Test Plan for Number to Word Converter

Introduction

This test plan outlines the testing strategy for the Number to Word Converter application, developed using .NET Core for the backend and React with TypeScript for the frontend. The purpose of this testing effort is to ensure the application functions as intended, meets specified requirements, and is free of defects.

Objectives

- Validate the accuracy of the number-to-word conversion functionality.
- Ensure the application provides appropriate responses for valid and invalid inputs.
- Confirm that the user interface is intuitive and user-friendly.

Scope

The testing scope includes:

- Unit testing for backend logic.
- Integration testing to validate communication between the frontend and backend.
- Functional testing for user interface components.

Types of Testing

- **Unit Testing**: Test individual components and methods in isolation to validate the correctness of the logic. This includes testing the number-to-words conversion logic and input validation.
- **Integration Testing**: Test interactions between the frontend and backend, ensuring that data is transmitted correctly, and the API responds as expected.
- **Functional Testing**: Validate that the application meets functional requirements by testing user interactions, input handling, and output accuracy.

Testing Tools

- **Backend Testing**: NUnit for unit tests in .NET Core.
- **Frontend Testing**: Jest and React Testing Library for unit and integration tests in the React application.

Test Cases

Unit Test Cases	
Test Description	**Expected Result**
Validate conversion of a whole number (e.g., "123")	ONE HUNDRED AND TWENTY-THREE
Validate conversion of a decimal number (e.g., "123.45")	ONE HUNDRED AND TWENTY-THREE DOLLARS AND FORTY-FIVE CENTS
Validate handling of invalid input (e.g., "abc")	BadRequestObjectResult
Integration Test Cases	
Ensure frontend sends valid request to backend API	API returns the expected response
Ensure frontend handles API errors gracefully	User sees an error message
Functional Test Cases	
Enter a valid number and click "Convert"	Converted words are displayed correctly
Enter invalid input and click "Convert"	Error message is displayed

Test Environment

- **Backend**: .NET Core environment with required dependencies installed.
- **Frontend**: React application running on Node.js.

Test Schedule

- **Unit Testing**: Ongoing during development.
- **Integration Testing**: After unit testing and before functional testing.
- **Functional Testing**: After integration testing.

Conclusion

This test plan outlines a comprehensive approach to testing the Number to Words Converter application. By following this plan, we aim to deliver a reliable and robust solution that meets user needs and expectations.