Test Cases Document

Synthetic Radio Host - Comprehensive Test Suite

Contents

1. Executive Summary

This document provides documentation of the test suite for the Synthetic Radio Host system. The test suite consists of 51-unit tests covering all 10 core functions, ensuring robust error handling, edge case coverage, and integration testing. All tests use mocking to avoid external API dependencies, making them fast, reliable, and repeatable. Test suite is developed using pytest framework. Pytest is a open-source testing framework used for unit, integration, and functional tests in Python.

2. Test Effectiveness Analysis

2.1 Coverage Metrics

The test suite provides comprehensive coverage across multiple dimensions:

Function Coverage: 100% - All 10 functions have dedicated test classes

Edge Case Coverage: High - Empty inputs, none values, exceptions

Error Handling Coverage: Comprehensive - All error paths tested

Integration Coverage: Complete - End-to-end workflow validated

2.2 Test Quality Indicators

- All external dependencies are mocked (Wikipedia, Ollama, ElevenLabs)
- Tests are isolated and independent - no shared state
- Tests are fast - complete suite runs in seconds
- Tests are repeatable - deterministic results
- Tests validate both happy paths and error scenarios
- Tests include output validation (emotional tags, Hinglish prompts)
- Integration tests verify complete workflows
- Edge cases are thoroughly covered

3. Test Execution

To execute the test suite:

1. Install test dependencies:

```
$ pip install -r requirements-test.txt
```

2. Run all tests:

```
$ pytest test_syntheticRadioHostScript.py -v -s
```

4. Test Coverage Overview

List of test cases ID along with the functionality the test case is testing and validating.

Function	Test Cases	Test IDs
get_elevenlabs_api_key	4	1-4
fetch_wikipedia_context	4	5-8
create_hinglish_prompt	3	9-11
check_ollama_connection	3	12-14
generate_script_with_ollama	7	15-21
parse_script	10	22-31
generate_segment	4	32-35
validate_output_path	3	36-38
generate_podcast	8	39-46
main() Integration	5	47-51

5. Detailed Test Cases

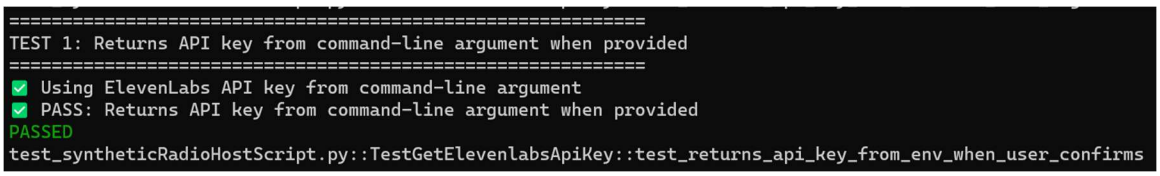
5.1. get_elevenlabs_api_key() - 4 Test Cases

Test 1: Returns API key from command-line argument when provided

Description: Verifies that API key from command-line argument takes priority

Coverage: Input handling - command-line argument

Effectiveness: High - Ensures command-line argument works correctly



Test 2: Returns API key from environment variable when user confirms

Description: Validates fallback to environment variable when user confirms

Coverage: User interaction - confirmation flow

Effectiveness: High - Ensures fallback mechanism works

```
============================================================
TEST 2: Returns API key from environment variable when user confirms
============================================================


 ElevenLabs API key not present in command-line argument

Usage is like this: python syntheticRadioHostScript.py <api_key> <output_file_path>
Will read api key from environment variable .env file whose location is C:/Users/vineet.srivastava/venvGenAIStudy/.env .


✅ Using ElevenLabs API key from environment variable
✅ PASS: Returns API key from environment variable when user confirms
PASSED
test_syntheticRadioHostScript.py::TestGetElevenlabsApiKey::test_exits_when_user_rejects_env_variable
```

Test 3: Exits when user rejects proceeding with environment variable

Description: Handles user rejection and exits gracefully

Coverage: User interaction - cancellation flow

Effectiveness: High - Prevents unwanted execution

```
============================================================
TEST 3: Exits when user rejects proceeding with environment variable
============================================================


 ElevenLabs API key not present in command-line argument

Usage is like this: python syntheticRadioHostScript.py <api_key> <output_file_path>
Will read api key from environment variable .env file whose location is C:/Users/vineet.srivastava/venvGenAIStudy/.env .


❌ Exiting. Please provide API key as command-line argument or set it in .env file in the path mentioned.
✅ Using ElevenLabs API key from environment variable
✅ PASS: Exits when user rejects proceeding with environment variable
PASSED
test_syntheticRadioHostScript.py::TestGetElevenlabsApiKey::test_returns_none_when_no_api_key_found
```

Test 4: Returns None when no API key found in either location

Description: Handles case where API key is not available from any source

Coverage: Error handling - missing configuration

Effectiveness: High - Graceful degradation

```
============================================================
TEST 4: Returns None when no API key found in either location
============================================================


 ElevenLabs API key not present in command-line argument

Usage is like this: python syntheticRadioHostScript.py <api_key> <output_file_path>
Will read api key from environment variable .env file whose location is C:/Users/vineet.srivastava/venvGenAIStudy/.env .


⚠️ No ElevenLabs API key found (neither in command-line nor environment)
✅ PASS: Returns None when no API key found in either location
PASSED
test_syntheticRadioHostScript.py::TestFetchWikipediaContext::test_valid_topic_returns_context
```

