**Classification of Research Articles Using NLP and Machine learning**

**Design Document**

**Version 1.0**



**Group Id: S2302A8683 (MC220204248)**

**Supervisor Name: Saad Ahmed**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date (dd/mm/yyyy)** | **Version** | **Description** | **Author** |
| 27/8/2023 | 1.0 | Introduction of the project | MC220204248 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

1. [Introduction of Design Document](#One)

1. [Methodology](#Methodology)
2. [Sequence Diagrams](#Six)
3. [Architecture Design Diagram](#Seven)
4. [Interface Design](#interfacedesign)
5. [Test Cases](#testcases)

1. **Introduction of Design Document**

The design document deals with system design to accurately implement all software requirements identified in the Software Requirements Specification (SRS) document. This document helps project supervisors to monitor the progress of the software development and analyze the effectiveness of the architecture proposed for the project. This exercise also helps developers to plan their schedule for development phase and decide the most efficient code structure for the implementation of devised design. Software testing team can use the design document to build test cases for verification of system use cases.

The software design phase involves three phases:

1. Architecture design: Architecture design involves the specification of the major components of a system, their responsibilities, properties and interfaces. In architectural design, the system is decomposed into major components, and functional responsibilities are allocated to each component. Each component is assigned performance constraints so that it can function reliably in the overall system. Component Interfaces are also defined for interaction between components. This ensures that the components can perform coherently to achieve desired functionality.
2. Interface design: Interface design involves the specification of the interaction between a system and its user. It describes the events from the user to which the system must respond along with the response that the system must produce. Data input elements specify the format of data received from the user, whereas data output elements specify the data format generated by the system. Additionally, sequence diagrams are used to specify the timing relationships between these incoming events and system outputs.
3. Detailed design: Detailed design involves the specification of the implementation details of all major system components. It includes allocation of functional responsibilities to system components, controlling flow of information between components, and choice of technology, algorithms and data structures for software development.

This section presented the introduction of design document, its importance and uses. Section 2 presents the design methodology for the Natural Language Processing system and Section 3 presents the sequence diagrams for each use case. The architecture design diagram of the system is presented in Section 4, whereas Section 5 presents the user interface diagram for the system. Finally, test cases are given in Section 6 for the testing of system functionality.

1. **Methodology**

This section presents the methodology for the design of the Natural Language Processing (NLP) system. The design methodology diagram is shown in Figure 1 below. The system is based on the Model View Controller (MVC) model.

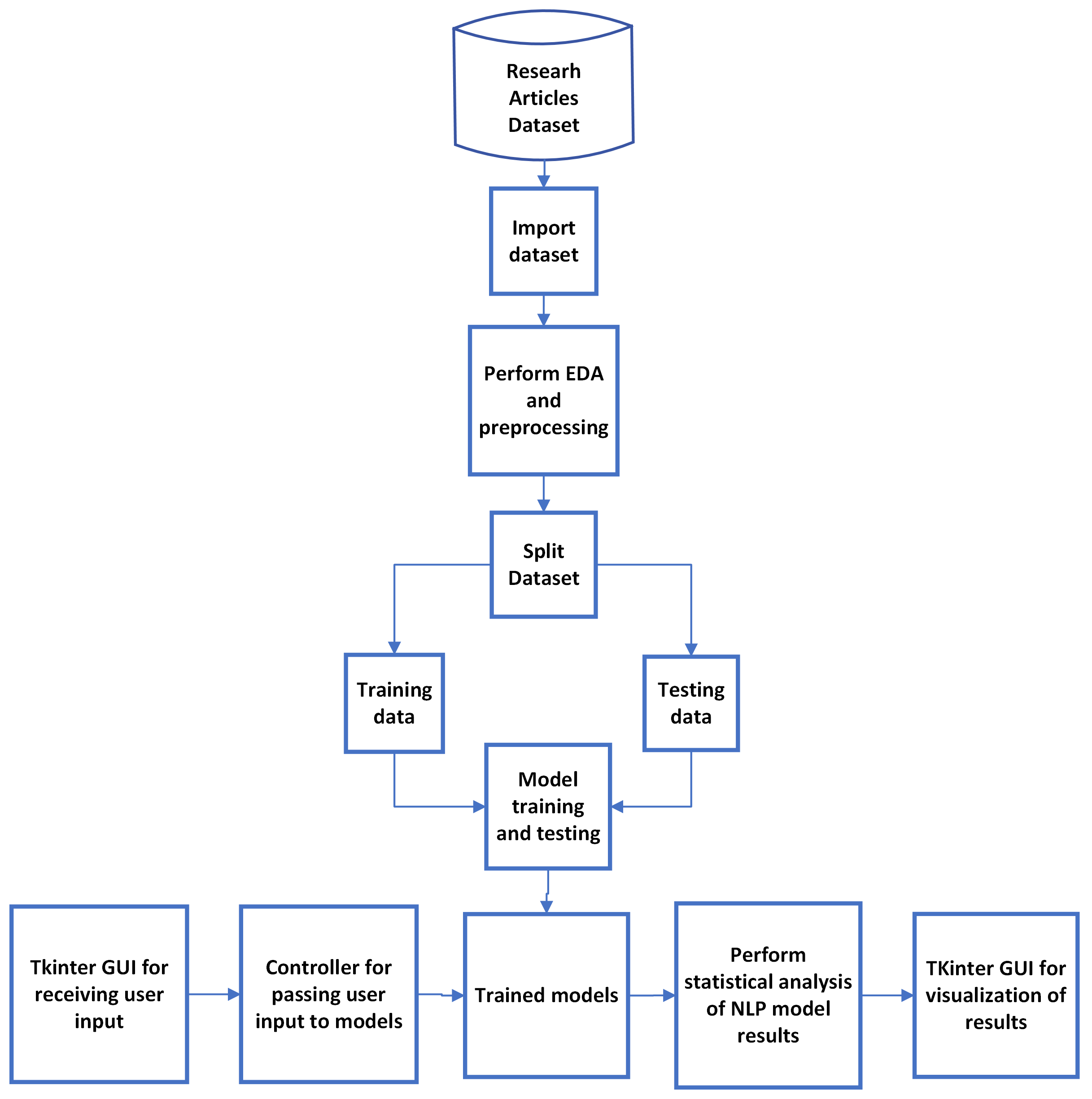


Figure 1: Design Methodology.

The design consists of four modules:

1. The I/O module will read dataset
2. The preprocessing module will perform EDA and preprocess training/ testing data.
3. The NLP training module will and train NLP models.
4. The NLP testing module will test NLP models.
5. The user input module will receive and preprocess user data.
6. The analysis module will analyze the results of NLP models.
7. The visualization module will display results of NLP models.
8. **Sequence Diagrams**
9. Use Case 1: Import dataset

The sequence diagram for use case 1 is given in Figure 2 below. The user will click button to import training/ testing data. This will cause the Tkinter event listener to read train and test data from storage. The data will be saved when it is read, and the user will be informed about the successful operation using Tkinter GUI update.

