliability, and performance of MedCall's AI-driven healthcare platform including chatbot interactions, speech-to-text/speech-to-speech conversion, WebSocket streaming, and LLM response accuracy.  
  
\*\*Scope:\*\* Web frontend, backend APIs, WebSocket integration, Digram-based STT, ElevenLabs integration, chatbot logic, and login flow.  
  
\*\*Testing Types:\*\*  
- Functional Testing  
- API Testing  
- WebSocket Testing  
- Integration Testing  
- Regression Testing  
- Security Testing  
- Performance Testing  
  
\*\*Test Deliverables:\*\*  
- Test Plan  
- Test Cases  
- API Test Strategy  
- Bug Reports  
- Test Summary Report  
  
\*\*Tools Used:\*\*  
- Postman  
- Playwright/Pytest  
- JMeter (for load)  
- Browser DevTools (for WebSocket inspection)

# 2. Test Strategy

\*\*Test Levels:\*\*  
- Unit Testing: Performed by developers.  
- Integration Testing: Validate frontend-backend communication (REST + WebSocket).  
- System Testing: Validate MedCall as a whole.  
- UAT: Performed with clinical scenarios.  
  
\*\*Test Data Management:\*\*  
- Use anonymized patient examples.  
- Generate multiple text/audio inputs for LLM and STT.  
  
\*\*Defect Management:\*\*  
- Logged via Jira.  
- Severity/Priority assignment based on patient impact.  
  
\*\*Exit Criteria:\*\*  
- All critical/blocker issues resolved.  
- ≥ 90% test coverage on critical workflows.

# 3. API Test Strategy

\*\*Scope:\*\* Backend REST APIs and WebSocket endpoints used for:  
- Sending user queries (text/audio)  
- Receiving LLM responses  
- Speech-to-text/audio synthesis  
  
\*\*Test Areas:\*\*  
- Authentication (Guest vs Logged-in user flow)  
- STT request: Validate base64 audio upload and transcription accuracy  
- LLM request: Validate proper routing and response formatting  
- ElevenLabs integration: Audio synthesis timing/streaming  
  
\*\*Validation Types:\*\*  
- Positive & negative tests (missing/invalid tokens, formats)  
- Load Testing (concurrent STT/LLM requests)  
- Latency benchmarks per endpoint  
  
\*\*Tools:\*\*  
- Postman (manual)  
- Pytest + Requests (automation)  
- WebSocket debugging tools

# 4. Sample Test Cases

\*\*TC001: Guest User Redirection\*\*  
- Step: Open chatbot without login and try to send message  
- Expected: Redirects to login page  
  
\*\*TC002: Audio Upload & Transcription\*\*  
- Step: Record and send audio via mic button  
- Expected: Correct transcription displayed  
  
\*\*TC003: LLM Response via WebSocket\*\*  
- Step: Send "How can I book an appointment?"  
- Expected: Streamed answer from LLM with no lag  
  
\*\*TC004: ElevenLabs Audio Generation\*\*  
- Step: Trigger audio response from LLM result  
- Expected: Audio output synthesized correctly with minimal delay

# 5. Sample Bug Report

\*\*ID:\*\* BUG-1023   
\*\*Title:\*\* Audio response delayed > 10 seconds after LLM processing   
\*\*Severity:\*\* High   
\*\*Reported By:\*\* QA Team   
\*\*Environment:\*\* Staging   
\*\*Steps to Reproduce:\*\*  
1. Open chatbot as logged-in user  
2. Speak query via mic  
3. Wait for response  
  
\*\*Expected Result:\*\* Audio response should stream within 2-4 seconds  
\*\*Actual Result:\*\* Audio streamed after ~12 seconds   
\*\*Status:\*\* Open   
\*\*Logs Attached:\*\* Yes (backend latency logs)   
\*\*Assigned To:\*\* Backend Dev Team