5.2 fetch_wikipedia_context() - 4 Test Cases

Test 5: Valid topic returns context

Description: Verifies that a valid Wikipedia topic returns contextual information

Coverage: Happy path - normal operation

Effectiveness: High - Ensures core functionality works correctly

```
============================================================
TEST 5: Valid topic returns context
============================================================
✅ PASS: Valid topic returns context
PASSED
test_syntheticRadioHostScript.py::TestFetchWikipediaContext::test_empty_topic_returns_none
```

Test 6: Empty/None topic returns None

Description: Validates that empty, None, or whitespace-only topics are rejected

Coverage: Input validation - edge case handling

Effectiveness: High - Prevents errors from invalid inputs

```
============================================================
TEST 6: Empty/None topic returns None
============================================================
❌ Topic is empty or invalid.
❌ Topic is empty or invalid.
❌ Topic is empty or invalid.
✅ PASS: Empty/None topic returns None
PASSED
test_syntheticRadioHostScript.py::TestFetchWikipediaContext::test_nonexistent_topic_returns_none
```

Test 7: Non-existent topic returns None

Description: Handles cases where Wikipedia page does not exist

Coverage: Error handling - API response validation

Effectiveness: High - Graceful degradation on missing content

```
============================================================
TEST 7: Non-existent topic returns None
============================================================
❌ Topic 'NonExistentTopic12345' not found on Wikipedia.
✅ PASS: Non-existent topic returns None
PASSED
test_syntheticRadioHostScript.py::TestFetchWikipediaContext::test_handles_wikipedia_api_exceptions
```

Test 8: Handles Wikipedia API exceptions gracefully

Description: Catches and handles network errors, API failures, etc.

Coverage: Exception handling - robustness

Effectiveness: Critical - Prevents system crashes on API failures

```
============================================================
TEST 8: Handles Wikipedia API exceptions gracefully
============================================================
❌ Error fetching Wikipedia context: Network error
✅ PASS: Handles Wikipedia API exceptions gracefully
PASSED
test_syntheticRadioHostScript.py::TestCreateHinglishPrompt::test_returns_formatted_prompt_with_topic_and_context
```

5.3 create_hinglish_prompt() - 3 Test Cases

Test 9: Returns formatted prompt with topic and context

Description: Verifies prompt contains both topic and context information

Coverage: Output validation - data integrity

Effectiveness: High - Ensures prompt structure is correct

```
============================================================
TEST 9: Returns formatted prompt with topic and context
============================================================
✅ PASS: Returns formatted prompt with topic and context
PASSED
test_syntheticRadioHostScript.py::TestCreateHinglishPrompt::test_contains_required_instructions_and_example
```

Test 10: Contains all required instructions and example format

Description: Validates prompt includes Hinglish instructions and example

Coverage: Content validation - prompt quality

Effectiveness: High - Ensures LLM receives proper guidance

```
============================================================
TEST 10: Contains all required instructions and example format
============================================================
✅ PASS: Contains all required instructions and example format
PASSED
test_syntheticRadioHostScript.py::TestCreateHinglishPrompt::test_handles_empty_topic_context
```

Test 11: Handles empty topic/context

Description: Tests behavior with empty inputs

Coverage: Edge case - input handling

Effectiveness: Medium - Handles edge cases gracefully

```
============================================================
TEST 11: Handles empty topic/context
============================================================
✅ PASS: Handles empty topic/context
PASSED
test_syntheticRadioHostScript.py::TestCheckOllamaConnection::test_returns_true_when_ollama_accessible
```

5.4 check_ollama_connection() - 3 Test Cases

Test 12: Returns True when Ollama is accessible

Description: Verifies successful connection to Ollama service

Coverage: Connection validation - service availability

Effectiveness: High - Early detection of service issues

```
============================================================
TEST 12: Returns True when Ollama is accessible
============================================================
✅ Ollama is running and accessible
✅ PASS: Returns True when Ollama is accessible
PASSED
test_syntheticRadioHostScript.py::TestCheckOllamaConnection::test_returns_false_when_ollama_returns_none
```

Test 13: Returns False when Ollama returns None

Description: Handles case where Ollama service is not responding

Coverage: Error detection - service state

Effectiveness: High - Prevents wasted API calls

```
============================================================
TEST 13: Returns False when Ollama returns None
============================================================
❌ Cannot connect to Ollama: Response is None
   Make sure Ollama is running: ollama serve
✅ PASS: Returns False when Ollama returns None
PASSED
test_syntheticRadioHostScript.py::TestCheckOllamaConnection::test_returns_false_when_ollama_raises_exception
```