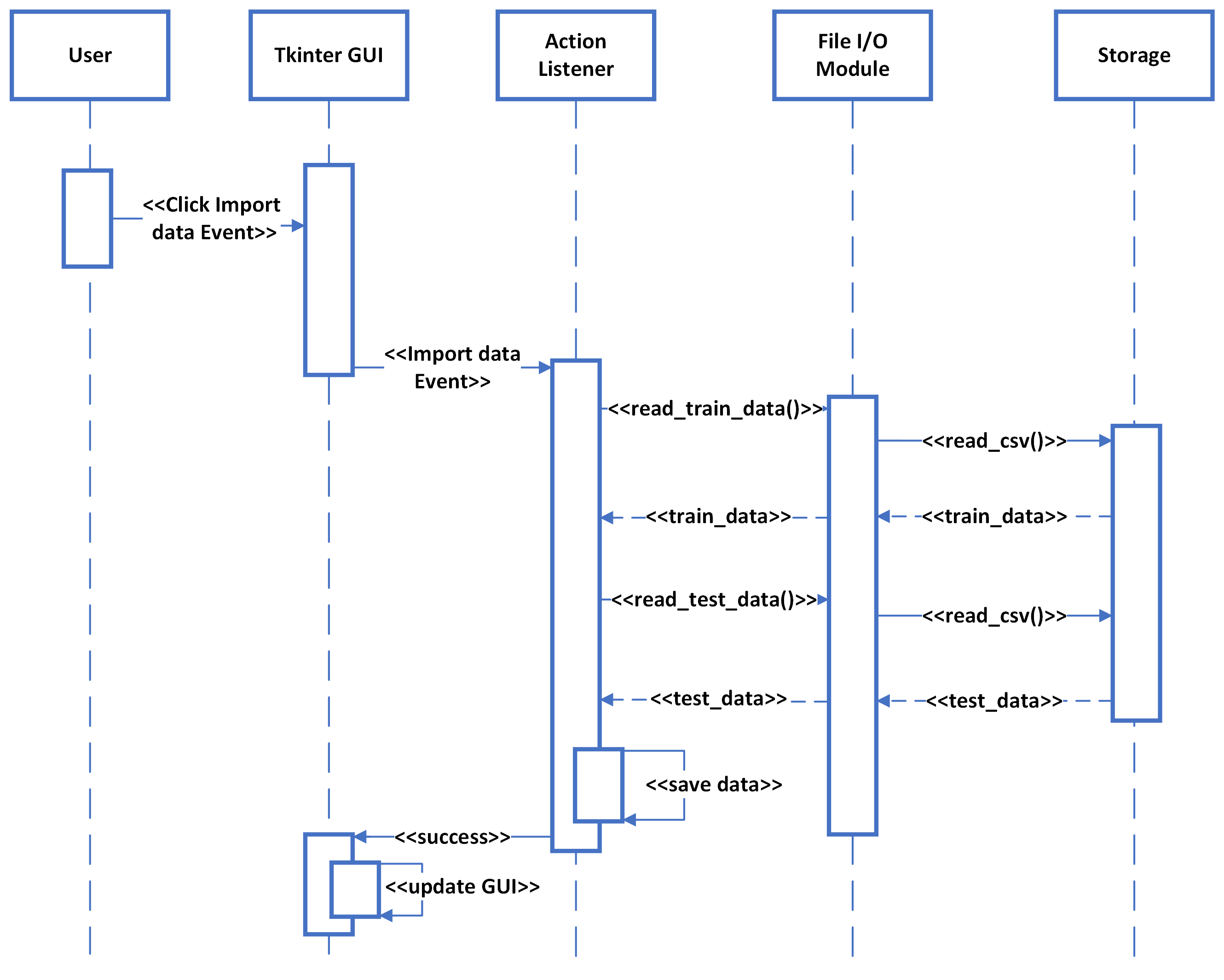


Figure 2: Sequence Diagram for Use Case 1.

1. Use Case 2: Perform EDA and preprocessing

The sequence diagram for use case 2 is given in Figure 3 below. The user will click button to perform EDA and preprocessing. This will cause the Tkinter event listener to preprocess train and test data. Preprocessing will involve tokenization, lemmatization, stop words removal and vectorization. The user will be informed about the EDA results using Tkinter GUI update.

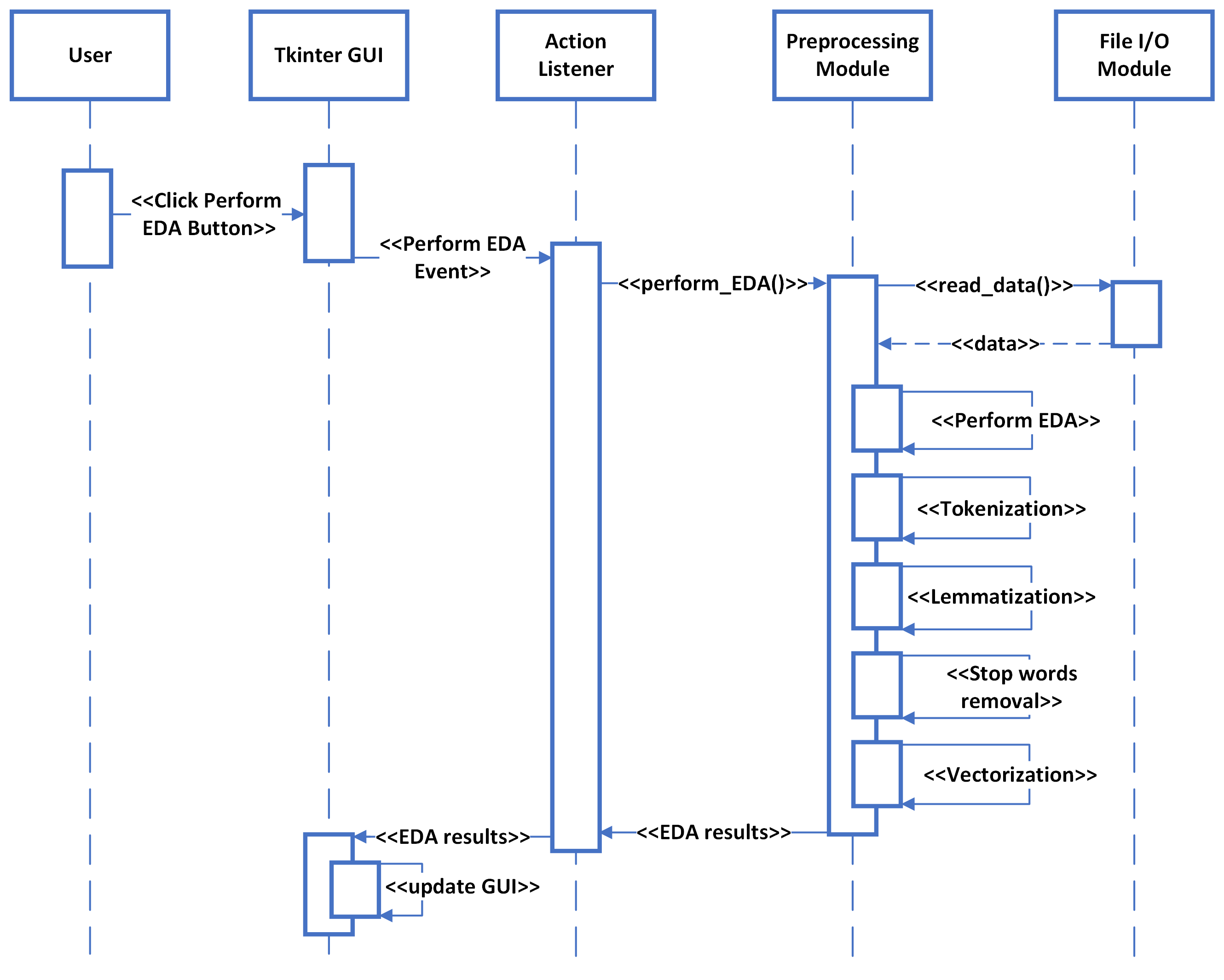


Figure 3: Sequence Diagram for Use Case 2.

