NLP based Duplicate Bug Report Detection using Supervised Machine Learning Algorithms

**Software Requirements Specification**

Version 1.0



**Group Id: F22024F062 (BC190400681)**

**Supervisor Name: Saad Ahmed**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date (dd/mm/yyyy)** | **Version** | **Description** | **Author** |
| 24/12/2022 | 1.0 | In this project, we aim to classify bug reports using machine learning models and natural language processing techniques. The project involves preprocessing the textual data in the bug reports using techniques such as tokenization, stop word removal, and lemmatization.  The processed data is then used to train and test machine learning algorithms, such as Naive Bayes, Support Vector Machines, and Random Forests, to classify the bug reports. The goal is to effectively triage and fix bugs by clearly identifying important features in the bug reports. | BC190400681 |

**Table of Contents**

1. [Scope (of the project) ----------------------------------------------------- Page No# 4](#Scope)
2. [Functional Requirements ------------------------------------------------- Page No# 4](#FR)
3. [Non-Functional Requirements (Optional) ----------------------------- Page No# 4](#NFR)
4. [Use Case Diagram (Optional) ------------------------------------------- Page No# 5](#UseCase)
5. [Usage Scenarios (Optional) ---------------------------------------------- Page No# 6](#UsageScenarios)
6. [Adopted Methodology ---------------------------------------------------- Page No# 10](#Adopted)
7. [Work Plan ------------------------------------------------------------------](#WorkPlan)[Page No# 13](#WorkPlan)

**SRS Document**

**Scope of Project:**

The project aims to classify duplicate bug reports using natural language processing (NLP) techniques to improve bug triaging and fixing efficiency.

The main tasks include setting up the system environment, preprocessing bug report data, building a corpus, splitting data into training and testing sets, training machine learning models (Naive Bayes, Support Vector Machine, and Random Forest), evaluating the models using metrics such as confusion matrix, accuracy, precision, and recall, and discussing and comparing the results of the algorithms. The project may involve retraining the model to improve the accuracy.

**Functional Requirements:**

The following are the functional requirements of the project separated by system and user requirements:

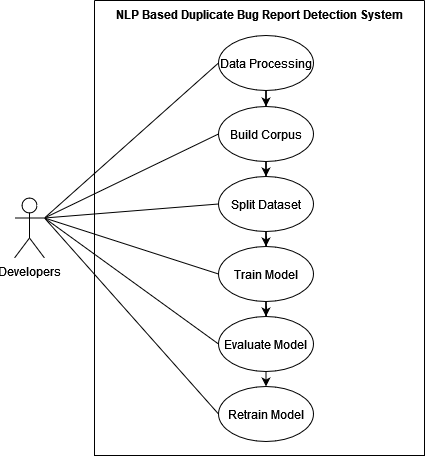
1. System must be set the environment online/offline (If required)
2. System applies different data processing techniques (Tokenization, Stop word removal, Lemmatization, etc.)
3. System must Build Corpus
4. System must be split the given dataset into testing and training.
5. System must train the specified model.
6. User must be evaluating mentioned models in the form of Confusion Matrix, Accuracy, Precision, Recall
7. User must be discussed the results of given algorithms (Naïve Bayes, Support Vector Machine, Random Forest)
8. User must retrain the model if accuracy is not good (less than 60%) by changing different training parameters (If Required)

**Non-Functional (Optional) Requirements:**

Non-functional requirements for this project/system may include:

* **Performance:** The system must be able to classify a bug report within a specific time frame, for example, within a few seconds, to ensure that the bug triaging process is not delayed.
* **Scalability:** The system must be able to handle an increasing number of bug reports without a significant decrease in performance.
* **Reliability:** The system must have a high level of reliability, with minimal downtime, to ensure that bug reports are classified and assigned to the appropriate developer in a timely manner.
* **Usability:** The system must be user-friendly, with an intuitive interface, to ensure that developers and customer support representatives are able to use it effectively.

**Use Case Diagram (Optional):**



**Usage Scenarios (Optional):**

Here are the usage scenarios of the use cases for this project:

**Usage Scenario#1:**

|  |  |
| --- | --- |
| Use Case Title | Data Processing |
| Use Case ID | UC-01 |
| Action(s) | 1. The user launches the system. 2. The user selects the option to apply data processing techniques. 3. The user chooses the techniques to apply such as Tokenization, Stop word removal, Lemmatization, etc. 4. The system applies the selected techniques on the data. |
| Description | The user selects the NLP techniques to be applied on the bug report data. These techniques help in preparing the data for further processing, such as building a corpus and training machine learning models. |
| Alternative Paths | None |
| Pre and Post Conditions | Pre-Conditions: The system environment must be configured. |
| Post-Conditions: The data has been processed using the selected NLP techniques. |
| Author(s) | User |
| Exceptions | The system may not work correctly if there is an error in the implementation of the techniques. |

