Verbatimly App Summarized Test-Plan

PREPARED BY: Theophilus Onolunose

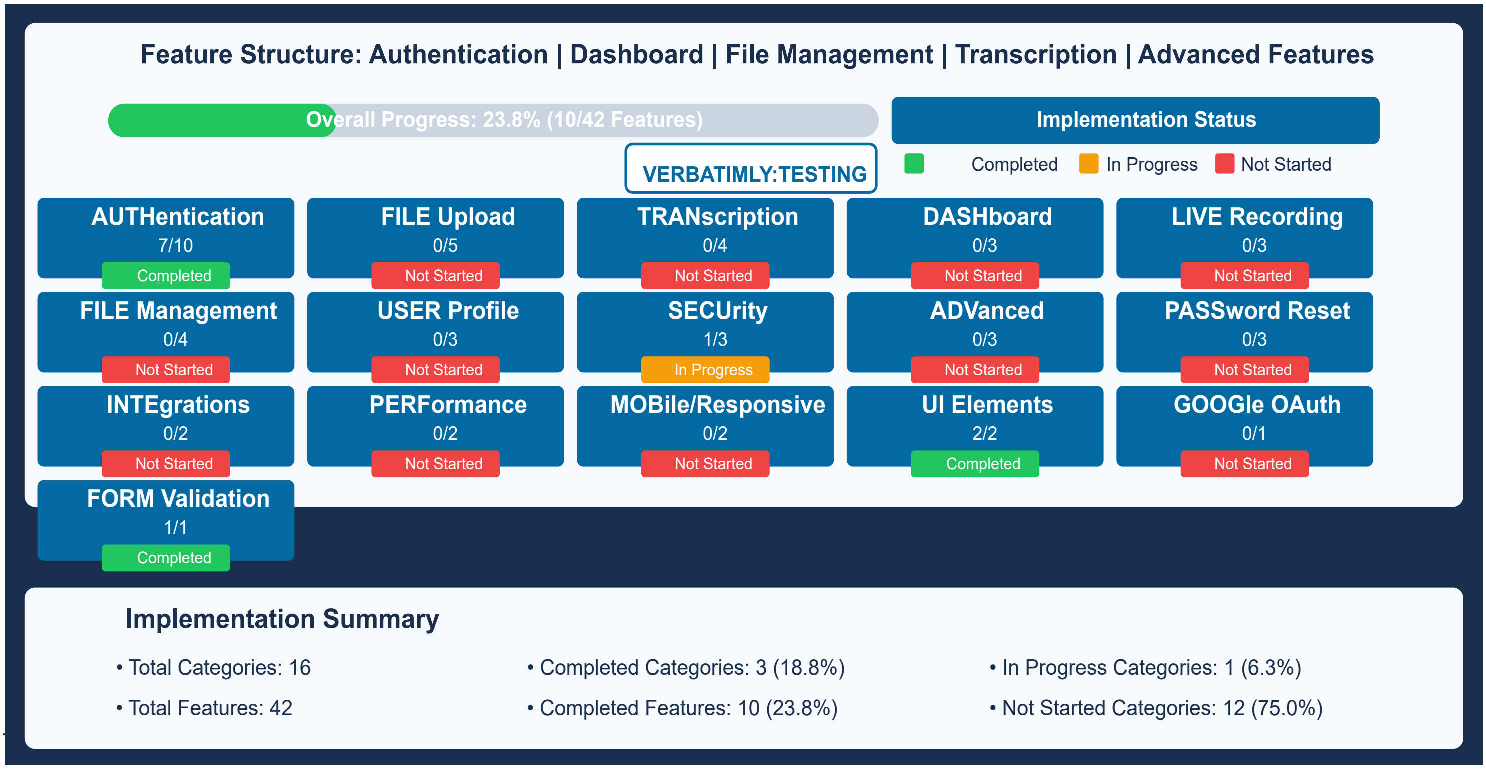
DATE: 28 July 2025

1. Overview

As a quality assurance initiative, I am motivated to execute a comprehensive automated testing regimen for the web application located at "https://verbatimly.com".