1. Use Case 3: Split train and test data

The sequence diagram for use case 3 is given in Figure 4 below. The user will click button to split data into train and test. This will cause the Tkinter event listener to split train and test data. The user will be informed about the success using Tkinter GUI update.

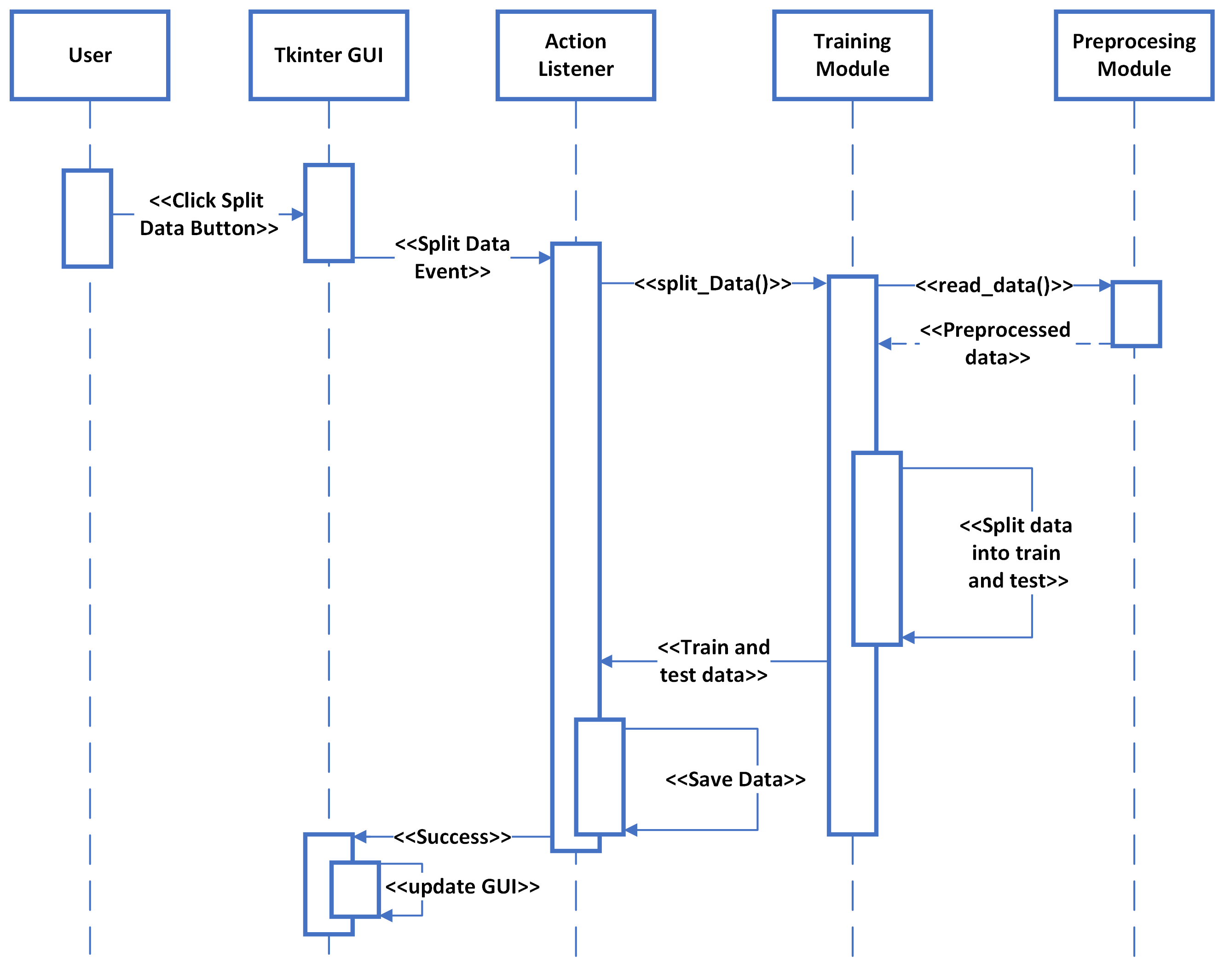


Figure 4: Sequence Diagram for Use Case 3.

1. Use Case 4: Train models

The sequence diagram for use case 4 is given in Figure 5 below. The user will click button to train NLP models. This will cause the Tkinter event listener to train four machine learning models using train dataset. The user will be informed about the training results using Tkinter GUI update.

