WEB APPLICATION

TEST PLAN

Table of Contents

1. Introduction

Purpose

The purpose of this test plan is to provide a proof of concept for a basic automation framework used for testing different API calls to a local web server.

Project Overview

The project is split into two parts : on one hand we have the source code for the web server and on the other a test script where we define the test conditions, test data and the test cases to be executed. The web server can be initiated from either debug mode (using Visual Studio) or from the build itself (worth mentioning that we need to modify the appsettings.json to have basic functionality working). The test script is built in the xUnit framework.

2. Scope

In-Scope

Test API endpoints using GET, POST, PUT, DELETE.

Validation of correct response status codes.

Testing of negative scenarios based on application's constraints.

Manual testing of the endpoints using web interface APIs.

Delivery of a test report with results.

Out-of-Scope

End-to-end automation testing in a CI/CD environment due to time constraints (could be considered for implementation when product reaches a higher level of maturity).

3. Testing Strategy

Test Objectives

Test the backend of an API driven web application using an automated test framework.

Test Assumptions

At this current stage of implementation, testing assumes that the web server is being run on the same environment as the test framework and that all dependencies have been covered.

Data Approach

*Currently the test data is hard-coded in the test script itself in the form of **InlineData** and covers both positive and negative scenarios, according to the requirements specified in the project's Readme file.*

Level of Testing

Test Type	Description	Responsible Parties
Functional	*Ensure the correct functionality of the APIs under test according to the requirements (query task list and their status, add/delete tasks, update tasks etc.)*	**Automation Testing Team**
Non-Functional	*Ensure that the web app responds within an acceptable amount of time; ensure redirection of http requests to https.*	
User Acceptance	*Access the web interface of the server and perform manual validation of the APIs.*	**Manual Testing Team**

4. Test Case List

Test Name	Test Steps	Expected Output	Execution
Check Server Response	Send GET request to server uri.	Response code is 200 or OK and format application/json	Automation
Check For Empty Response Content	1. Delete all existing tasks. 2. Send GET request to server uri.	1. Response code is 200 or OK 2. Server returns an empty list	
Create Task With Valid Data			
Create Task with Invalid Data			
Update Existing Task			
Update Non-Existing Task			

Delete Existing Task			
Delete Non-Existing Task			
Check Response Time			
Check Server Limit			

5. Execution Strategy

Entry Criteria

Entry Criteria	Test Team	Notes
Test environment(s) is available		
Test data is available		
Code has been merged successfully		
Development has completed unit testing		
Test scripts are completed, reviewed and approved by the Test Manager and Customer		

Exit criteria

Exit Criteria	Test Team	Notes
100% Test Scripts executed		
90% pass rate of Test Scripts		
No open Critical and High severity defects		
All remaining defects are either cancelled or documented as Change Requests for a future release		
All expected and actual results are captured and documented with the test script		
All defects logged in -Defect Tracker/Spreadsheet		

Test environment cleanup completed		

Validation and Defect Management

Validation of test output should be done against the requirements, by analyzing the test output and the execution logs.

Defects found during the Testing should be categorized as below:

Severity	Impact
1 (Critical)	▪ *Functionality is blocked and no testing can proceed* ▪ *Application/program/feature is unusable in the current state*
2 (High)	▪ *Functionality is not usable and there is no workaround but testing can proceed*
3 (Medium)	▪ *Functionality issues but there is workaround for achieving the desired functionality*
4 (Low)	▪ *Unclear error message or cosmetic error which has minimum impact on product use.*

6. Environment Requirements

Test Environments

- *Specify the test environment(s) requirements*
- *Specify the security requirements.*

7. Dependencies

Identify any dependencies on testing, such as test-item availability, testing-resource availability, and deadlines.