The product owned by Tech von chu' at Kolpingstr. 83, 53359, Rheinbach, Germany

Fig. 1: Overview of the Features to be Tested (Test Items) and the Current Test Coverage:



Feature Structure: Authentication | Dashboard | File Management | Transcription | Advanced Features

Overall Progress: 23.8% (10/42 Features)

VERBATIMLY:TESTING

Implementation Status

🟩 Completed 🟧 In Progress 🟥 Not Started

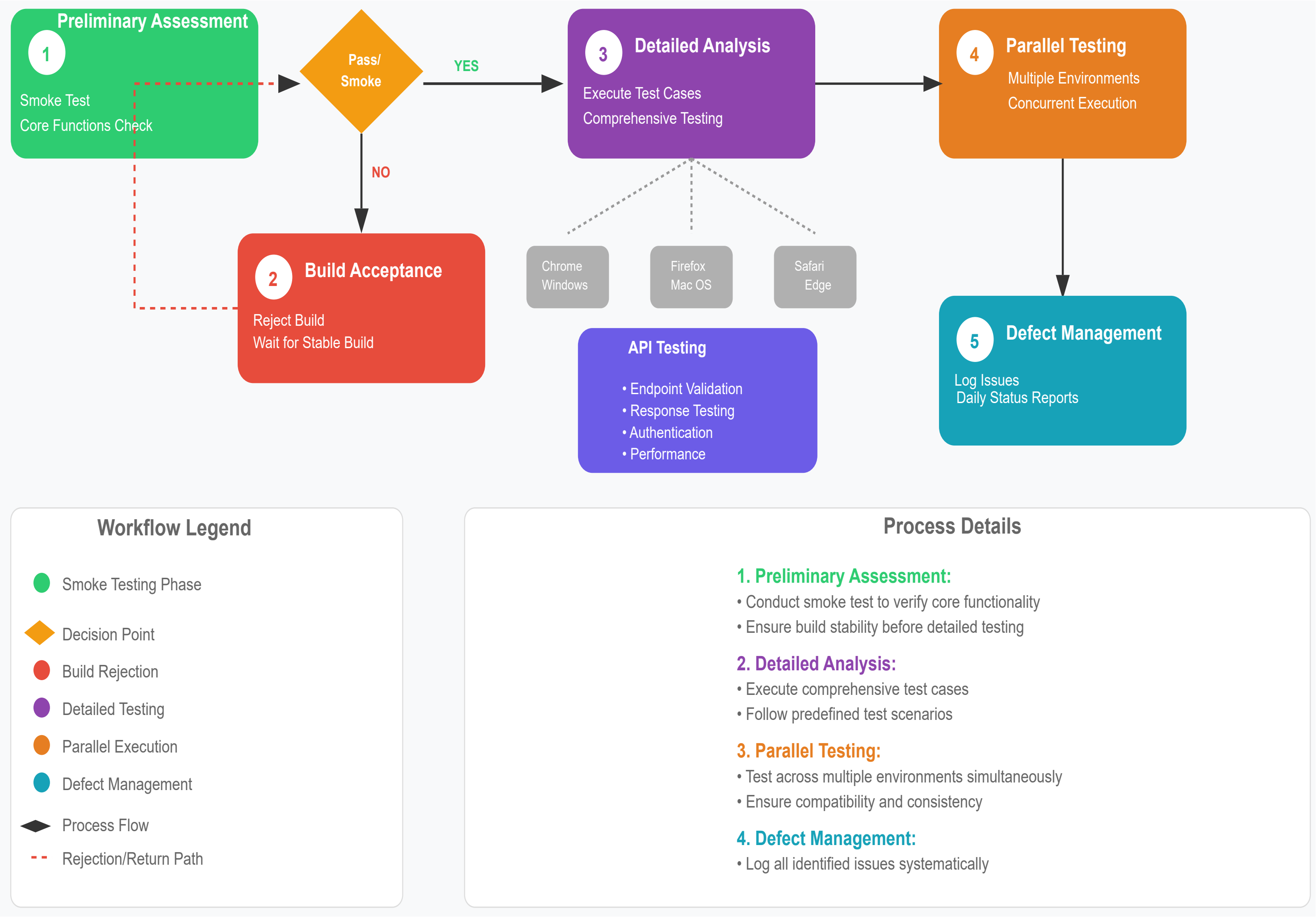
AUTHentication 7/10 Completed	FILE Upload 0/5 Not Started	TRANScription 0/4 Not Started	DASHboard 0/3 Not Started	LIVE Recording 0/3 Not Started
FILE Management 0/4 Not Started	USER Profile 0/3 Not Started	SECUrity 1/3 In Progress	ADVanced 0/3 Not Started	PASSword Reset 0/3 Not Started
INTEgrations 0/2 Not Started	PERFormance 0/2 Not Started	MOBile/Responsive 0/2 Not Started	UI Elements 2/2 Completed	GOOGle OAuth 0/1 Not Started
FORM Validation 1/1 Completed				

Implementation Summary

- Total Categories: 16
- Total Features: 42
- Completed Categories: 3 (18.8%)
- Completed Features: 10 (23.8%)
- In Progress Categories: 1 (6.3%)
- Not Started Categories: 12 (75.0%)