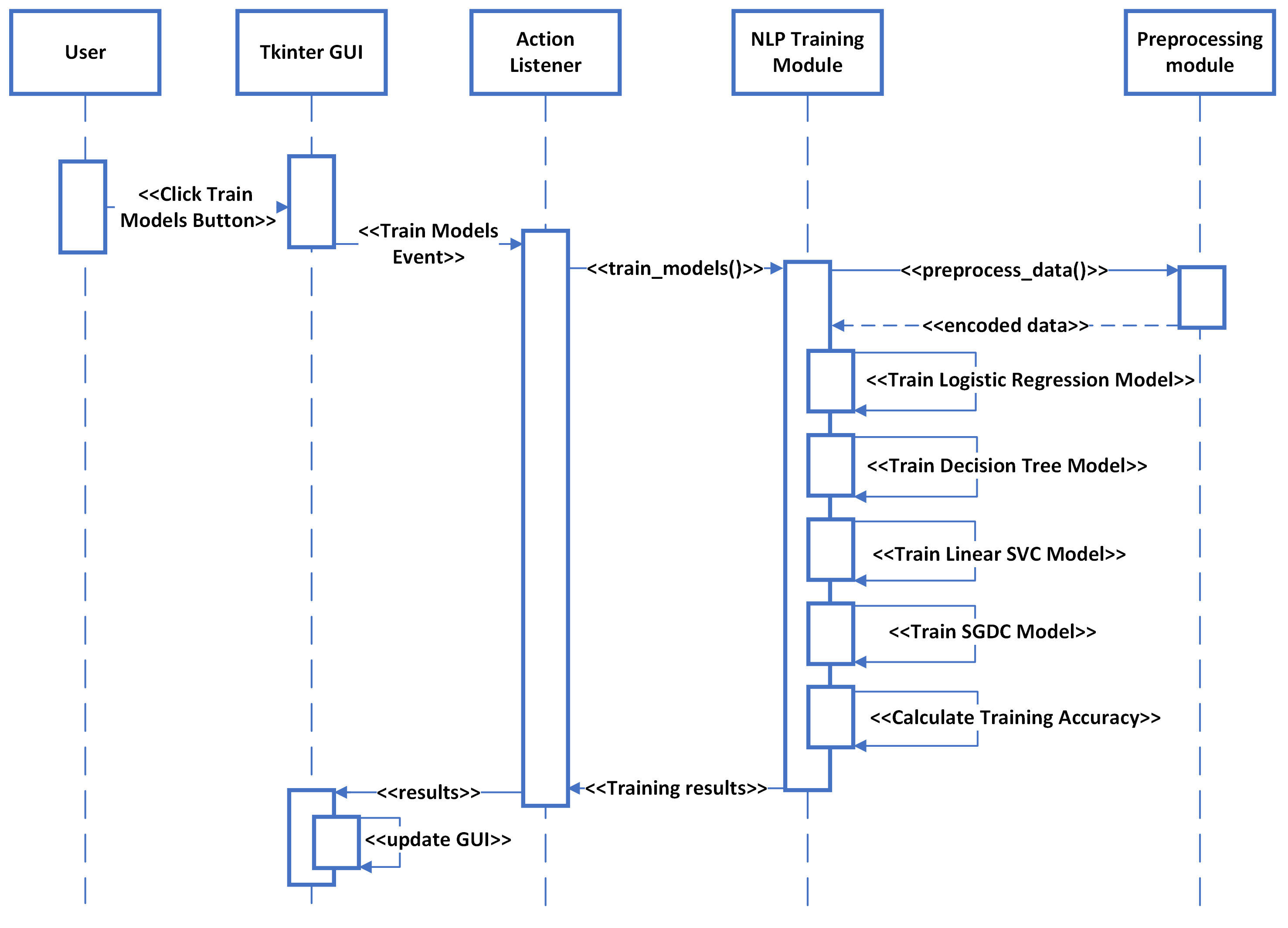


Figure 5: Sequence Diagram for Use Case 4.

1. Use Case 5: Save trained models

The sequence diagram for use case 5 is given in Figure 6 below. The user will click button to save NLP models. This will cause the Tkinter event listener to save four machine learning models. The user will be informed about the successful export using Tkinter GUI update.

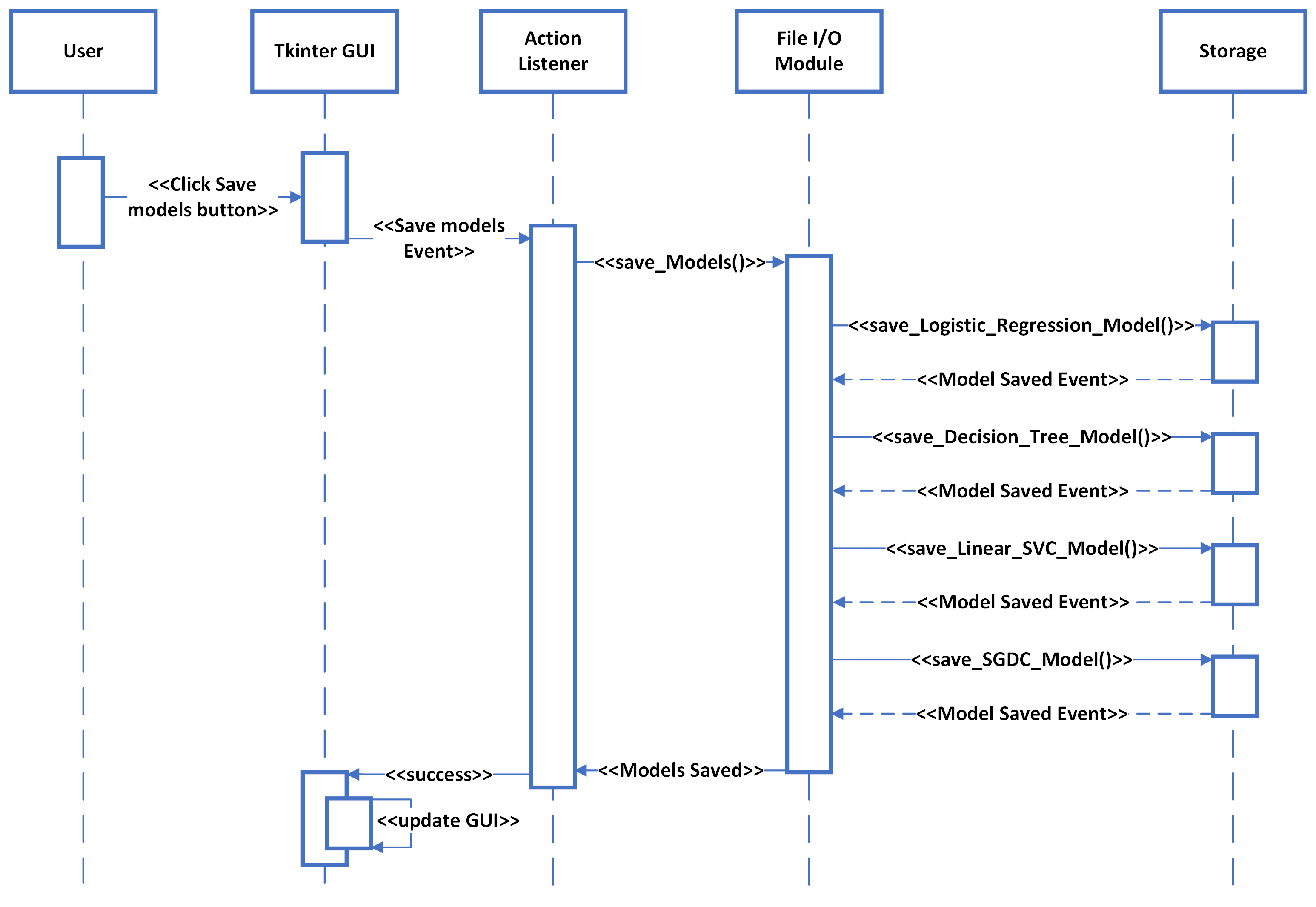


Figure 6: Sequence Diagram for Use Case 5.

1. Use Case 6: Test models

The sequence diagram for use case 6 is given in Figure 7 below. The user will click button to test the trained NLP models. This will cause the Tkinter event listener to test the four trained machine learning models using test data. The user will be informed about the testing results using Tkinter GUI update.

