****Acceptance Test Procedure (ATP)****

#### ****Project Information:****

* **Project Name:** [Your Project Name]
* **Project ID:** [Project ID]
* **Version:** [Version Number]
* **Date:** [Date]

#### ****Document Control:****

* **Prepared By:** [Your Name]
* **Reviewed By:** [Reviewer's Name]
* **Approval By:** [Approver's Name]
* **Document ID:** [Document ID]
* **Version History:**

| Version | Date | Author | Description |
| --- | --- | --- | --- |
| 1.0 | [Date] | [Your Name] | Initial draft |
| 1.1 | [Date] | [Reviewer Name] | Reviewed and suggested modifications |
| 1.2 | [Date] | [Approver Name] | Approved |

#### ****Table of Contents:****

1. **Introduction** 1.1 Purpose  
   1.2 Scope  
   1.3 Objectives
2. **Features to be Tested** 2.1 User Authentication  
   2.2 Recasting Functionality  
   2.3 Data Security  
   2.4 Error Handling  
   2.5 Performance
3. **Testing Approach** 3.1 Test Levels  
   3.2 Test Types  
   3.3 Entry Criteria  
   3.4 Exit Criteria
4. **Test Cases**

4.1 Test Case ID 1: Recaster configuration  
4.2 Test Case ID 2: Recasting Functionality  
4.3 Test Case ID 3: Data Security  
4.4 Test Case ID 4: Error Handling  
4.5 Test Case ID 5: Performance

1. **Test Environment** 5.1 Hardware Requirements  
   5.2 Software Requirements  
   5.3 Browser Compatibility
2. **Roles and Responsibilities** 6.1 Test Team  
   6.2 Development Team  
   6.3 Product Owner
3. **Schedule** 7.1 Test Preparation  
   7.2 Test Execution  
   7.3 Defect Tracking and Resolution  
   7.4 Final Report
4. **Risks and Contingencies** 8.1 Identified Risks  
   8.2 Mitigation Strategies
5. **Approval and Sign-off**

#### 1. Introduction:

##### 1.1 Purpose:

The purpose of this document is to outline the acceptance testing plan for the web server-based recasting application. It provides a comprehensive overview of the testing scope, objectives, and approach.

##### 1.2 Scope:

The acceptance testing will cover the functional and non-functional aspects of the recasting application, ensuring that it meets the specified requirements and performs reliably in a production environment.

##### 1.3 Objectives:

* To validate the Recaster server configuration using the yaml configuration file.
* To verify the recasting functionality, including input and output.
* To assess the data security measures implemented.
* To evaluate error handling and recovery mechanisms.
* To measure the performance of the application under specified conditions.

#### 2. Features to be Tested:

##### 2.1 Server configuration:

* Verify that the application can be configured using yaml configuration file.
* Check the config file validity before allowing the configuration.
  + Check that there are only unique platform names.
  + Check that there are only unique channel names for each platform.
  + Check that the input URL is not null.
  + Check the validity of the input URL.
  + Check that there are only unique output URLs.
  + Check the validity of the output URLs.
  + Ensure that there is no collision between input and output URLs.

##### 2.2 Recasting Functionality:

* Test the recasting process for accuracy and completeness.
* Validate that input parameters are processed correctly.
* Ensure that the output is generated as expected.

##### 2.3 Data Security:

* Verify the encryption and protection of sensitive data.
* Confirm that user data is securely stored and accessed.

##### 2.4 Error Handling:

* Test the application's response to invalid inputs.
* Confirm that error messages are clear and informative.

##### 2.5 Performance:

* Evaluate the application's response time under normal and peak loads.
* Assess the scalability of the web server.

#### 3. Testing Approach:

##### 3.1 Test Levels:

* Acceptance Testing

##### 3.2 Test Types:

* Functional Testing
* Security Testing
* Performance Testing

##### 3.3 Entry Criteria:

* Completion of unit and integration testing.
* Availability of the test environment.

##### 3.4 Exit Criteria:

* Successful completion of all test cases.
* Resolution of critical defects.

**4.1 Testing Objective**

The primary objectives of this testing are:

- To validate that the validation functions identify and handle duplicate names and URLs appropriately.

- To ensure that the functions raise errors for invalid or missing data.

- To confirm that the functions work as intended for various input scenarios.

**4.2 Scope of Testing**

The testing will cover the following aspects:

- Execution of test cases for each validation function.

- Verification of error messages and expected behavior.

- Integration testing to ensure proper collaboration between validation functions.

**4.3 Test Environment**

The testing environment should include the necessary dependencies for running the TypeScript code, including Node.js and any required modules. Ensure that the environment mirrors the production environment as closely as possible.

**4.4 Test Approach**

The testing approach involves functional testing, creating test cases that cover a range of scenarios, including valid and invalid data, edge cases, and expected error conditions.

**4.5 Testing results:**

**Configuration Validation Tests**

* Unique Platform Names:

The platforms were named with the same name – an error was thrown.

The platforms were named with unique names and the test passed.

* Unique Channel Names for Each Platform:

The channels were named with the same names in the same platform – an error was thrown.

The channels were named with unique names in each platform and the test passed.

* Input URL isn't Null:

An input URL was set as null – an error was thrown.

An input URL was not set as null and the test passed.

* Input URL is Valid:

An invalid URL was set – an error was thrown.

A valid URL was set and the test passed.

* Check the Uniqueness of the Output URL:

The same output URLs were placed – an error was thrown.

Unique output URLs were set and the test passed.

* Output URL is Valid:

An invalid output URL was set – an error was thrown.

A valid output URL was set and the test passed.

* Check For Collision Between the Input and Output URLs:

The same URLs were placed, and thus, causing a collision - an error was thrown.

Unique URLs were set and the test passed.

**Functional Validation Tests**

* Stream Timeout:

When the stream is cut off, the circle's color is orange, the bitrate is 0 Mbits/s and the output bitrate is also 0 Mbits/s.

When the stream is resumed, the circle's color is green, the bitrate is at the correct rate and so is the output bitrate – test passed.

* Stop Platform:

When the stream in on, and the button is pressed, the stream stops.

* Start Platform:

When pressing the start button, the stream starts.

* Adding a New Platform:

A new platform will be added if all the necessary fields are filled correctly.

If at least one of the fields are incorrectly set, then an error sign will be shown.

* Adding a New Channel:

A new channel will be added if all the necessary fields are filled correctly.

If at least one of the fields are incorrectly set, then an error sign will be shown.

* Adding a New Output:

A new Output will be added if all the necessary fields are filled correctly.

If at least one of the fields are incorrectly set, then an error sign will be shown.