Test 14: Returns False when Ollama raises exception

Description: Catches network errors and connection failures

Coverage: Exception handling - robustness

Effectiveness: Critical - Prevents crashes on connection failures

```
============================================================
TEST 14: Returns False when Ollama raises exception
============================================================
❌ Cannot connect to Ollama: Connection error
   Error type: Exception
   Make sure Ollama is running: ollama serve
✅ PASS: Returns False when Ollama raises exception
PASSED
test_syntheticRadioHostScript.py::TestGenerateScriptWithOllama::test_returns_tuple_on_success
```

5.5 generate_script_with_ollama() - 7 Test Cases

Test 15: Returns (script, elapsed_time) tuple on success

Description: Verifies correct return type and structure

Coverage: Output validation - return type

Effectiveness: High - Ensures API contract is maintained

```
============================================================
TEST 15: Returns (script, elapsed_time) tuple on success
============================================================
⚠️ Warning: Context is very short. Script quality may be affected.
generate_script_with_ollama: topic is Test and context is Context
🤖 Generating script with Ollama (llama3.2)...
✅ Script generated successfully!
script has 20 words
Script is:  Vineet: Hello everyone! [laughs] This is a longer script to pass validation.

Simran: Hi there! Welcome to our show today!
✅ PASS: Returns (script, elapsed_time) tuple on success
PASSED
test_syntheticRadioHostScript.py::TestGenerateScriptWithOllama::test_returns_none_on_empty_short_script
```

Test 16: Returns (None, elapsed_time) on empty/short script

Description: Handles cases where LLM generates insufficient content

Coverage: Quality validation - output length

Effectiveness: High - Prevents processing of invalid scripts

```
============================================================
TEST 16: Returns (None, elapsed_time) on empty/short script
============================================================
⚠️ Warning: Context is very short. Script quality may be affected.
generate_script_with_ollama: topic is Test and context is Context
🤖 Generating script with Ollama (llama3.2)...
⚠️ Warning: Generated script is very short or empty.
✅ PASS: Returns (None, elapsed_time) on empty/short script
PASSED
test_syntheticRadioHostScript.py::TestGenerateScriptWithOllama::test_returns_none_zero_on_exception
```

Test 17: Returns (None, 0) on exception

Description: Handles API errors and exceptions

Coverage: Exception handling - error recovery

Effectiveness: Critical - Prevents system crashes

```
============================================================
TEST 17: Returns (None, 0) on exception
============================================================
⚠️ Warning: Context is very short. Script quality may be affected.
generate_script_with_ollama: topic is Test and context is Context
🤖 Generating script with Ollama (llama3.2)...
❌ Error generating script with Ollama: API error
✅ PASS: Returns (None, 0) on exception
PASSED
test_syntheticRadioHostScript.py::TestGenerateScriptWithOllama::test_measures_time_correctly
```

Test 18: Measures time correctly

Description: Validates timing measurement accuracy

Coverage: Performance monitoring - metrics

Effectiveness: Medium - Useful for performance analysis

```
============================================================
TEST 18: Measures time correctly
============================================================
⚠️ Warning: Context is very short. Script quality may be affected.
generate_script_with_ollama: topic is Test and context is Context
🤖 Generating script with Ollama (llama3.2)...
✅ Script generated successfully!
script has 13 words
Script is:  Vineet: Hello! [laughs]

Simran: Hi there! This is a longer script to test.
✅ PASS: Measures time correctly
PASSED
test_syntheticRadioHostScript.py::TestGenerateScriptWithOllama::test_handles_ollama_api_errors
```

Test 19: Handles Ollama API errors

Description: Catches and handles specific API errors

Coverage: Error handling - API failures

Effectiveness: High - Robust error handling

```
============================================================
TEST 19: Handles Ollama API errors
============================================================
⚠️ Warning: Context is very short. Script quality may be affected.
generate_script_with_ollama: topic is Test and context is Context
🤖 Generating script with Ollama (llama3.2)...
❌ Error generating script with Ollama: Ollama connection failed
✅ PASS: Handles Ollama API errors
PASSED
test_syntheticRadioHostScript.py::TestGenerateScriptWithOllama::test_output_contains_emotional_tags
```

Test 20: Output contains emotional tags like [laughs]

Description: Validates LLM output includes required emotional tags

Coverage: Output validation - content quality

Effectiveness: High - Ensures script quality for audio generation

```
============================================================
TEST 20: Output contains emotional tags like [laughs] at least once
============================================================
⚠️ Warning: Context is very short. Script quality may be affected.
generate_script_with_ollama: topic is Test and context is Context
🤖 Generating script with Ollama (llama3.2)...
✅ Script generated successfully!
script has 21 words
Script is:  Vineet: Hello everyone! [laughs] This is a longer script to pass validation.

Simran: Hi there! [giggles] Welcome to our show today!
✅ PASS: Found emotional tags: ['laughs', 'giggles']
PASSED
test_syntheticRadioHostScript.py::TestGenerateScriptWithOllama::test_output_contains_hinglish_prompts
```

Test 21: Output contains Hinglish prompts (Hmmm, hmm, umm)