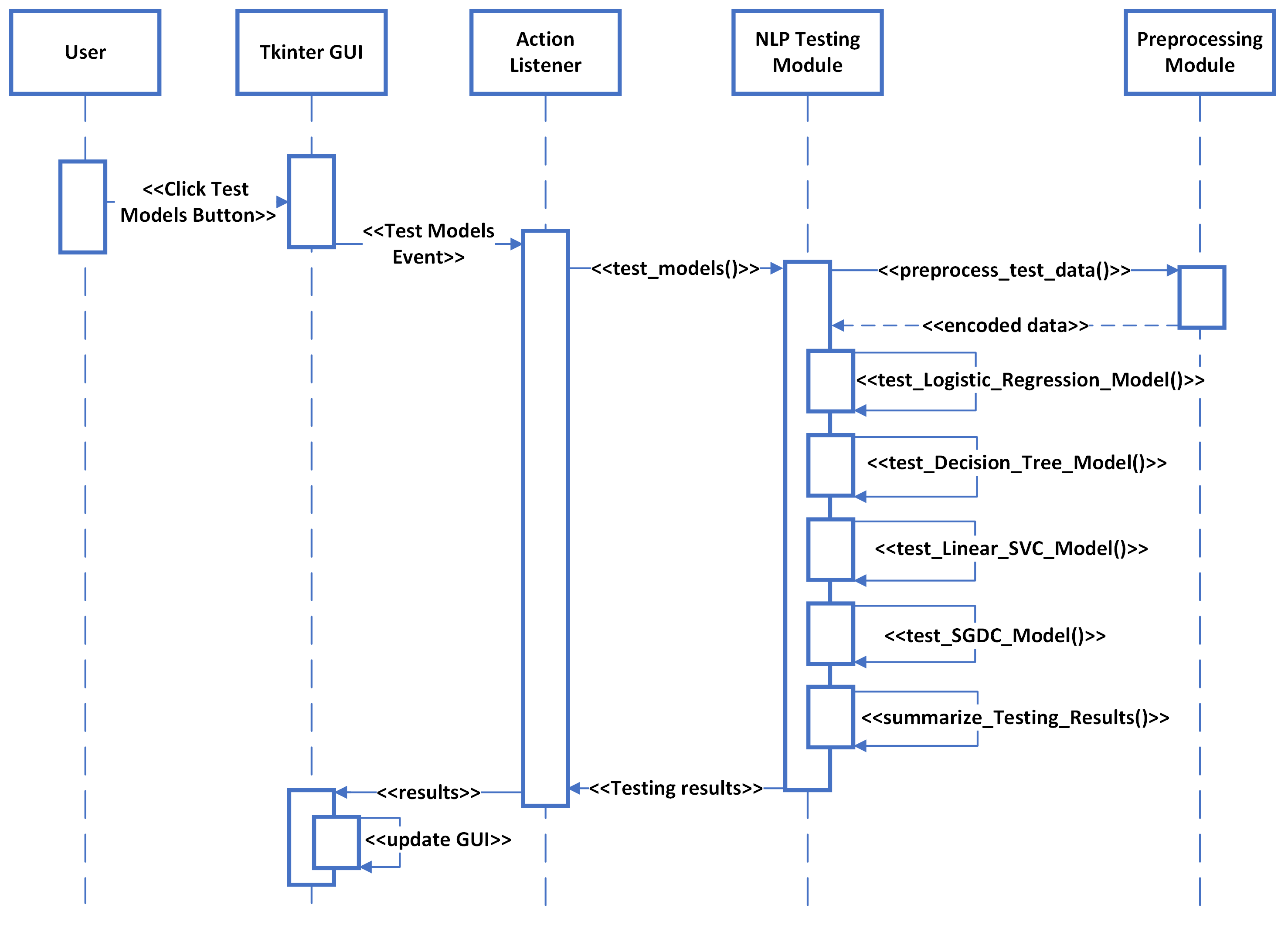


Figure 7: Sequence Diagram for Use Case 6.

1. Use Case 7: Import trained models

The sequence diagram for use case 7 is given in Figure 8 below. The user will click button to import the trained NLP models. This will cause the Tkinter event listener to import the four trained machine learning models. The user will be informed about the successful import using Tkinter GUI update.

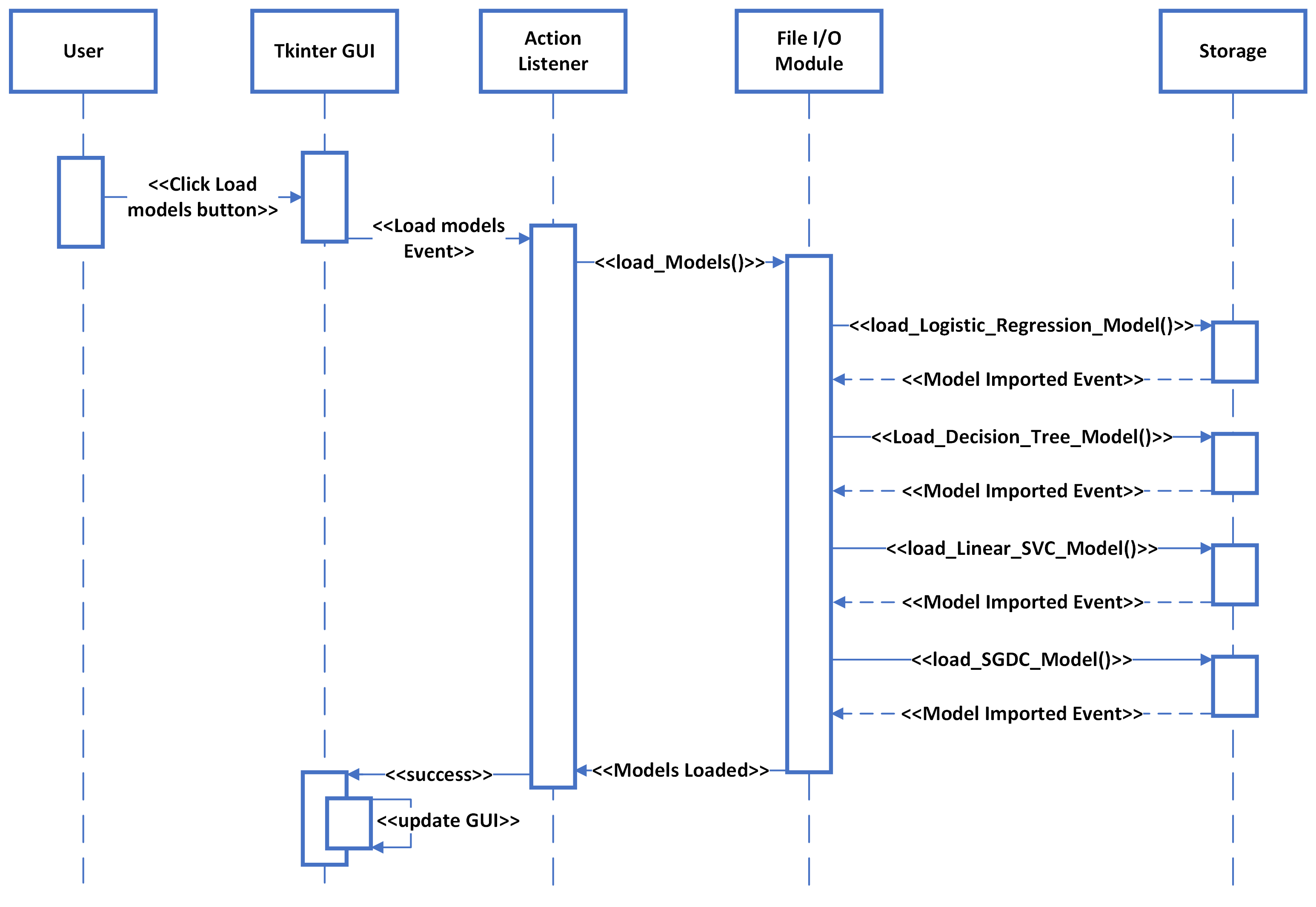


Figure 8: Sequence Diagram for Use Case 7.

1. Use Case 8: Enter title and abstract of research article and Use Case 9: Test model to categorize new research article

The sequence diagram for use case 8 and 9 is given in Figure 9 below. The user will enter research article title and abstract, and click the button to classify the research article. This will cause the Tkinter event listener to pass the data to preprocessing module. The preprocessed data will be input to the four trained machine learning models. The user will be informed about the classification results using Tkinter GUI update.