2. Structured Test Workflow

Fig. 2: Overview of the Structured Workflow to Test the Product.:



Process Details

1. Preliminary Assessment:
- Conduct smoke test to verify core functionality
- Ensure build stability before detailed testing

2. Detailed Analysis:
- Execute comprehensive test cases
- Follow predefined test scenarios

3. Parallel Testing:
- Test across multiple environments simultaneously
- Ensure compatibility and consistency

4. Defect Management:
- Log all identified issues systematically

3. Test Environment:

- **Windows 11** - Chrome, Firefox, and Edge
- **Mac OS** – Safari Browser

4. Test Strategy

With the application's user interface and functionality provided by "Verbatimly" as their online platform, I have discovered the necessity for a comprehensive functional testing regimen tailored to the functionalities delineated in the "Test Items" section. The strategy approach is illustrated below:

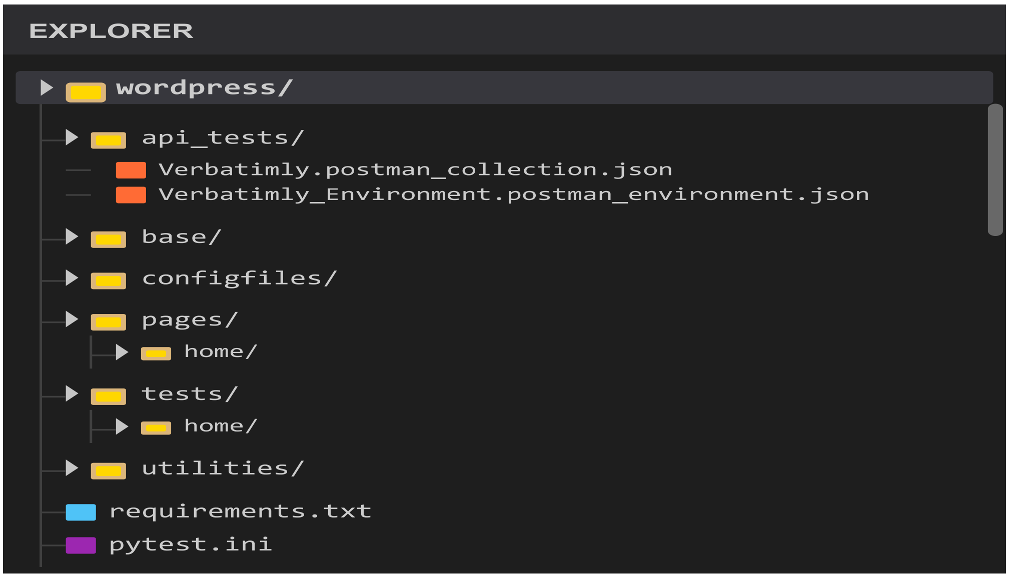
- **Test Design:** Systematic development of test scenarios that encapsulate the diverse functionalities within the testing scope.
- **Test Execution:** Detailed set of test steps is the testing procedure encompassing input data, expected response, executing a holistic evaluation of each feature. Though the test cases delineation is still in progress as of this report..
- **Execution Strategy:** Leveraging automated testing tools and methodologies to ensure precision and efficiency in the testing efforts.

5. Key Testing Tools and Management:

- **Selenium WebDriver:** Automates browser actions to validate end-to-end web application functionalities as part of a free source tool.
- **Pytest Framework:** Provides structured test execution with fixtures, assertions, and parameterization for maintainable automation.
- **Postman for API Test REST APIs:** Validates responses, and automates API workflows with collection. As of now only few endpoints are testable (Observable and controllable)

6. Verbatimly Test Automation Framework

1. Project Structure



EXPLORER

```
wordpress/
    api_tests/
        Verbatimly.postman_collection.json
        Verbatimly_Environment.postman_environment.json
    base/
    configfiles/
    pages/
        home/
    tests/
        home/
    utilities/
    requirements.txt
    pytest.ini
```

2. Prerequisites

- pip install -r requirements.txt (Check the ReadMe documentation it has steps to set up the framework and run tests in less than 2mins.

3. Run API Tests (Postman via Newman)

```
cd verbatimly_tests
pytest tests/test_API_tests/Verbatimly.postman_collection.json \
   -e api_tests/Verbatimly_Environment.postman_environment.json
```

4. Run Selenium UI Tests (Pytest)

Login tests

```
pytest tests/home/login_tests.py -v --disable-warnings -s
```

Create Account Tests

```
pytest tests/home/createAccount_tests.py -v --disable-warnings -s
```

5. Logs Reports

- Execution logs: automation.log
- Screenshots: Screenshot shows failed test screenshots