**Usage Scenario#2:**

|  |  |
| --- | --- |
| Use Case Title | Build Corpus |
| Use Case ID | UC-02 |
| Action(s) | 1. The user launches the system. 2. The user selects the option to build the corpus. 3. The system builds the corpus using the processed data. |
| Description | The system builds a corpus using the processed data to train machine learning models. |
| Alternative Paths | None |
| Pre and Post Conditions | Pre-Conditions: The data must be processed using NLP techniques. |
| Post-Conditions: The corpus has been built and is ready to use for training machine learning models. |
| Author(s) | User |
| Exceptions | The system may not work correctly if there is an error in the implementation of building the corpus or if the data is not in the correct format. |

**Usage Scenario#3:**

|  |  |
| --- | --- |
| Use Case Title | Data Splitting |
| Use Case ID | UC-03 |
| Action(s) | 1. The user launches the system. 2. The user selects the option to split the data. 3. The user inputs the percentage of data to be used for testing. 4. The system splits the data into training and testing sets. |
| Description | The user splits the data into training and testing sets. The training data is used to train the machine learning models, while the testing data is used to evaluate the performance of the models. |
| Alternative Paths | None |
| Pre and Post Conditions | Pre-Conditions: The corpus must be built. |
| Post-Conditions: The data has been split into training and testing sets. |
| Author(s) | User |
| Exceptions | The system may not work correctly if the percentage of data for testing is set to a value that is too low or too high. |

**Usage Scenario#4:**

|  |  |
| --- | --- |
| Use Case Title | Model Training |
| Use Case ID | UC-04 |
| Action(s) | 1. User selects the specified model to be trained. 2. User sets the training parameters. 3. System starts the training process. 4. System saves the trained model. |
| Description | After splitting the dataset into testing and training sets, the user selects the model to be trained from the available options. The user sets the training parameters and initiates the training process. The system uses the training data to train the selected model and save the trained model for future use. |
| Alternative Paths | None |
| Pre and Post Conditions | Pre-Conditions: The dataset must be split into training and testing sets. |
| Post-Conditions: The system must save the trained model. |
| Author(s) | User |
| Exceptions | None |

**Usage Scenario#5:**

|  |  |
| --- | --- |
| Use Case Title | Model Evaluation |
| Use Case ID | UC-05 |
| Action(s) | 1. The user selects the algorithm they want to evaluate and compare the results for. 2. The system presents the evaluation results for the selected algorithm, including confusion matrix, accuracy, precision, and recall. 3. The user discusses the results and compares them with the results of other algorithms. 4. The user decides on the algorithm to use for the bug report classification. |
| Description | This use case represents the step where the user evaluates the performance of different algorithms and compares their results to make a decision on which algorithm to use for the classification of duplicate bug reports. |
| Alternative Paths | None |
| Pre and Post Conditions | Pre-Conditions: The system must have the evaluation results for the algorithms. The user must have knowledge of the performance metrics. |
| Post-Conditions: The results of the evaluation are documented for future reference. |
| Author(s) | User |
| Exceptions | None |

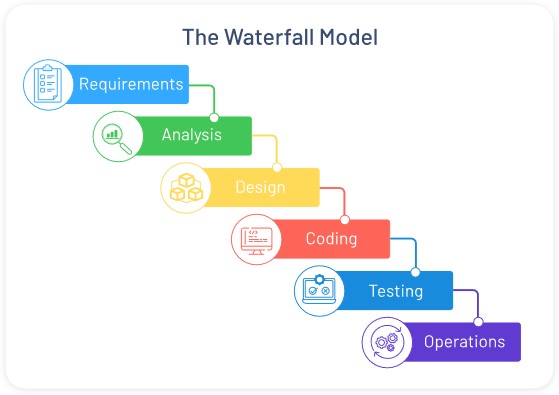
**Usage Scenario#6:**

|  |  |
| --- | --- |
| Use Case Title | Retrain Model |
| Use Case ID | UC-06 |
| Action(s) | 1. The user checks the accuracy of the current model and finds it to be less than 60%. 2. The user selects the option to retrain the model. 3. The system suggests different training parameters to adjust. 4. The user selects the training parameters they want to adjust. 5. The system re-trains the model with the updated training parameters. 6. The system evaluates the new model. 7. The user checks the accuracy of the new model and finds it to be satisfactory. |
| Description | This use case represents the step where the user re-trains the model to improve accuracy if the accuracy of the current model is less than 60%. |
| Alternative Paths | 1. If the user does not want to retrain the model, they can choose to use the existing model with less accuracy. 2. If the adjusted training parameters do not improve accuracy, the user can adjust other parameters or try a different algorithm. |
| Pre and Post Conditions | Pre-Conditions: The user must check the accuracy of the current model and find it to be less than 60%. |
| Post-Conditions: The system has a new model with improved accuracy. The user can use the new model for bug report classification. |
| Author(s) | User |
| Exceptions | If the accuracy of the model is greater than or equal to 60%, no retraining is required, and the system will continue with the next step(s). |

**Adopted Methodology:**

I’ve adopted the VU Process Model which is a combination of the Waterfall and Spiral models. It can be used to manage the development of this project. This methodology is chosen because it allows for the integration of the best practices of both models, resulting in a more efficient and effective development process.