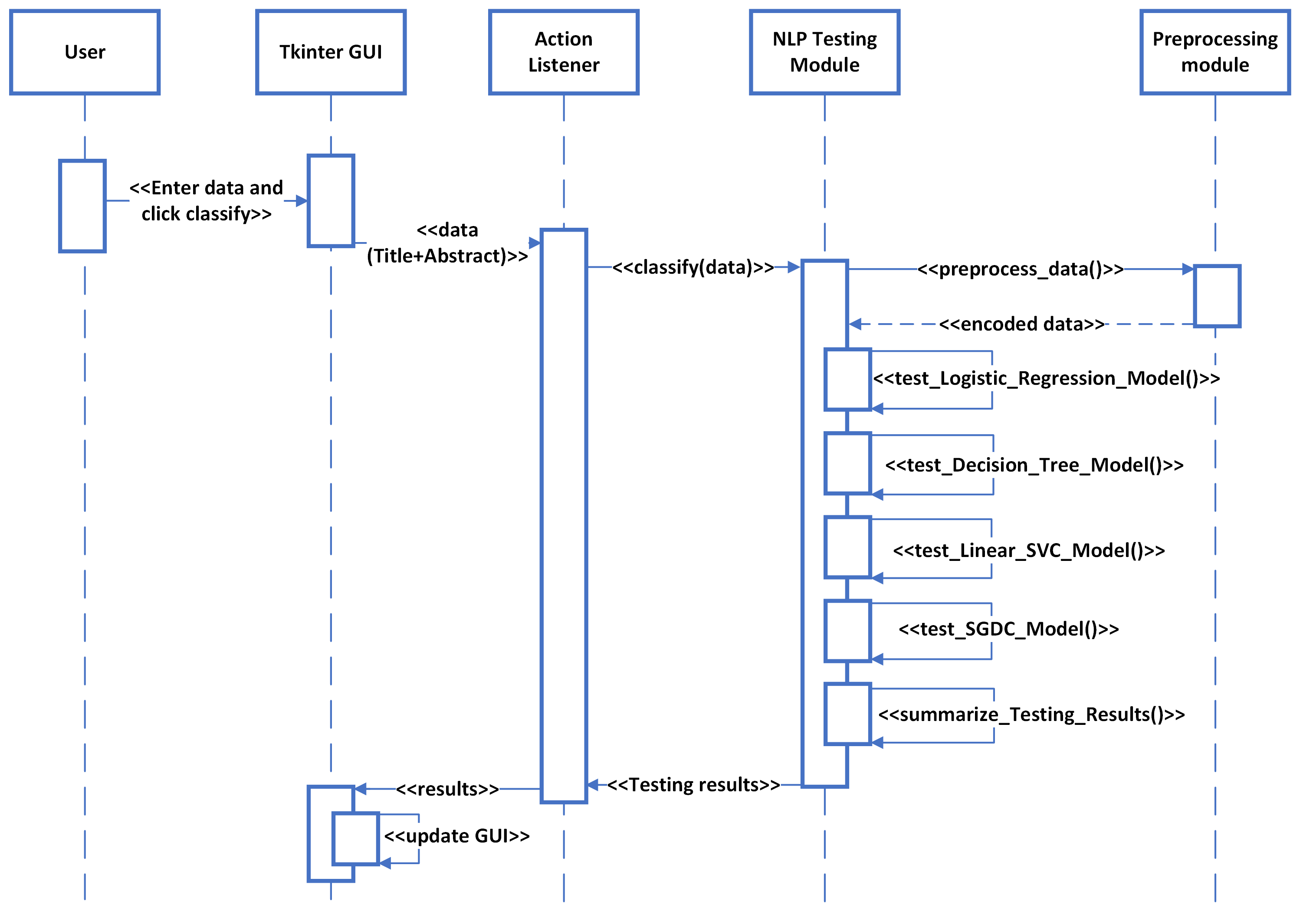


Figure 9: Sequence Diagram for Use Case 8 and Use Case 9.

1. **Architecture Design Diagram**

The architecture diagram is shown in Figure 10 below.

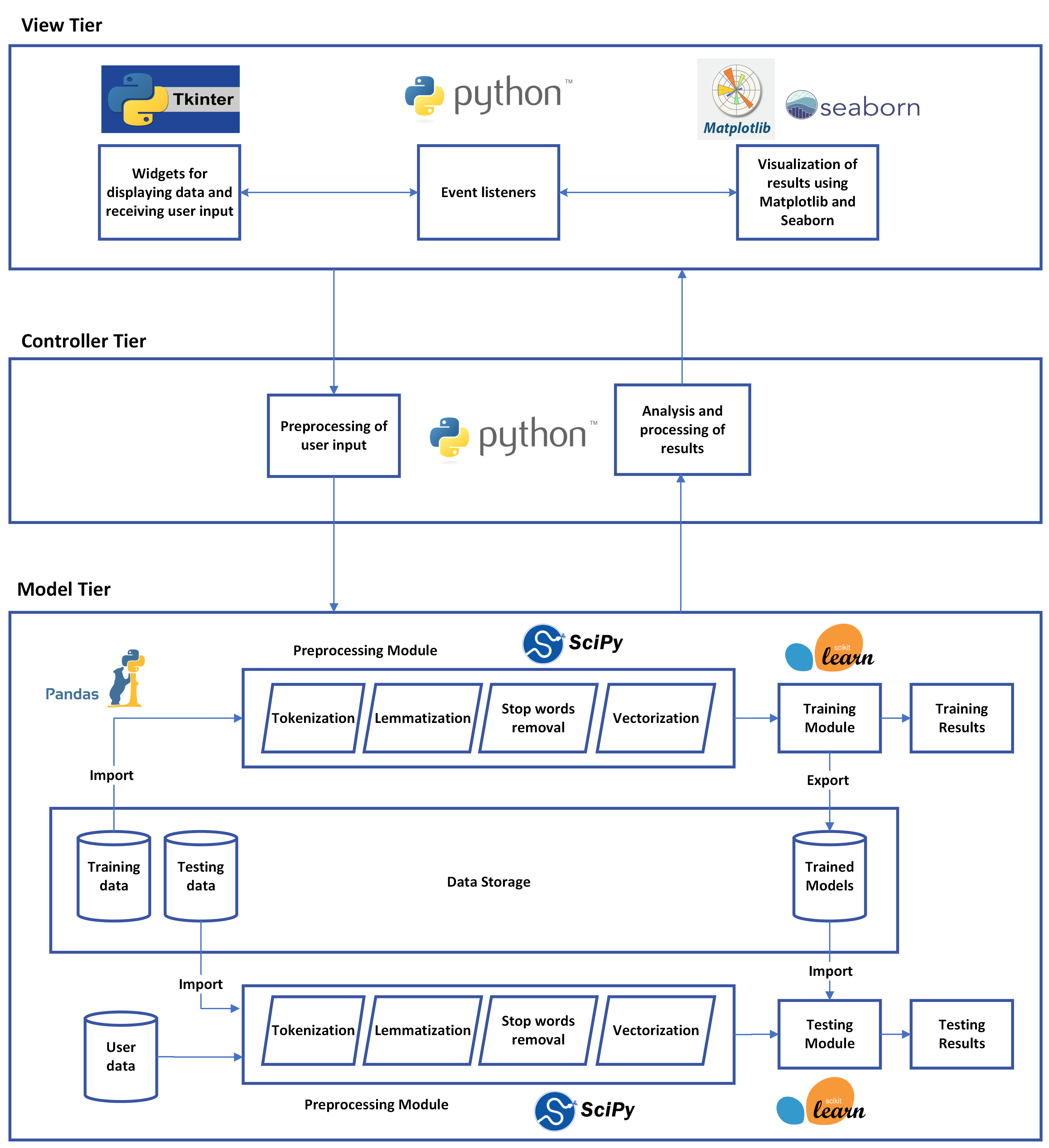


Figure 10: Architecture Design Diagram.