Description: Validates LLM output includes natural conversation fillers

Coverage: Output validation - naturalness

Effectiveness: High - Ensures natural-sounding conversations

```
============================================================
TEST 21: Output contains Hinglish prompts like Hmmm, hmm, umm (case-insensitive)
============================================================
⚠️ Warning: Context is very short. Script quality may be affected.
generate_script_with_ollama: topic is Test and context is Context
🤖 Generating script with Ollama (llama3.2)...
✅ Script generated successfully!
script has 25 words
Script is:  Vineet: Hmmm... let me think about this. This is a longer script to pass validation.

Simran: Umm.. that is interesting! Welcome to our show today!
✅ PASS: Found Hinglish prompts: ['Hmmm', 'Umm']
PASSED
test_syntheticRadioHostScript.py::TestParseScript::test_parses_double_newline_dialogues
```

5.6 parse_script() - 10 Test Cases (Most Critical)

Test 22: Parses double newline separated dialogues

Description: Handles scripts with \n\n separators

Coverage: Format parsing - primary format

Effectiveness: Critical - Core functionality

```
============================================================
TEST 22: Parses double newline separated dialogues
============================================================
parse_script: Entering
parse_script: Using double newline splitting (3 segments)
parse_script: Parsed 3 dialogue segments
✅ PASS: Parses double newline separated dialogues
PASSED
test_syntheticRadioHostScript.py::TestParseScript::test_parses_single_newline_dialogues
```

Test 23: Parses single newline separated dialogues

Description: Handles scripts with \n separators

Coverage: Format parsing - alternative format

Effectiveness: High - Format flexibility

```
============================================================
TEST 23: Parses single newline separated dialogues
============================================================
parse_script: Entering
parse_script: Using single newline splitting (3 segments)
parse_script: Parsed 3 dialogue segments
✅ PASS: Parses single newline separated dialogues
PASSED
test_syntheticRadioHostScript.py::TestParseScript::test_handles_empty_none_script
```

Test 24: Handles empty/None script

Description: Validates empty input handling

Coverage: Input validation - edge case

Effectiveness: High - Prevents errors

```
============================================================
TEST 24: Handles empty/None script
============================================================
parse_script: Entering
parse_script: Entering
✅ PASS: Handles empty/None script
PASSED
test_syntheticRadioHostScript.py::TestParseScript::test_identifies_male_female_hosts_correctly
```

Test 25: Identifies male and female hosts correctly

Description: Correctly assigns host1/host2 based on names

Coverage: Logic validation - speaker identification

Effectiveness: Critical - Correct voice assignment

```
============================================================
TEST 25: Identifies male and female hosts correctly
============================================================
parse_script: Entering
parse_script: Using double newline splitting (3 segments)
parse_script: Parsed 3 dialogue segments
✅ PASS: Identifies male and female hosts correctly
PASSED
test_syntheticRadioHostScript.py::TestParseScript::test_handles_custom_host_names_and_markdown
```

Test 26: Handles custom host names and markdown

Description: Removes markdown and handles custom names

Coverage: Format handling - robustness

Effectiveness: High - Format flexibility

```
============================================================
TEST 26: Handles custom host names and markdown in names
============================================================
parse_script: Entering
parse_script: Using double newline splitting (3 segments)
parse_script: Parsed 3 dialogue segments
✅ PASS: Handles custom host names and markdown in names
PASSED
test_syntheticRadioHostScript.py::TestParseScript::test_handles_multiline_dialogue_text
```

Test 27: Handles multi-line dialogue text

Description: Preserves multi-line dialogue content

Coverage: Content preservation - data integrity

Effectiveness: High - Maintains script quality

```
============================================================
TEST 27: Handles multi-line dialogue text
============================================================
parse_script: Entering
parse_script: Using single newline splitting (2 segments)
parse_script: Parsed 2 dialogue segments
✅ PASS: Handles multi-line dialogue text
PASSED
test_syntheticRadioHostScript.py::TestParseScript::test_handles_scripts_with_intro_outro_text
```

Test 28: Handles scripts with intro/outro text

Description: Skips non-dialogue content

Coverage: Content filtering - noise removal

Effectiveness: High - Handles real-world LLM output

```
============================================================
TEST 28: Handles scripts with intro/outro text
============================================================
parse_script: Entering
parse_script: Using double newline splitting (3 segments)
parse_script: Parsed 2 dialogue segments
✅ PASS: Handles scripts with intro/outro text
PASSED
test_syntheticRadioHostScript.py::TestParseScript::test_handles_missing_colons_and_empty_text
```

Test 29: Handles missing colons and empty text

Description: Skips malformed segments

Coverage: Error handling - malformed input

Effectiveness: High - Robustness

```
============================================================
TEST 29: Handles missing colons and empty text after colon
============================================================
parse_script: Entering
parse_script: Using double newline splitting (3 segments)
parse_script: Parsed 2 dialogue segments
✅ PASS: Handles missing colons and empty text after colon
PASSED
test_syntheticRadioHostScript.py::TestParseScript::test_assigns_unknown_speakers_correctly
```