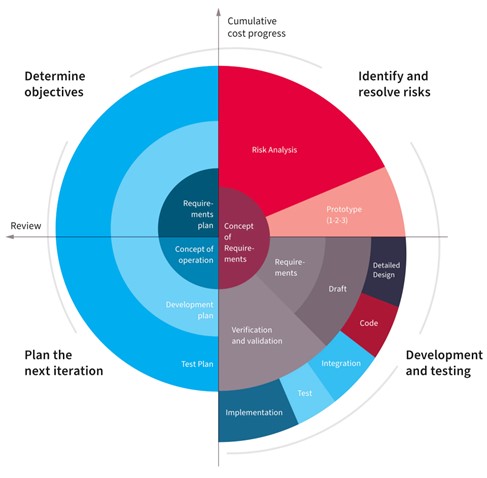
**Waterfall Model:**



The Waterfall model is known for its linear and sequential approach, which is ideal for projects that have well-defined and fixed requirements. This model is used in the initial stages of the project, during the requirements gathering and analysis phase. This phase is critical in this project as it is important to understand the specific requirements of the project such as the type of dataset and the machine learning algorithm to be used, so that the project can be defined clearly and accurately.

In the Waterfall model, the requirements gathering, and analysis phase is followed by the design phase where the system architecture and design are developed based on the requirements gathered. This phase is also critical for this project as it involves deciding on the technical design, including the NLP techniques, machine learning algorithms, and data preprocessing methods that will be used.

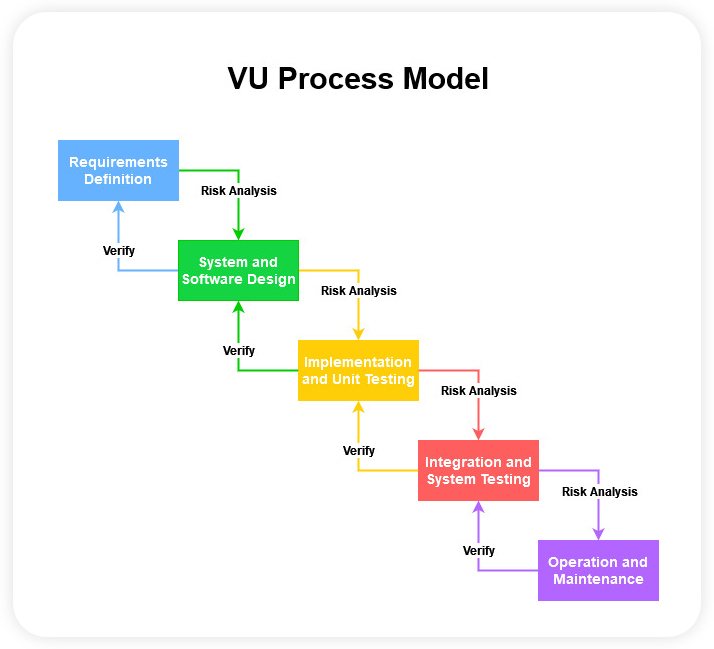
**Spiral Model:**



The Spiral model is known for its iterative and incremental approach, which is ideal for projects that have changing and evolving requirements. This model is used in the later stages of the project, during the design, development, and testing phases. In this project, the spiral model will be used to develop, test, and improve the system by using different machine learning algorithms and parameters.

This approach allows for the system to be developed and tested incrementally, and for any issues or bugs to be identified and fixed early in the development process. Additionally, it allows for the system to be improved and optimized by experimenting with different machine learning algorithms and parameters, and for changes to be made based on the results obtained from the testing phase. This approach will ensure that the system is developed to the highest standard and meets the requirements of the project.

**VU Process Model:**



The VU process model combines these two models to provide an efficient development process. In this project, the Waterfall model will be used in the initial stages to define the project requirements, and the Spiral model will be used in the later stages to develop and test the system. This approach allows for a more flexible development process and allows for changes and improvements to be made throughout the project.

Overall, the VU Process Model allows for a structured and iterative approach to the development of this NLP project, ensuring that all requirements are properly gathered and analyzed, the design is well thought out and implemented, the system is thoroughly tested and verified, and it is well maintained over time.

**Work Plan:**