The architecture is divided into three tiers:

1. Model tier: The NLP system model is used to train and test machine learning models for classification of research articles. It imports and preprocesses train data for training machine learning models. It also tests machine learning models using test/ input data and outputs classification results. Sci-kit learn library will be used for the training and testing of machine learning models. The Pandas library will be used for importing/ exporting files from/ to the storage system.
2. View tier: The user interface contains Tkinter widgets to read user input and display results of NLP models. It also contains event listeners for reacting to user and system events. Tkinter will be used for building the desktop application GUI. Matplotlib will be used for visualization of classification results.
3. Controller tier: The controller is used to interface the user interface with the NLP model. It preprocessed user input data and classification results so that they can be properly formatted for the Tkinter GUI and machine learning model.
4. **Interface Design**

This section shows the user interface design for the complete Natural Language Processing system. The user interface has nine pages to implement nine use cases specified in the Software Requirements Specification (SRS) document.

The user interface for Exploratory Data Analysis (EDA) page is shown in Figure 10 below. It will be used to display results of preprocessing and EDA for train and test data.

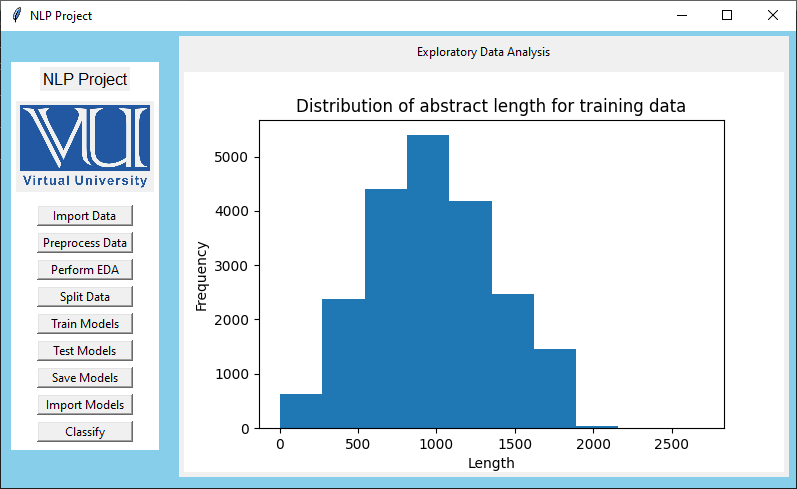


Figure 10: User interface diagram for Exploratory Data Analysis page.

The user interface for NLP models training page is shown in Figure 11 below. It will be used to train machine learning models and display results of training.

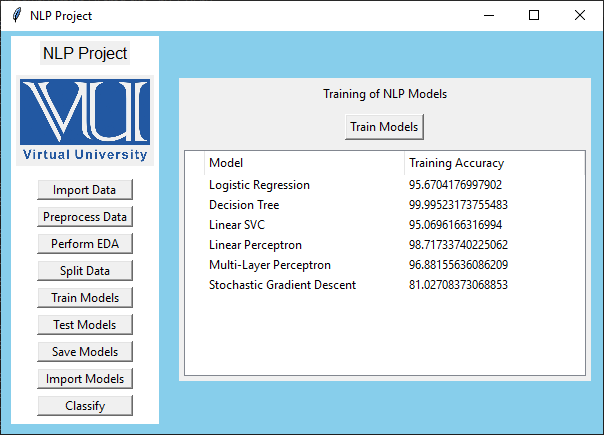


Figure 11: User interface diagram of models training page.

The user interface for research article classification page is shown in Figure 12 below. It will be used to enter research article title and data for classification of research article. It will also display the results of classification.

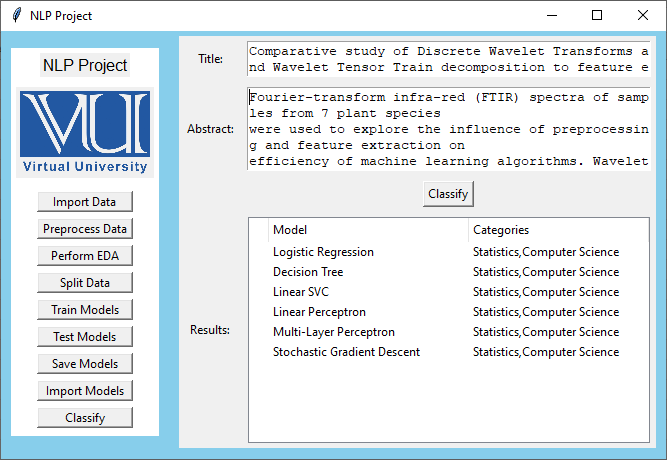


Figure 12: User interface diagram for article classification page.

1. **Test Cases**

This section presents the test cases for testing each functionality specified in the use case diagram.

1. Test case 1A: Test the import dataset functionality

|  |  |  |  |
| --- | --- | --- | --- |
| Test case ID | | TC-1A | |
| Test case title | | Test the import dataset functionality | |
| Use case | | Use Case 1: Import dataset | |
| Description | | Test the dataset import functionality of file I/O module by importing train and test data | |
| Preconditions | | The Tkinter GUI must be initialized and import dataset page must be operational. | |
| Step No. | Test step | Expected result | Status |
| 1 | Navigate to import data page | The import page must appear with import data button enabled. |  |
| 2 | Click import dataset button | The data must be imported and saved successfully. Two separate data objects must store all values contained in train and test files. The UI must be updated to show that import was successful. |  |

1. Test case 2A: Test word tokenization functionality

|  |  |  |  |
| --- | --- | --- | --- |
| Test case ID | | TC-2A | |
| Test case title | | Test word tokenization functionality | |
| Use case | | Use Case 2: Perform EDA and preprocessing | |
| Description | | Test the word tokenization functionality of data preprocessing module by analyzing train and test data. | |
| Preconditions | | The train and test data must be imported successfully. The Tkinter GUI must be initialized and preprocess data page must be operational. | |
| Step No. | Test step | Expected result | Status |
| 1 | Navigate to preprocess data page | The preprocessing page must appear with preprocess data button enabled. |  |
| 2 | Click preprocess button | The abstract and title data must be broken down into separate words. The GUI must be updated to show the results of word tokenization. |  |

1. Test case 2B: Test lemmatization functionality

|  |  |  |  |
| --- | --- | --- | --- |
| Test case ID | | TC-2B | |
| Test case title | | Test lemmatization functionality | |
| Use case | | Use Case 2: Perform EDA and preprocessing | |
| Description | | Test the word lemmatization functionality of data preprocessing module by processing tokenized train and test data | |
| Preconditions | | The word tokenization of train and test data must be completed successfully. The Tkinter GUI must be initialized and preprocess data page must be operational | |
| Step No. | Test step | Expected result | Status |
| 1 | Navigate to preprocess data page | The preprocessing page must appear with preprocess data button enabled. |  |
| 2 | Click preprocess button | After tokenization, words must be grouped based on inflected forms. The GUI must be updated to show the results of lemmatization. |  |