Test 30: Assigns unknown speakers correctly

Description: Handles speakers not matching expected names

Coverage: Logic validation - fallback behavior

Effectiveness: High - Handles edge cases

```
============================================================
TEST 30: Assigns unknown speakers correctly
============================================================
parse_script: Entering
parse_script: Using double newline splitting (3 segments)
parse_script: Parsed 3 dialogue segments
✅ PASS: Assigns unknown speakers correctly
PASSED
test_syntheticRadioHostScript.py::TestParseScript::test_chooses_best_splitting_strategy
```

Test 31: Chooses best splitting strategy automatically

Description: Selects optimal parsing method based on script format

Coverage: Algorithm validation - intelligent parsing

Effectiveness: High - Adaptive behavior

```
============================================================
TEST 31: Chooses best splitting strategy automatically
============================================================
parse_script: Entering
parse_script: Using double newline splitting (4 segments)
parse_script: Parsed 4 dialogue segments
parse_script: Entering
parse_script: Using single newline splitting (3 segments)
parse_script: Parsed 3 dialogue segments
✅ PASS: Chooses best splitting strategy automatically
PASSED
test_syntheticRadioHostScript.py::TestGenerateSegment::test_returns_audio_bytes_on_success
```

5.7 generate_segment() - 4 Test Cases

Test 32: Returns audio bytes on success (200 status)

Description: Verifies successful audio generation

Coverage: Happy path - normal operation

Effectiveness: High - Core functionality

```
============================================================
TEST 32: Returns audio bytes on success (200 status)
============================================================
✅ PASS: Returns audio bytes on success (200 status)
PASSED
test_syntheticRadioHostScript.py::TestGenerateSegment::test_returns_none_on_api_error
```

Test 33: Returns None on API error (non-200 status)

Description: Handles API errors gracefully

Coverage: Error handling - API failures

Effectiveness: High - Prevents crashes

```
============================================================
TEST 33: Returns None on API error (non-200 status)
============================================================
❌ Error generating segment: 400
Message: Bad Request
✅ PASS: Returns None on API error (non-200 status)
PASSED
test_syntheticRadioHostScript.py::TestGenerateSegment::test_returns_none_on_network_exception
```

Test 34: Returns None on network exception

Description: Catches network errors

Coverage: Exception handling - network issues

Effectiveness: High - Robustness

```
============================================================
TEST 34: Returns None on network exception
============================================================
❌ Error: Network error
✅ PASS: Returns None on network exception
PASSED
test_syntheticRadioHostScript.py::TestGenerateSegment::test_detects_emotion_tags_and_sets_voice_settings
```

Test 35: Detects emotion tags and sets correct voice settings

Description: Validates emotion detection and voice configuration

Coverage: Feature validation - emotion support

Effectiveness: High - Ensures quality audio output

```
============================================================
TEST 35: Detects emotion tags and sets correct voice settings
============================================================
✅ PASS: Detects emotion tags and sets correct voice settings
PASSED
test_syntheticRadioHostScript.py::TestValidateOutputPath::test_creates_directory_and_returns_true
```

5.8 validate_output_path() - 3 Test Cases

Test 36: Creates directory if it doesn't exist and returns True

Description: Validates directory creation and write permissions

Coverage: File system - directory management

Effectiveness: High - Prevents file write errors

```
============================================================
TEST 36: Creates directory if it doesn't exist and returns True
============================================================
📁 Creating output directory: /test/dir
✅ PASS: Creates directory if it doesn't exist and returns True
PASSED
test_syntheticRadioHostScript.py::TestValidateOutputPath::test_returns_false_for_no_write_permissions
```

Test 37: Returns False for path without write permissions

Description: Detects permission issues

Coverage: Error detection - permission validation

Effectiveness: High - Early error detection

```
============================================================
TEST 37: Returns False for path without write permissions
============================================================
❌ No write permission for: /test/dir/output.mp3
✅ PASS: Returns False for path without write permissions
PASSED
test_syntheticRadioHostScript.py::TestValidateOutputPath::test_handles_exceptions_gracefully
```

Test 38: Handles exceptions gracefully

Description: Catches file system exceptions

Coverage: Exception handling - robustness

Effectiveness: High - Prevents crashes

```
============================================================
TEST 38: Handles exceptions gracefully
============================================================
❌ Error validating output path: Path error
✅ PASS: Handles exceptions gracefully
PASSED
test_syntheticRadioHostScript.py::TestGeneratePodcast::test_returns_true_on_successful_generation
```

5.9 generate_podcast() - 8 Test Cases

Test 39: Returns True on successful generation

Description: Verifies complete podcast generation workflow

Coverage: Integration - end-to-end flow

Effectiveness: Critical - Core functionality

```
============================================================
TEST 39: Returns True on successful generation
============================================================
🎙️ Starting podcast generation...
============================================================
📝 Found 2 dialogue segments
🔧 Male Host: Vineet
🔧 Female Host: Simran

🎵 [1/2] Vineet
   Text: Hello!
   ✅ Success (saved to C:/LEARNING/LANGCHAIN/temp_segment_001.mp3)

🎵 [2/2] Simran
   Text: Hi there!
   ✅ Success (saved to C:/LEARNING/LANGCHAIN/temp_segment_002.mp3)

💾 Combining all segments into final podcast...
============================================================
✅ Podcast generated successfully!
📁 File: output.mp3
📝 Combined 2 segments

✅ PASS: Returns True on successful generation
PASSED
test_syntheticRadioHostScript.py::TestGeneratePodcast::test_returns_false_on_invalid_inputs
```

