System Test Plan

For

EasyNN/EasyAI

Team member:

Version/Author	Date
Nathan Rose	3/11
Jack Nguyen	3/11
David Fadini	3/11
Liam Kehoe	3/11
Nathan Foster	3/11

Table of Contents

1. Introduction

1.1 Purpose

This document is a test plan for the EasyNN/EasyAI System Testing, produced by the System Testing team. It describes the testing strategy and approach to testing the team will use to verify that the application meets the established requirements of the business before release.

1.2 Objectives

- Meets the requirements, specifications, and Business rules.
- Supports the intended business functions and achieves the required standards.
- Satisfies the Entrance Criteria for User Acceptance Testing.

2. Functional Scope

The Modules in the scope of testing for the EasyNN/EasyAI System Testing are mentioned in the document attached in the following path: Overall strategy, execution plan, traceability matrix

3. Overall Strategy and Approach

3.1 Testing Strategy

EasyNN/EasyAI System Testing will include testing of all functionalities that are in scope (Refer Functional Scope Section) identified. System testing activities will include the testing of new functionalities, modified functionalities, screen level validations, workflows, functionality access, testing of internal & external interfaces.

3.2 System Testing Entrance Criteria

To start system testing, certain requirements must be met for testing readiness. The readiness can be classified into a functioning server, an operable neural network algorithm, and some level of integration of the two.

3.3 Testing Types

3.3.1 Usability Testing

User interface attributes, cosmetic presentation, and the content will be tested for accuracy and general usability. The goal of Usability Testing is to ensure that the User Interface is comfortable to use and provides the user with consistent and appropriate access and navigation through the functions of the application (e.g., access keys, consistent tab order, readable fonts, etc.)

3.3.1.1

The EasyAI developers will test EasyNN for usability.

3.3.1.2

The EasyNN developers will test EasyAI for usability.

3.3.2 Functional Testing

The objective of this test is to ensure that each element of the component meets the functional requirements of the business as outlined in the:

- Business / Functional Requirements
- Business rules or conditions
- Other functional documents produced during the project i.e. resolution to issues/change requests/feedback

3.4 Suspension Criteria and Resumption Requirements

This section will specify the criteria that will be used to suspend all or a portion of the testing activities on the items associated with this test plan.

3.4.1 Suspension Criteria

Testing will be suspended if the incidents found will not allow further testing of the system/application under-test. If testing is halted, and changes are made to the hardware, software, or database, it is up to the Testing Manager to determine whether the test plan will be re-executed or part of the plan will be re-executed.

3.4.2 Resumption Requirements

Resumption of testing will be possible when the functionality that caused the suspension of testing has been retested successfully.

4. Execution Plan

4.1 Execution Plan

The execution plan will detail the test cases to be executed. The Execution plan will be put together to ensure that all the requirements are covered. The execution plan will be designed to accommodate some changes if necessary if testing is incomplete on any day. All the test cases of the projects under test in this release are arranged in a logical order depending upon their interdependency.

EasyNN:

1. EasyNN will be downloaded via pip.
2. An EasyNN non-dense model will be trained on the MNest database with default values
    a. The model shall achieve 90% accuracy within an hour of training.
3. An EasyNN dense model will be trained on the MNest database with default values
    a. The model shall achieve 90% accuracy within an hour of training.
4. The Machine Learning methods will be tested with the following parameters:
    a. Known functions: https://en.wikipedia.org/wiki/Test_functions_for_optimization.
5. The model shall be saved and then a new instance of the model will be loaded that matches the previous.
6. The EasyNN model is configured to use the following structures:
    a. non-dense feed-forward
    b. dense feed-forward
    c. non-dense recurrent
    d. dense recurrent
    e. Markov chain
    f. Convolutional
    g. NeuroEvolving non-dense feed-forward
7. An EasyNN model is configured to run with the following optimizers
    a. Gradient descent
    b. Momentum-based gradient descent

c. Stochastic gradient descent
d. Perturbed gradient descent
e. Adagrad
f. Adadelta
g. RMSprop
h. BFGS
i. Broyden's method
j. PDF
k. SR1
l. L-BFGS

EasyAI:

1. EasyAI will be accessible via http://3.92.61.129/
2. The test user's input of account registration will be saved to a database
    a. Verify that the test user's credentials and information that was inputted were successfully saved
3. The test user's account credentials will be entered into the log in page
4. EasyAI integration with EasyGA
    a. A form to pass the parameters into a function for calling the EasyGA module
5. EasyAI integration with EasyNN
    a. A form to pass the parameters into a function for calling the EasyNN module
6. The server creates a storage area for users to have
7. An admin page will be accessible to only those with admin credentials
8. A User page will be made for any registered user

5. Traceability Matrix & Defect Tracking

5.1 Traceability Matrix

List of requirement, corresponding test cases

Requirement Number (from SRS)	Covered by Test:
3.1.1.1	EasyNN: 4
3.1.1.2	EasyNN: 2, 3
3.1.1.3	EasyNN: 5
3.1.1.4	EasyNN: 6
3.1.1.5	EasyNN: 7

3.1.1.6	EasyNN: 2, 3
3.1.2.1	EasyAI: 1
3.1.2.2	EasyAI: 2
3.1.2.3	EasyAI: 4, 5
3.1.2.4	EasyAI: 4, 5
3.1.2.5	EasyAI: 4, 5
3.1.2.6	EasyAI: 6
3.1.2.7	EasyAI: 8
3.1.2.8	EasyAI: 7

5.2 Defect Severity Definitions

Critical	The defect causes a catastrophic or severe error that results in major problems and the functionality rendered is unavailable to the user. A manual procedure cannot be either implemented or a high effort is required to remedy the defect. Examples of a critical defect are as follows: ● System abends ● Data cannot flow through a business function/lifecycle ● Data is corrupted or cannot post to the database
Medium	The defect that does not seriously impair system function can be categorized as a medium Defect. A manual procedure requiring medium effort can be implemented to remedy the defect. Examples of a medium defect are as follows: ● Form navigation is incorrect ● Field labels are not consistent with global terminology
Low	The defect is cosmetic or has little to no impact on system functionality. A manual procedure requiring low effort can be implemented to remedy the defect. Examples of a low defect are as follows: ● Repositioning of fields on screens ● Text font on reports is incorrect

6. Environment

6.1 Environment

- The System Testing Environment will be used for System Testing.

7. Assumptions

- The user will have a computer with an internet connection.
- The user will have pip installed on their computer.

8. Risks and Contingencies

Define risks and contingencies.

9. Appendices