1. Test case 2C: Test stop words removal functionality

|  |  |  |  |
| --- | --- | --- | --- |
| Test case ID | | TC-2C | |
| Test case title | | Test stop words removal functionality | |
| Use case | | Use Case 2: Perform EDA and preprocessing | |
| Description | | Test the stop words removal functionality of data preprocessing module by processing tokenized train and test data | |
| Preconditions | | The lemmatization of train and test data must be completed successfully. The Tkinter GUI must be initialized and preprocess data page must be operational | |
| Step No. | Test step | Expected result | Status |
| 1 | Navigate to preprocess data page | The preprocessing page must appear with preprocess data button enabled. |  |
| 2 | Click preprocess button | After lemmatization, stop words must be removed from bag of words. The GUI must be updated to show the results of stop words removal. |  |

1. Test case 2D: Test vectorization functionality

|  |  |  |  |
| --- | --- | --- | --- |
| Test case ID | | TC-2D | |
| Test case title | | Test vectorization functionality | |
| Use case | | Use Case 2: Perform EDA and preprocessing | |
| Description | | Test the vectorization functionality of data preprocessing module by processing tokenized train and test data | |
| Preconditions | | The stop words removal of train and test data must be completed successfully. The Tkinter GUI must be initialized and preprocess data page must be operational | |
| Step No. | Test step | Expected result | Status |
| 1 | Navigate to preprocess data page | The preprocessing page must appear with preprocess data button enabled. |  |
| 2 | Click preprocess button | After stop words removal, words will be vectorized using TF-IDF vectorizer. The GUI must be updated to show the results of vectorization. |  |

1. Test case 2E: Test Exploratory Data Analysis functionality

|  |  |  |  |
| --- | --- | --- | --- |
| Test case ID | | TC-2E | |
| Test case title | | Test Exploratory Data Analysis functionality | |
| Use case | | Use Case 2: Perform EDA and preprocessing | |
| Description | | Test the Exploratory Data Analysis functionality of data preprocessing module by processing train and test data. | |
| Preconditions | | The preprocessing of train and test data must be completed successfully. The Tkinter GUI must be initialized and preprocess data page must be operational | |
| Step No. | Test step | Expected result | Status |
| 1 | Navigate to preprocess data page | The preprocessing page must appear with preprocess data button enabled. |  |
| 2 | Click perform EDA button | The summary of EDA results will be displayed. The GUI must be updated to show the results of EDA. |  |

1. Test case 3A: Test splitting of train and test datasets

|  |  |  |  |
| --- | --- | --- | --- |
| Test case ID | | TC-3A | |
| Test case title | | Test splitting of train and test datasets | |
| Use case | | Use Case 3: Split dataset into test and train | |
| Description | | Test the functionality of training module to split data into train and test subsets. | |
| Preconditions | | The preprocessing of train and test data must be completed. The Tkinter GUI must be initialized and train models page must be operational. | |
| Step No. | Test step | Expected result | Status |
| 1 | Navigate to train models page | The train models page must appear with split data button enabled. |  |
| 2 | Click split data button | The data must be split into train and test subsets successfully. The GUI must be updated to show the results of data splitting. |  |

1. Test case 4A: Test training of NLP models

|  |  |  |  |
| --- | --- | --- | --- |
| Test case ID | | TC-4A | |
| Test case title | | Test training of NLP models | |
| Use case | | Use Case 4: Train models | |
| Description | | Test the functionality of training module to test four machine learning models. | |
| Preconditions | | The preprocessing and splitting of train and test data must be completed. The Tkinter GUI must be initialized and train models page must be operational. | |
| Step No. | Test step | Expected result | Status |
| 1 | Navigate to train models page | The train models page must appear with train models button enabled. |  |
| 2 | Click train models button | The NLP models must be trained successfully. The GUI must be updated to show the results of the training. |  |

1. Test case 5A: Test the export of trained models

|  |  |  |  |
| --- | --- | --- | --- |
| Test case ID | | TC-5A | |
| Test case title | | Test export of trained NLP models | |
| Use case | | Use Case 5: Save trained models | |
| Description | | Test the functionality of training module to export trained NLP models. | |
| Preconditions | | The training of all NLP models must be completed successfully. The Tkinter GUI must be initialized and the train models page must be operational. | |
| Step No. | Test step | Expected result | Status |
| 1 | Navigate to train models page | The train models page must appear with save models button enabled. |  |
| 2 | Click save models button | The NLP models must be saved successfully. The GUI must be updated to show the results of the training. |  |

1. Test case 6A: Test trained models using test dataset

|  |  |  |  |
| --- | --- | --- | --- |
| Test case ID | | TC-6A | |
| Test case title | | Test trained models using test dataset | |
| Use case | | Use Case 6: Test models | |
| Description | | Test the functionality of testing module to test trained NLP models using imported test dataset. | |
| Preconditions | | The training of all NLP models must be completed successfully and the test dataset must be available after preprocessing. The Tkinter GUI must be initialized and the test models page must be operational. | |
| Step No. | Test step | Expected result | Status |
| 1 | Navigate to test models page | The test models page must appear with test models button enabled. |  |
| 2 | Click test models button | The NLP models must be tested successfully using imported test dataset. The GUI must be updated to show the results of the testing. |  |

1. Test case 7A: Test import of trained models

|  |  |  |  |
| --- | --- | --- | --- |
| Test case ID | | TC-7A | |
| Test case title | | Test import of trained models | |
| Use case | | Use Case 7: Import trained models | |
| Description | | Test the functionality of importing trained NLP models file I/O module. | |
| Preconditions | | The trained NLP models must be saved successfully. The Tkinter GUI must be initialized successfully and the test models page must be operational. | |
| Step No. | Test step | Expected result | Status |
| 1 | Navigate to test models page | The test models page must appear with import models button enabled. |  |
| 2 | Click import models button | The NLP models must be imported successfully. The GUI must be updated to show the results of the import. |  |

1. Test case 8A: Test the user input reading functionality

|  |  |  |  |
| --- | --- | --- | --- |
| Test case ID | | TC-8A | |
| Test case title | | Test the user input reading functionality | |
| Use case | | Use Case 8: Enter title and abstract of research article | |
| Description | | Test the functionality of reading user input data for research article title and abstract using Tkinter action listeners. | |
| Preconditions | | The trained NLP models must be loaded successfully. The Tkinter GUI must be initialized successfully and the test models page must be operational. | |
| Step No. | Test step | Expected result | Status |
| 1 | Navigate to test models page | The test models page must appear with text fields and classify button enabled. |  |
| 2 | Enter tile and abstract. Then, click classify button | The input text for title and abstract must be passed successfully by the action handler. The GUI must be updated to show that data was read successfully. |  |

1. Test case 9A: Test the classification of new research article

|  |  |  |  |
| --- | --- | --- | --- |
| Test case ID | | TC-9A | |
| Test case title | | Test the classification of new research article | |
| Use case | | Use Case 9: Test model to categorize new research article | |
| Description | | Test the functionality of classifying a new research article based on title and abstract data provided by user. | |
| Preconditions | | The trained NLP models must be loaded successfully. The user input must be read successfully by the system. The Tkinter GUI must be initialized successfully and the test models page must be operational. | |
| Step No. | Test step | Expected result | Status |
| 1 | Navigate to test models page | The test models page must appear with text fields and classify button enabled. |  |
| 2 | Enter tile and abstract. Then, click classify button | The research article must be classified successfully by the trained NLP models. The GUI must be updated to show the results of classification. |  |