Test 40: Returns False on invalid inputs

Description: Validates API key, dialogue, and voice IDs

Coverage: Input validation - parameter checking

Effectiveness: High - Prevents invalid operations

```
============================================================
TEST 40: Returns False on invalid API key, empty dialogue, or invalid voice IDs
============================================================
⚠️ Please set your ELEVEN_LABS_API_KEY first!
❌ No dialogue provided. Cannot generate podcast.
❌ Invalid voice IDs provided.
✅ PASS: Returns False on invalid API key, empty dialogue, or invalid voice IDs
PASSED
test_syntheticRadioHostScript.py::TestGeneratePodcast::test_returns_false_on_path_validation_failure
```

Test 41: Returns False on path validation failure

Description: Handles file system validation errors

Coverage: Error handling - path validation

Effectiveness: High - Prevents file errors

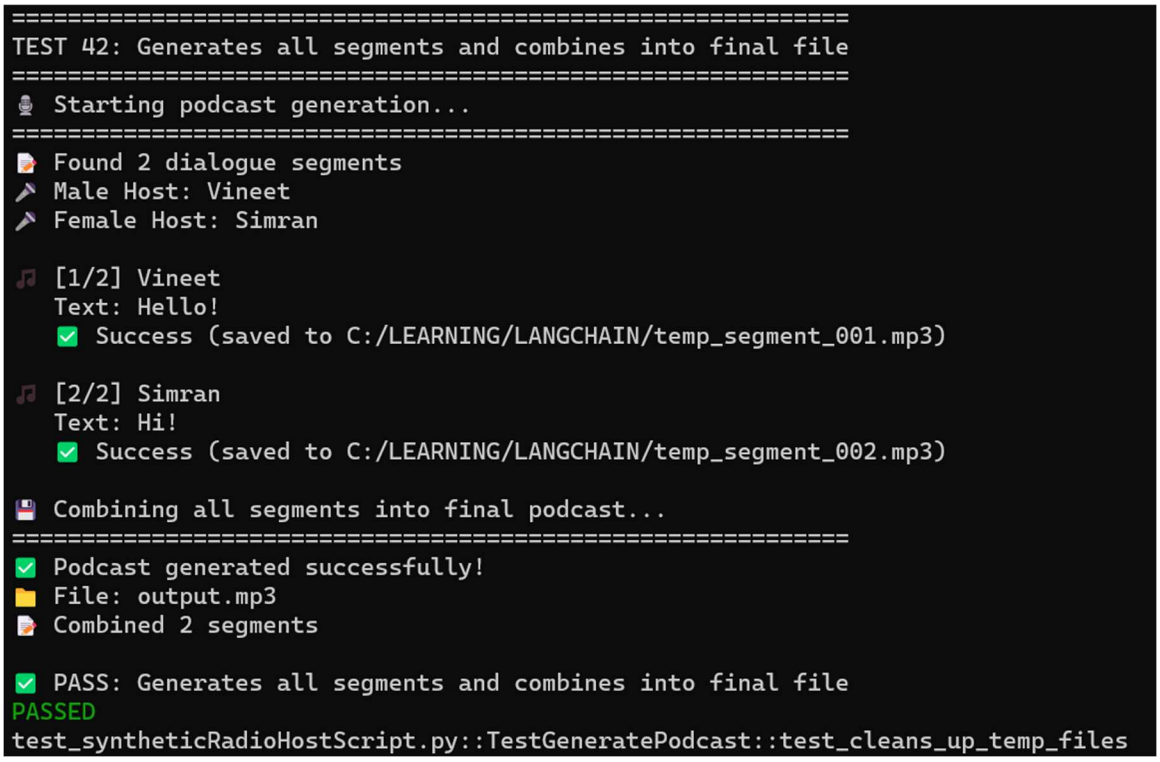
```
============================================================
TEST 41: Returns False on path validation failure
============================================================
✅ PASS: Returns False on path validation failure
PASSED
test_syntheticRadioHostScript.py::TestGeneratePodcast::test_generates_segments_and_combines_file
```

Test 42: Generates all segments and combines into final file

Description: Verifies complete audio generation and combination

Coverage: Integration - workflow completion

Effectiveness: Critical - End-to-end validation



Test 43: Cleans up temp files

Description: Verifies temporary file removal

Coverage: Resource management - cleanup

Effectiveness: High - Prevents disk space issues

```
============================================================
TEST 43: Cleans up temp files
============================================================
🎙️ Starting podcast generation...
============================================================
📝 Found 1 dialogue segments
🔨 Male Host: Vineet
🔨 Female Host: Host2

🎵 [1/1] Vineet
   Text: Hello!
   ✅ Success (saved to C:/LEARNING/LANGCHAIN/temp_segment_001.mp3)

💾 Combining all segments into final podcast...
============================================================
✅ Podcast generated successfully!
📁 File: output.mp3
📝 Combined 1 segments

✅ PASS: Cleans up temp files
PASSED
test_syntheticRadioHostScript.py::TestGeneratePodcast::test_handles_segment_generation_failures_gracefully
```

Test 44: Handles segment generation failures gracefully

Description: Continues processing even if some segments fail

Coverage: Error handling - partial failures

Effectiveness: High - Robustness

```
============================================================
TEST 44: Handles segment generation failures gracefully
============================================================
🎙️ Starting podcast generation...
============================================================
📝 Found 2 dialogue segments
🔨 Male Host: Vineet
🔨 Female Host: Simran

🎵 [1/2] Vineet
   Text: Hello!
   ✅ Success (saved to C:/LEARNING/LANGCHAIN/temp_segment_001.mp3)

🎵 [2/2] Simran
   Text: Hi!
   ❌ Failed to generate

💾 Combining all segments into final podcast...
============================================================
✅ Podcast generated successfully!
📁 File: output.mp3
📝 Combined 1 segments

✅ PASS: Handles segment generation failures gracefully
PASSED
test_syntheticRadioHostScript.py::TestGeneratePodcast::test_handles_file_io_errors
```

Test 45: Handles file I/O errors

Description: Catches file read/write exceptions

Coverage: Exception handling - I/O errors

Effectiveness: High - Prevents crashes

```
============================================================
TEST 45: Handles file I/O errors
============================================================
🎙️ Starting podcast generation...
============================================================
📝 Found 1 dialogue segments
🔧 Male Host: Vineet
🔧 Female Host: Host2

🎵 [1/1] Vineet
   Text: Hello!
   ❌ Failed to save temp file: Disk full

❌ No segments were generated successfully!
✅ PASS: Handles file I/O errors
PASSED
test_syntheticRadioHostScript.py::TestGeneratePodcast::test_handles_partial_segment_failures
```

Test 46: Handles partial segment failures

Description: Handles mixed success/failure scenarios

Coverage: Error handling - complex scenarios

Effectiveness: High - Real-world robustness

```
============================================================
TEST 46: Handles partial segment failures (some succeed, some fail)
============================================================
🎙️ Starting podcast generation...
============================================================
📝 Found 3 dialogue segments
🎤 Male Host: Vineet
🎤 Female Host: Simran

🎵 [1/3] Vineet
   Text: Hello!
   ✅ Success (saved to C:/LEARNING/LANGCHAIN/temp_segment_001.mp3)

🎵 [2/3] Simran
   Text: Hi!
   ❌ Failed to generate

🎵 [3/3] Vineet
   Text: Again!
   ✅ Success (saved to C:/LEARNING/LANGCHAIN/temp_segment_003.mp3)

💾 Combining all segments into final podcast...
============================================================
✅ Podcast generated successfully!
📁 File: output.mp3
📝 Combined 2 segments

✅ PASS: Handles partial segment failures (some succeed, some fail)
PASSED
test_syntheticRadioHostScript.py::TestMainIntegration::test_complete_flow_success
```

5.10 main() Integration Tests - 5 Test Cases

Test 47: Complete flow: context → script → parse → podcast

Description: Tests entire system workflow end-to-end

Coverage: Integration - full system test

Effectiveness: Critical - System validation

```
============================================================
TEST 47: Complete flow: context → script → parse → podcast
============================================================
✅ Using output file path from command-line argument: -v
============================================================
🎙️  SYNTHETIC RADIO HOST - HINGLISH PODCAST GENERATOR
============================================================
📌 Topic: Mumbai Indians
📁 Output File: -v

🔍 Checking Ollama connection...

📚 Fetching Wikipedia context...
✅ Context fetched (12 characters)

⏱️  Time taken to generate script: 2 second(s)

📝 Generated Script Preview:
------------------------------------------------------------
Vineet: Hello!

Simran: Hi!
------------------------------------------------------------

📋 Parsing script into dialogue segments...
   Looking for hosts: Vineet and Simran
⚠️ Warning: Only 2 dialogue segments found. Expected more.
   This might indicate a parsing issue. Check script format.
   Debug: First few segments found:
   1. Vineet: Hello!...
   2. Simran: Hi!...
✅ Parsed 2 dialogue segments
dialogue:  [('host1', 'Vineet', 'Hello!'), ('host2', 'Simran', 'Hi!')]

============================================================
✅ Proceeding with audio generation...

🎉 All done! Your podcast is ready at: -v
✅ PASS: Complete flow: context → script → parse → podcast
PASSED
test_syntheticRadioHostScript.py::TestMainIntegration::test_handles_missing_api_key
```

Test 48: Handles missing API key gracefully

Description: Validates behavior when API key is missing

Coverage: Error handling - configuration issues

Effectiveness: High - User experience

```
============================================================
TEST 48: Handles missing API key gracefully
============================================================
✅ Using output file path from command-line argument: -v
============================================================
🎙️  SYNTHETIC RADIO HOST - HINGLISH PODCAST GENERATOR
============================================================
📌 Topic: Mumbai Indians
📁 Output File: -v

🔍 Checking Ollama connection...

📚 Fetching Wikipedia context...
✅ Context fetched (7 characters)

⏱️  Time taken to generate script: 2 second(s)

📝 Generated Script Preview:
------------------------------------------------------------
Script
------------------------------------------------------------

📋 Parsing script into dialogue segments...
   Looking for hosts: Vineet and Simran
⚠️ Warning: Only 1 dialogue segments found. Expected more.
   This might indicate a parsing issue. Check script format.
   Debug: First few segments found:
   1. Vineet: Hello!...
✅ Parsed 1 dialogue segments
dialogue:  [('host1', 'Vineet', 'Hello!')]

============================================================
🎵 Do you want to generate audio files? (Y/N): ❌ Audio generation cancelled by user. Exiting from Program. BYE!!!
✅ PASS: Handles missing API key gracefully
PASSED
test_syntheticRadioHostScript.py::TestMainIntegration::test_handles_ollama_connection_failure
```

Test 49: Handles Ollama connection failure

Description: Tests early exit on service unavailability

Coverage: Error handling - service availability

Effectiveness: High - Prevents wasted processing

```
============================================================
TEST 49: Handles Ollama connection failure
============================================================
✅ Using output file path from command-line argument: -v
============================================================
🎙️  SYNTHETIC RADIO HOST - HINGLISH PODCAST GENERATOR
============================================================
📌 Topic: Mumbai Indians
📁 Output File: -v

🔍 Checking Ollama connection...
❌ Ollama connection check failed. Exiting.
✅ PASS: Handles Ollama connection failure
PASSED
test_syntheticRadioHostScript.py::TestMainIntegration::test_handles_wikipedia_fetch_failure
```

Test 50: Handles Wikipedia fetch failure

Description: Tests behavior when content cannot be fetched

Coverage: Error handling - content retrieval

Effectiveness: High - Graceful degradation

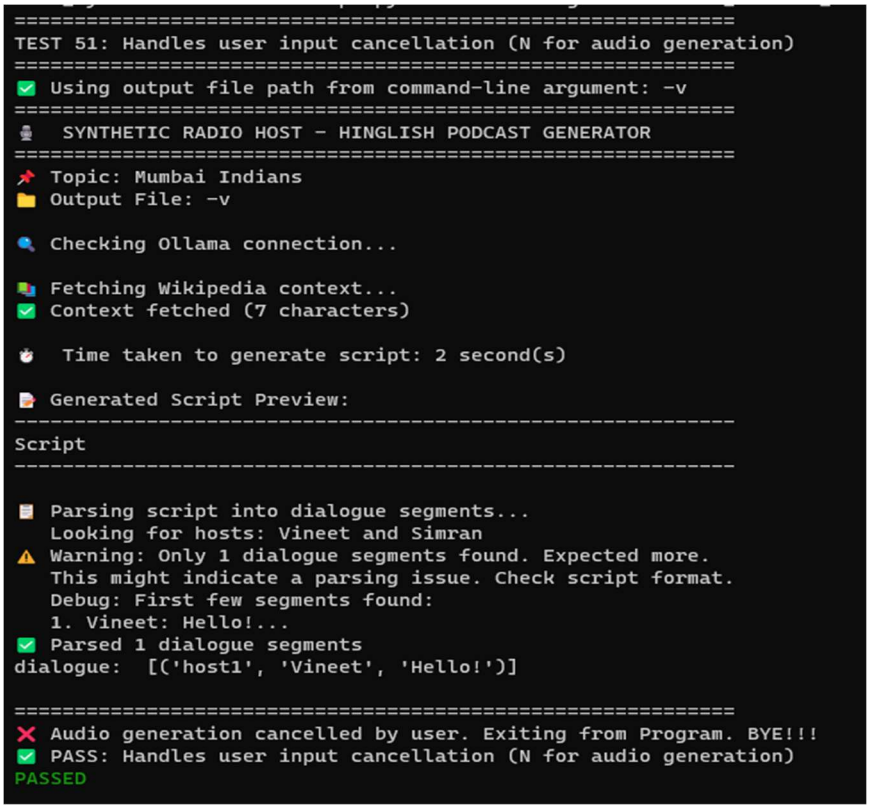
```
============================================================
TEST 50: Handles Wikipedia fetch failure
============================================================
✅ Using output file path from command-line argument: -v
============================================================
🎙️  SYNTHETIC RADIO HOST - HINGLISH PODCAST GENERATOR
============================================================
📌 Topic: Mumbai Indians
📁 Output File: -v

🔍 Checking Ollama connection...

📚 Fetching Wikipedia context...
❌ Failed to fetch context. Exiting.
✅ PASS: Handles Wikipedia fetch failure
PASSED
test_syntheticRadioHostScript.py::TestMainIntegration::test_handles_user_input_cancellation
```

Test 51: Handles user input cancellation

Description: Tests user cancellation of audio generation

Coverage: User interaction - cancellation flow

Effectiveness: High - User experience and cost control



AND FINALLY, THE OUTPUT COMES LIKE AS GIVEN BELOW: