

A Major Project Report  
On  
**“Nature-Based Prediction Model of Bug Reports Based  
on Ensemble Machine Learning Model”**

Submitted to the CMR Institute of Technology in partial fulfillment of the Academic  
Requirement for the Award of Degree

**BACHELOR OF TECHNOLOGY**

In

Computer Science and Engineering (Artificial Intelligence and Machine learning)

Submitted By:

B. Sai Charan	(21R01A6678)
K. Ashwini	(21R01A6689)
KB. Aakash	(21R01A6690)

Under the esteemed guidance of

Dr. L. Arokia Jesu Prabhu  
(Assoc.Prof, CSE(AI&ML))



**CMR INSTITUTE OF TECHNOLOGY**

**(UGC AUTONOMOUS)**

Approved by AICTE, Affiliated to JNTUH, Accredited by NAAC with A+ Grade,

Kandlakoya(V), Medchal Dist - 501401

[www.cmrithyderabad.edu.in](http://www.cmrithyderabad.edu.in)

**2024-25**

# CMR INSTITUTE OF TECHNOLOGY

(UGC AUTONOMOUS)

Approved by AICTE, Affiliated to JNTUH, Accredited by NAAC with A+ Grade

Kandlakoya (V), Medchal Dist - 501401

[www.cmrityderabad.edu.in](http://www.cmrityderabad.edu.in)



## CERTIFICATE

This is to certify that an Industry oriented Major Project entitled with “Nature-Based Prediction Model of Bug Reports Based on Ensemble Machine Learning Model” is being submitted by:

B. Sai Charan	(21R01A6678)
K. Ashwini	(21R01A6689)
KB. Aakash	(21R01A6690)

To JNTUH, Hyderabad, in partial fulfillment of the requirement for award of the degree of B. Tech in CSE (AI&ML) and is a record of a bonafide work carried out under our guidance and supervision. The results in this project have been verified and are found to be satisfactory. The results embodied in this work have not been submitted to have any other University for award of any other degree or diploma.

Signature of Guide  
Dr. L. Arokia Jesu Prabhu

Signature of Project Coordinator  
Dr. L. Arokia Jesu Prabhu

Signature of HOD  
Prof. P. Pavan Kumar

EXTERNAL EXAMINER

## **ACKNOWLEDGEMENT**

We are extremely grateful to Dr. M Janga Reddy, Director, Dr. G. Madhusudhan Rao, Principal and Prof. P. Pavan Kumar, Head of Department, Dept of Computer Science and Engineering (Artificial Intelligence and Machine Learning), CMR Institute of Technology for their inspiration and valuable guidance during entire duration.

We are extremely thankful to, Dr. L. Arokia Jesu Prabhu, Project Coordinator and internal guide, Dept of Computer Science and Engineering (Artificial Intelligence and Machine Learning), CMR Institute of Technology for their constant guidance, encouragement and moral support throughout the project.

We will be failing in duty if we do not acknowledge with grateful thanks to the authors of the references and other literatures referred in this Project.

We express our thanks to all staff members and friends for all the help and coordination extended in bringing out this Project successfully in time.

Finally, we are very much thankful to our parents and relatives who guided directly or indirectly for every towards success.

B. SAI CHARAN	21R01A6678
K. ASHWINI	21R01A6689
KB. AAKASH	21R01A6690

## **ABSTRACT**

Software system maintenance in software engineering is required to fix the defects that are found during testing. Bug reports (BRs) play a significant role in this regard and contain critical information like descriptions, severity, and priority. These reports are manually analyzed, which is time-consuming and inefficient due to the fact that the number of BRs is growing. Research so far has taken into account automating detection of severity and priority but not classifying bugs according to their nature. This project proposes a new ensemble machine learning and natural language processing (NLP) Nature-Based Bug Prediction Model to predict bug reports automatically. The model is trained on Mozilla and Eclipse datasets and predicts bugs into six categories: Program Anomaly, GUI, Network/Security, Configuration, Performance, and Test-Code. Text augmentation techniques are employed for implementation to enhance prediction accuracy. The results validate that the model discussed achieves 90.42% accuracy without the use of text augmentation and 96.72% accuracy using text augmentation, which is an improvement compared to other models in existence. The research assists in improving software maintenance effectiveness through improved accuracy and an automated bug classifying system.

## **INDEX**

<b>ACKNOWLEDGEMENT</b>	<b>i</b>
<b>ABSTRACT</b>	<b>ii</b>
<b>INDEX</b>	<b>iii</b>
<b>LIST OF FIGURES</b>	<b>iv</b>
<b>LIST OF TABLES</b>	<b>v</b>
<b>1.INTRODUCTION</b>	<b>1</b>
1.1 ABOUT PROJECT	1
1.2 EXISTING SYSTEM	2
1.3 PROPOSED SYSTEM	3
<b>2. LITERATURE SURVEY</b>	<b>5</b>
<b>3. SYSTEM DESIGN</b>	<b>10</b>
3.1 ARCHITECTURE / BLOCK DIAGRAM	10
3.2 DATA FLOW DIAGRAM	11
3.3 UML DIAGRAM	12
<b>4. IMPLEMENTATION</b>	<b>16</b>
4.1 PROJECT MODULES	16
4.2 ALGORITHMS	17
<b>5. TESTING</b>	<b>21</b>
5.1 TESTING METHODS	21
5.2 USER TRAINING	24
5.3 MAINTAINENCE	24
<b>6. RESULTS</b>	<b>26</b>
<b>7. CONCLUSION</b>	<b>28</b>
<b>8. REFERENCES</b>	<b>29</b>
<b>APPENDIX I - SCREENSHOTS</b>	<b>32</b>
<b>APPENDIX II – SAMPLE CODE</b>	<b>37</b>

## LIST OF FIGURES

Figure No.	Particulars	Page No.
3.1.1	Architecture Diagram	10
3.2.1	Data Flow Diagram	11
3.3.1	Class Diagram	13
3.3.2	Sequence Diagram	14
3.3.3	Use Case Diagram	15
6.1	Model vs Accuracy Graph	27

## LIST OF TABLES

Table No.	Particulars	Page No.
6.1	Model Type and Accuracy	26

# **1. INTRODUCTION**

## **1.1 ABOUT PROJECT**

In modern software development, bug reports (BRs) are precious resources for detection and repair of software faults. Bug reports contain essential information such as bug descriptions, severity, priority, and responsible developers and are a critical component of the software maintenance process. But as software systems become more complicated, bug reports increase exponentially, and their manual analysis is time-consuming and inefficient.

Traditional approaches in bug classification rely on determining the priority or severity of bugs through machine learning and natural language processing (NLP). Most of the present approaches, however, fail to consider the type of bug, which is important for debugging as well as software maintenance. Classification of the type of a bug involves multi-class classification, and it has been difficult due to the diversity and complexity of bug reports.

This paper introduces a Nature-Based Bug Prediction Model that employs ensemble machine learning techniques to classify bug reports automatically based on their nature. The method integrates NLP, text mining, and machine learning techniques to analyze bug reports effectively. Employing text augmentation and an ensemble of classifiers, the model improves the efficiency of classification accuracy and debugging process.

For evaluating the model's performance, we use a publicly accessible dataset of Mozilla and Eclipse software repositories' bug reports. The model labels bug reports into six categories: Program Anomaly, GUI, Network/Security, Configuration, Performance, and Test-Code. Experimental results demonstrate that the proposed ensemble learning approach is highly accurate in comparison to previous classification models.



### 1.2 EXISING SYSTEM

Kukkar and Mohana [10] combine text mining, natural language processing, and machine learning techniques to classify bug and non-bug reports to solve the problem of misclassification of bug reports, which negatively affects the overall performance of the prediction process. This model uses bigram and TF-IDF in feature selection. However, the performance of the KNN algorithm in this model changes when the data-set changes.

Some researchers have different classifications for bug reports. In [11], researchers use machine learning algorithms to classify new bug reports as either corrective (defect fixing) reports or perfective (major maintenance) reports by using support vector machines (SVM), naive Bayes (NB), and random trees (RT). The results show that SVM achieved the highest accuracy, 93.1%.

To support the software engineering process, researchers have introduced a well-known schema called Orthogonal Defect Classification (ODC) [12]. ODC has eight orthogonal attributes to characterize software defects and several analytical methods for test process analysis and software development [51]. Extracting valuable information from defects can be done by ODC, which provides insights and helps diagnosis in software engineering processes [13]. The authors in [13] use machine learning algorithms to classify bug reports according to ODC. Their study uses 4096 ODC annotated bug reports. However, they conclude that there is difficulty in automating the ODC attributes using only bug reports.

Hirsch and Hofer [14] classified each new bug according to a bug report using three classes, concurrency, memory, and semantic bugs. They used 369 bug reports, and the highest mean precision and recall were 0.74 and 0.72, respectively.

### Disadvantages

- An existing system is not hybrid deep learning detection policy to improve the efficiency and effectiveness of Bug Report Generation.
- An existing system never used Nature-Based Ensemble Machine Learning Bug Prediction Model which is more accurate and efficient.

### **1.3 PROPOSED SYSTEM**

The Nature-Based Bug Prediction Model introduced here introduces a novel approach to bug report classification based on machine learning (ML), natural language processing (NLP), and text augmentation techniques. The research compares different base machine learning algorithms to evaluate their applicability for automated bug classification and then proposes an ensemble learning-based approach for improving predictive accuracy. Through text augmentation, feature extraction from the model is enhanced to make way for an improved classification of bug reports as per their type. For assessing the effectiveness of its method, the model is evaluated on a benchmark test set with bug reports collected from Mozilla and Eclipse. The experimentation outcome indicates how ensemble learning methodology significantly boosts accuracy, with comparison to a majority of standard models. This is important since the existing models have a tendency to focus solely on bug priority or severity classification rather than the type of the bug.

One of the greatest strengths of this model is its ability to use multiple ML classifiers in an attempt to improve prediction accuracy.

By being trained on a benchmark dataset, the model ensures consistent performance with diverse bug reports. The inclusion of NLP and text-mining technologies also enhances its ability to extract pertinent information from unstructured bug reports. The new algorithm categorizes 504 bug reports into four broad categories: assignment/initialization, external interface, internal interface, and other. Depending on the categorization, the model achieves a high accuracy of 73.70%, indicating that it is a valuable resource for automated software debugging.

#### **Advantages:**

- **Enhanced Precision:** Ensemble learning method and text augmentation increase classification precision.
- **scalability:** The model can be applied to large bug databases with fewer manual interactions.
- **Bug Analysis Automation:** Reduces manual work involved in bug categorization, thereby software maintenance more effective and economical.
- **First-of-Its-Kind Methodology:** This research is the first known study to integrate ensemble machine learning with text augmentation for bug nature prediction.

## **Nature-Based Prediction Model of Bug Reports Based on Ensemble Machine Learning Model**

By addressing the limitations of traditional bug categorization methods and introducing a better and automated way, this research assists enhancing software maintenance practices such that developers can identify, categorize, and correct bugs faster and more accurately.

## 2. LITERATURE SURVEY

### 2.1 Predicts each bug report as (Configuration) or (Non-configuration) bug reports and if it is configurable, outputs the associated configuration names

**Authors:** Wei Wen, Tingting Yu, Jane Huffman Hayes

The study presented at the 2016 IEEE 27th International Symposium by Wei Wen, Tingting Yu, and Jane Huffman Hayes introduces an approach for predicting and classifying bug reports as either Configuration or Non-Configuration related issues. Additionally, if a bug report is categorized as Configuration-related, the model identifies the associated configuration names that are responsible for the bug. The following tools and libraries were used in the implementation are Weka, NLTK (Natural Language Toolkit), Sklearn (Scikit-learn)

The accuracy of predicting configuration-related bugs varied across different repositories:

- Mozilla: 92% (0.92) accuracy in correctly classifying configuration bug reports.
- Apache: 88% (0.88) accuracy in identifying configuration issues.
- MySQL: 74% (0.74) accuracy, indicating relatively lower performance, possibly due to the complexity of MySQL's configuration issues.

The model's accuracy for MySQL (0.74) suggests that further improvements are needed, such as enhancing feature extraction techniques or using more advanced deep learning methods. The study focuses on historical bug reports, so adapting it for real-time bug classification could further enhance its usability in software development.

### 2.2 Predicts each bug report to be either a Corrective (defect fixing) report or a Perfective (major maintenance) report

**Authors:** Otoom, A. F., Al-jdaeh

The research by Otoom, A. F., and Al-Jdaeh, presented at the 9th International Conference in 2019, focuses on automated bug report classification. The study introduces NB-SVM-RT, a hybrid Naïve Bayes and Support Vector Machine (SVM)-based model designed to predict whether a bug report is:

1. Corrective Report – Related to defect fixing.
2. Perfective Report – Related to major maintenance and improvements.

Key Contributions of the Study:

- Hybrid Model Approach: The combination of Naïve Bayes (NB) and Support Vector

## Nature-Based Prediction Model of Bug Reports Based on Ensemble Machine Learning Model

Machine (SVM) enhances classification accuracy.

- **Feature Extraction & Processing:** The study applies text mining and natural language processing (NLP) techniques to extract meaningful features from bug reports.
- **Performance Improvement:** The NB-SVM-RT model achieves an impressive 93.1% accuracy, demonstrating the effectiveness of SVM-based classification for bug report categorization.
- **Application in Software Maintenance:** This method helps software developers automate bug report handling, improving efficiency in software maintenance workflows.

This research contributes significantly to software quality assurance, enabling faster and more accurate classification of bug reports for better maintenance decisions.

### 2.3 Predicts the severity of bug reports

**Authors:** Anh-Hien Dao and Cheng-Zen Yang

The study by Anh-Hien Dao and Cheng-Zen Yang, published in MDPI in 2021, focuses on predicting the severity of bug reports using advanced machine learning techniques. The research employs N-gram, Convolutional Neural Networks (CNN), and Random Forest with Boosting to enhance the accuracy and reliability of severity classification in bug tracking systems.

**Key Contributions of the Study:**

- **Severity Prediction:** The model automatically classifies bug reports based on severity levels, helping developers prioritize critical issues efficiently.
- **Machine Learning Techniques:**
  - N-gram: Extracts textual features from bug reports.
  - CNN: Captures deep semantic features for better classification.
  - Random Forest with Boosting: Enhances prediction accuracy and robustness.
- **Performance Metrics:**
  - Average Accuracy: 96.34%, indicating high precision in classification.
  - Average F-measure: 96.43%, demonstrating the model's effectiveness in handling imbalanced data.

This research significantly improves bug severity classification, aiding software teams in managing defect resolution efficiently and enhancing software maintenance processes.

## 2.4 Classifies bug reports according to ODC

**Authors:** Sarfraz Khurshid, Mariano Moscato

The study by Sarfraz Khurshid and Mariano Moscato, published in Springer in 2022, focuses on classifying bug reports based on Orthogonal Defect Classification (ODC), a structured approach to defect categorization. The research evaluates multiple machine learning models to determine their effectiveness in classifying bug reports.

Key Contributions of the Study:

- Bug Report Classification: The study categorizes bug reports according to ODC, which helps in understanding defect patterns and improving software quality.
- Machine Learning Models Used:
  - Naïve Bayes (NB)
  - Support Vector Machine (SVM)
  - K-Nearest Neighbors (KNN)
  - Recurrent Neural Network (RNN)
  - Nearest Centroid
  - Random Forest (RF)
- Dataset Size Impact: The study highlights that the accuracy of classifiers is affected by the dataset size, with larger datasets generally leading to better performance.

This research contributes to automating defect classification, enabling software teams to analyze and manage bug reports efficiently, improving defect prediction and software maintenance processes.

## 2.5 Predicts and assigns a priority level in binary classification (high or low)

**Authors:** M. Sharma, P. Bedi, K.K. Chaturvedi, and V.B. Singh,

The study by M. Sharma, P. Bedi, K.K. Chaturvedi, and V.B. Singh, presented at the 12th International Conference on Intelligent Systems Design and Applications in 2012, focuses on predicting and assigning priority levels (high or low) to bug reports using binary classification.

Key Contributions of the Study:

- Bug Report Prioritization: The model predicts whether a bug report should be classified as high-priority (requiring immediate attention) or low-priority (less urgent).
- Machine Learning Models Used:

## Nature-Based Prediction Model of Bug Reports Based on Ensemble Machine Learning Model

- Multinomial Naïve Bayes (MNB)
- Linear Support Vector Machine with Stochastic Gradient Descent (SGDC)
- Logistic Regression (LRC)
- Performance Metrics:
  - Accuracy: 0.908 (90.8%), indicating a high level of correctness in classification.
  - AUC (Area Under the Curve): 0.95, showing strong discriminative power.
  - F-measure: 0.892 (89.2%), reflecting a good balance between precision and recall.

This research contributes to automating bug report prioritization, improving software maintenance efficiency by helping teams focus on the most critical issues first.

## 2.6 A model for bug assignment recommendation

**Authors:** Mina Samir, Nada Sherief, and Walid Abdelmoez

The study by Mina Samir, Nada Sherief, and Walid Abdelmoez, published in Computers in 2023, presents a bug assignment recommendation model that helps automatically assign reported bugs to the most suitable developers.

Key Contributions of the Study:

- Bug Assignment Automation: The model predicts the best developer to handle a given bug report, improving software maintenance efficiency.
- Ensemble Learning Approach:
  - Utilizes Stacked Generalization (SG), an advanced ensemble learning technique.
  - Combines multiple classifiers, including:
    - Naïve Bayes (NB)
    - Support Vector Machine (SVM)
    - K-Nearest Neighbors (KNN)
    - Decision Tree (DT)
    - Bayes Net
- Performance Metrics:
  - Achieves accuracies ranging from 50% to 89%, depending on the dataset and classifier combination.

This research contributes to intelligent bug tracking systems, reducing manual effort in bug triaging and improving developer workload distribution.

## **2.7 Web-based tool for Bug assignment recommendation**

**Authors:** Caterina De Marco

The study by Caterina De Marco, presented at SANER 2023, introduces a web-based tool for bug assignment recommendation that helps automate the process of assigning bug reports to the most suitable developers.

Key Contributions of the Study:

- **Web-Based Bug Assignment System:** The tool provides an online platform for recommending developers for bug fixes, improving efficiency in software maintenance workflows.
- **Machine Learning Models Used:**
  - Support Vector Machine (SVM)
  - Naïve Bayes (NB)
  - C4.5 Decision Tree
  - Rule-Based Classifiers
- **Performance Metrics:**
  - Top-1 recommendation accuracy: 50% to 95% (correct developer at first suggestion).
  - Top-3 recommendation accuracy: 20% to 80% (correct developer within the first three suggestions).
  - Top-5 recommendation accuracy: 10% to 70% (correct developer within the first five suggestions).

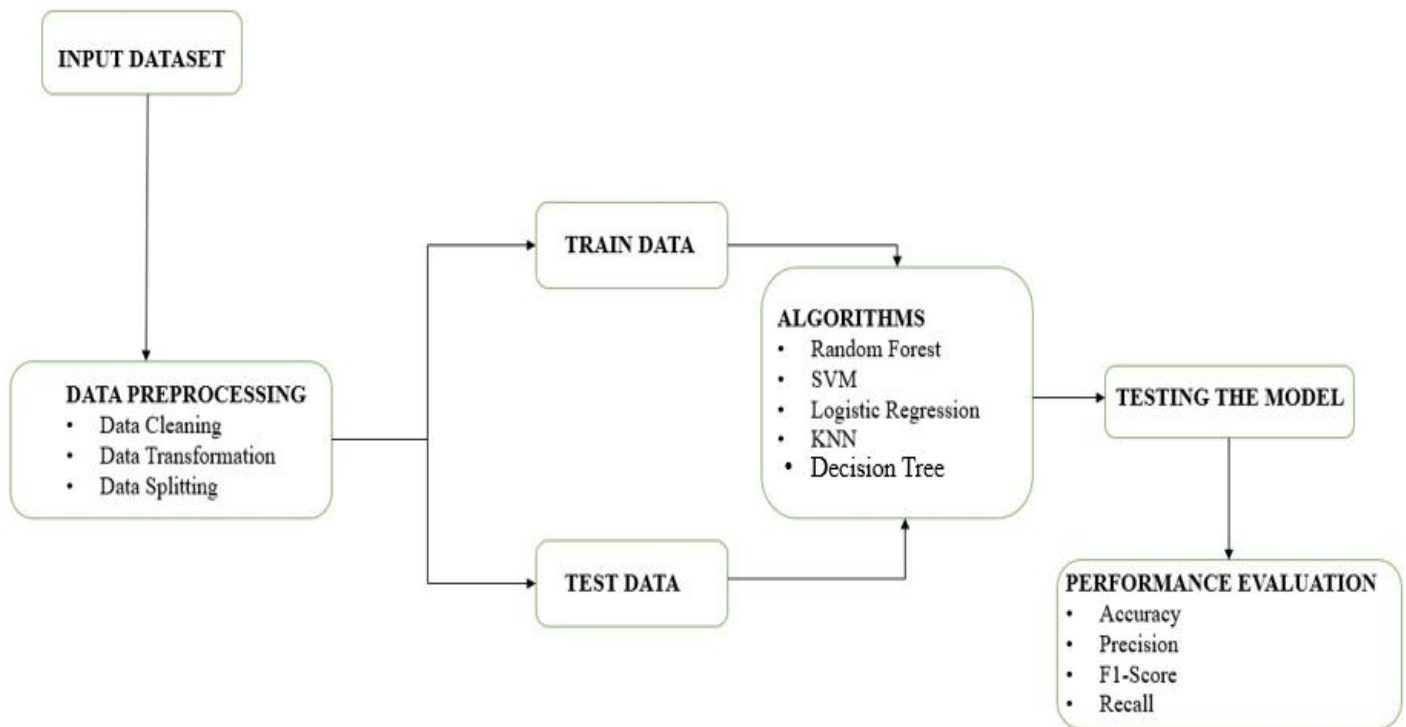
This research contributes to automated bug triaging, reducing manual effort in bug assignment and optimizing developer workload distribution in software development teams.



### 3. SYSTEM DESIGN

#### 3.1 ARCHITECTURE / BLOCK DIAGRAM

The architecture of a nature-inspired prediction model for bug reports integrates bio-inspired algorithms (such as Genetic Algorithms, Particle Swarm Optimization) with ensemble machine learning to enhance the accuracy of bug classification, severity prediction, and assignment. The architecture consists of the following stages



**Fig 3.1.1 Architecture Diagram**

#### Key Components:

- 1. Input Dataset:** The process begins with raw data, which serves as the foundation for model training and testing.
- 2. Data Preprocessing:** The dataset undergoes cleaning, transformation, and splitting to prepare. It for machine learning algorithms. This ensures data quality and readiness for training.
- 3. Train Data and Test Data:** The preprocessed data is divided into training and testing sets to Train models and evaluate their performance, respectively.

**4. Algorithms:** Classification models, including Naïve Bayes, SVM, Logistic Regression, and Decision Tree Classifier, are applied to the training data for risk classification.

**5. Testing the model:** The trained models are tested on unseen test data to assess their generalization capabilities.

**6. Performance Evaluation:** Finally, the models are evaluated using key metrics such as accuracy, precision, F1-score, and recall to determine their effectiveness in financial risk classification.

### 3.2 DATA FLOW DIAGRAM

Data Flow Diagram (DFD) is a visual representation that illustrates how data moves through a system. It's a valuable tool used in software engineering and systems analysis to understand, analyze, and design information systems. Below is the data flow diagram where the key components are service provider, user, web server and web storage. By understanding the flow of data within a system, DFDs help in making informed decisions about system design.

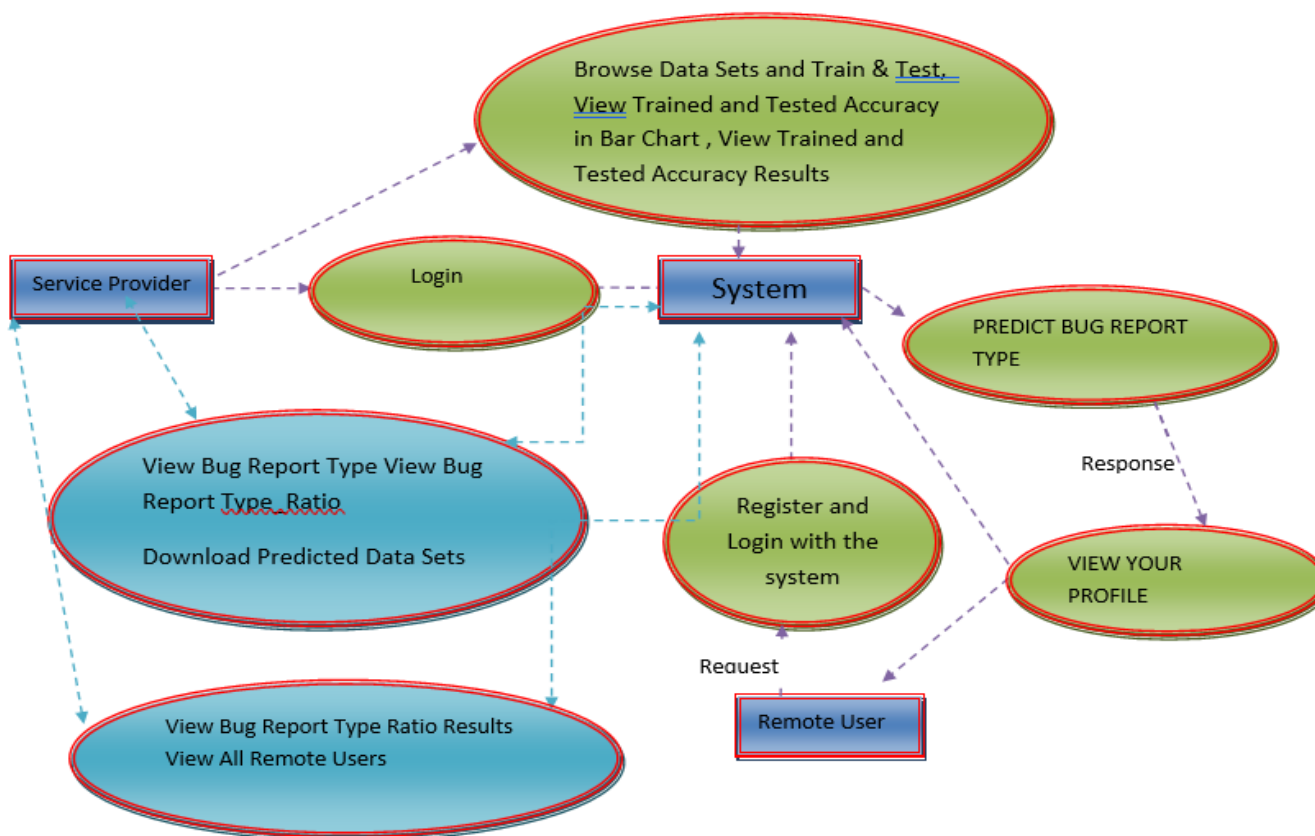


Fig 3.2.1 Data Flow Diagram

## **Nature-Based Prediction Model of Bug Reports Based on Ensemble Machine Learning Model**

### **User Interaction:**

The remote user initiates the data flow by interacting with the system through the “Login” or “Register and Login with the system” options.

### **Request:**

The user’s request, which could be to predict a used bug report type, view their profile, or access other functionalities, is sent to the system.

### **System Processing:**

The system receives the request and processes it based on the user’s input.

### **Prediction or Calculation:**

The system performs the required calculations or predictions, such as determining the used bug type based on the input data.

### **Response Generation:**

The system generates a response to the user’s request, which could be a predicted bug type, a profile view, or other relevant information.

### **Response Delivery:**

The system sends the generated response back to the remote user.

## **3.3 UML DIAGRAM**

In software engineering, a class diagram in Unified Modeling Language(UML) is a type of static structure diagram that describes the structure of a system by showing the system’s classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

### **GOALS:**

1. The Primary goals in the design of the UML are as follows:

## Nature-Based Prediction Model of Bug Reports Based on Ensemble Machine Learning Model

2. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
3. Provide extendibility and specialization mechanisms to extend the core concepts.
4. Be independent of particular programming languages and development process.
5. Provide a formal basis for understanding the modeling language.
6. Encourage the growth of OO tools market.
7. Support higher level development concepts such as collaborations, frameworks, patterns and components.
8. Integrate best practices.

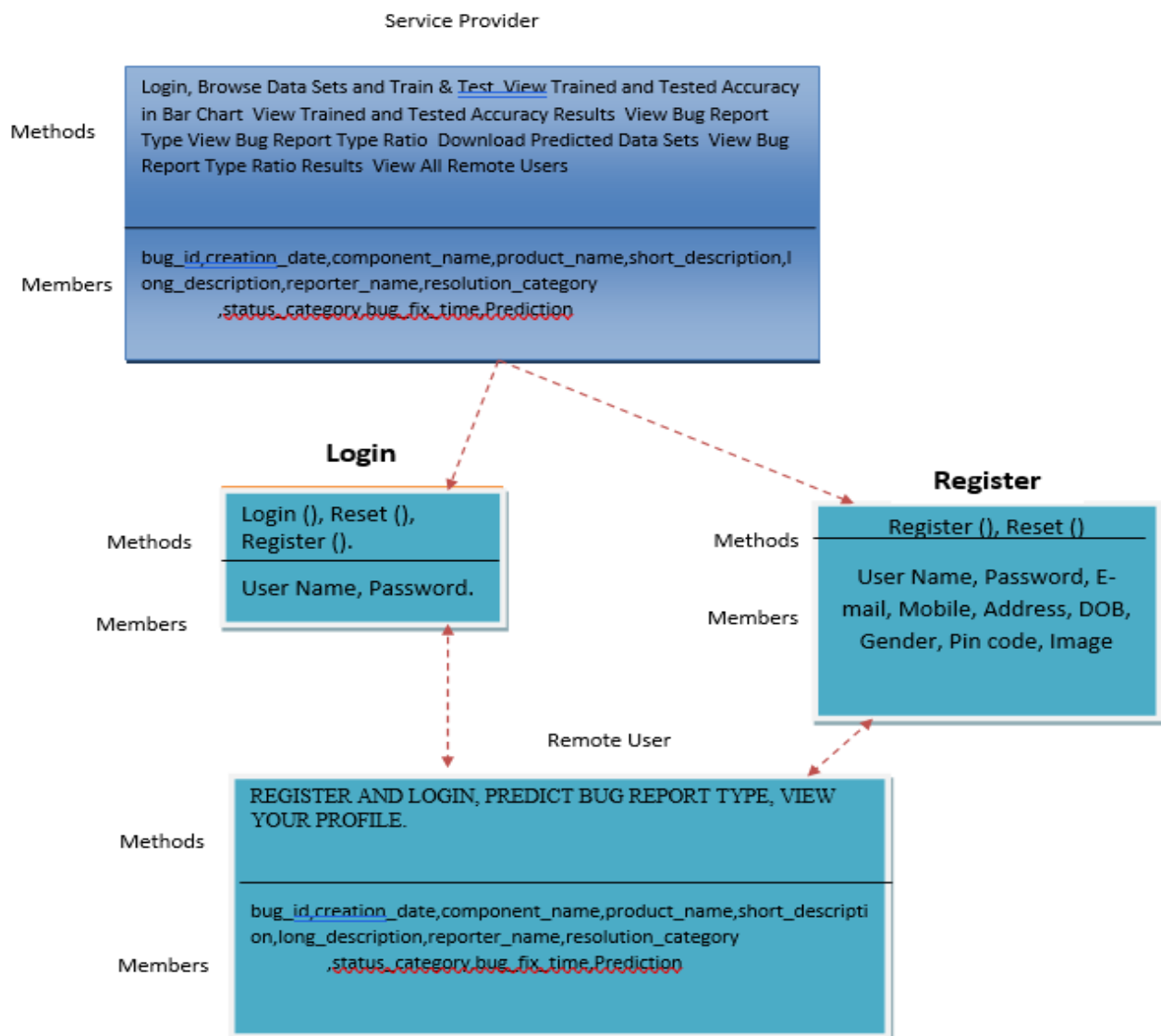


Fig 3.3.1 Class Diagram

## Nature-Based Prediction Model of Bug Reports Based on Ensemble Machine Learning Model

### Service Provider:

This class is responsible for handling various operations related to used bug data, including training, testing, prediction, and data analysis.

### User:

This class represent a user of the system, who can register, login, predict used bug type, and view their profile.

### Register:

This class is a specialized class for handling user registration and login functionalities.

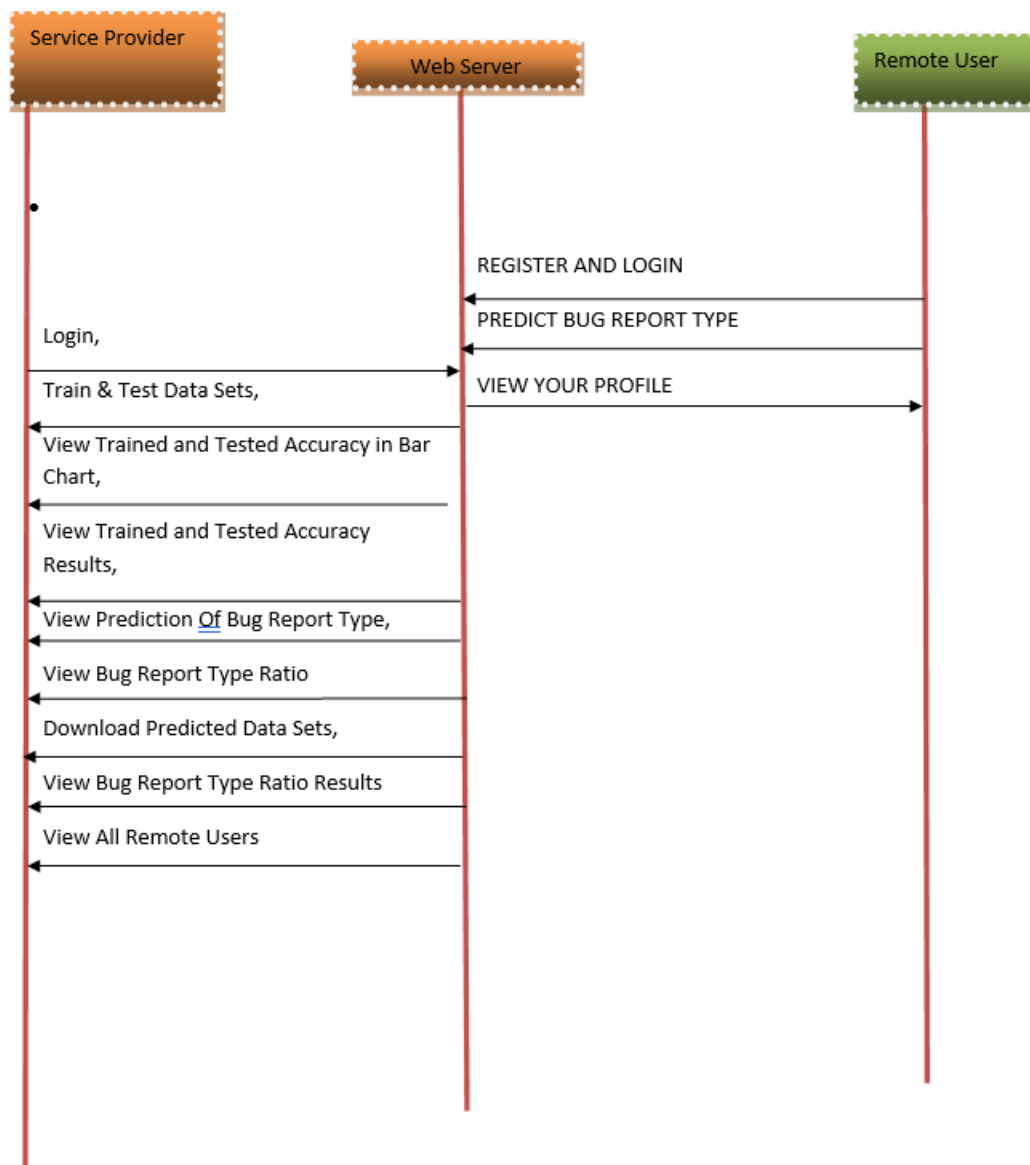


Fig: 3.2.2 Sequence Diagram

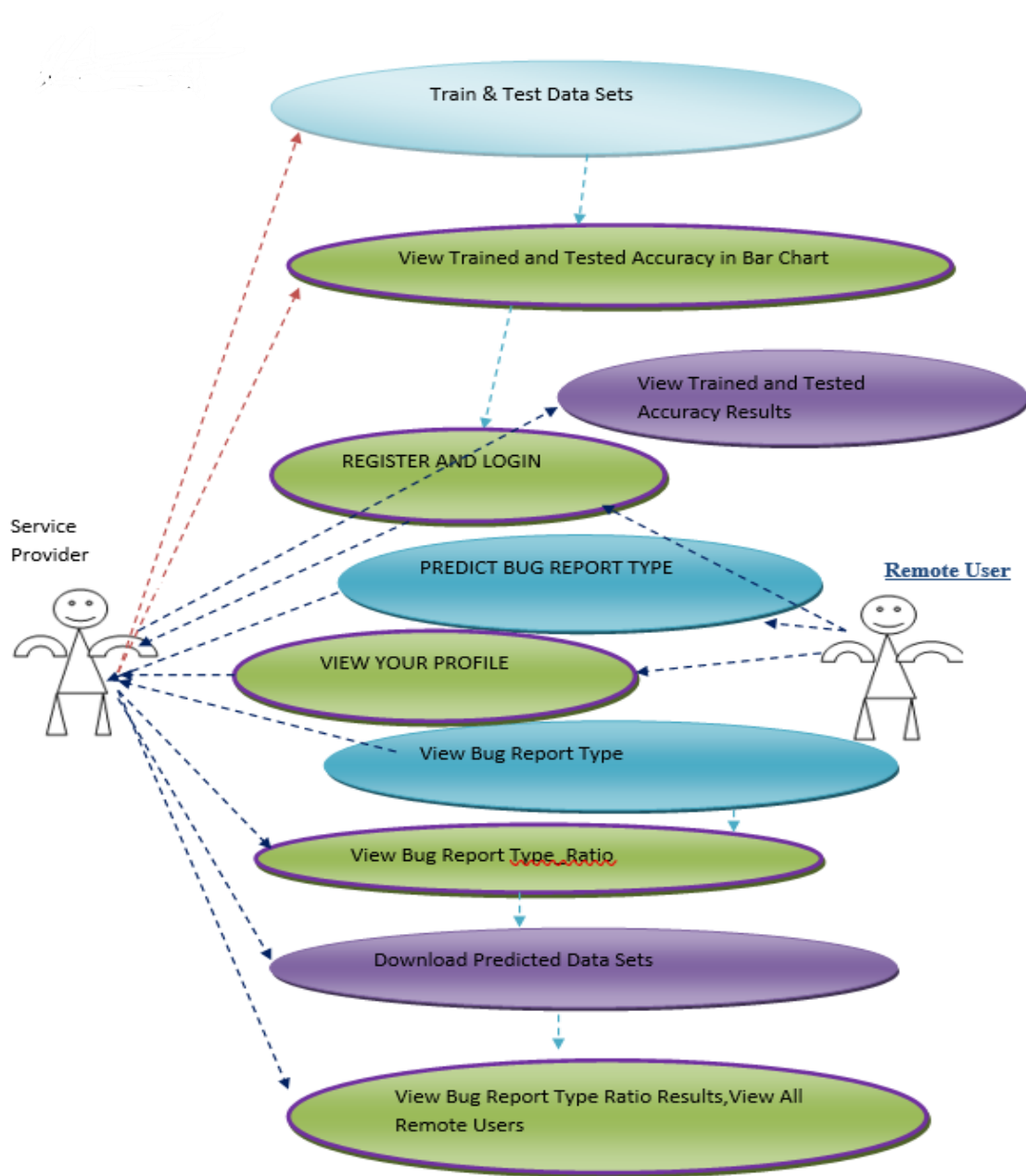


Fig: 3.2.3 Use case Diagram

## 4. IMPLEMENTATION

### 4.1 PROJECT MODULES

#### 1. Data Collection and Preprocessing

- Data was gathered from websites and newspaper advertisements comprising 200 records with attributes such as year, make, engine capacity, mileage, and price.
- Preprocessing involved handling missing values, normalizing features, and encoding categorical data.

#### 2. Model Development

Algorithms used:

- Random Forest
- Support Vector Machine (SVM)
- Logistic Regression
- KNN
- Decision Tree Classifier

#### 3. Evaluation and Results

- Performance Metrics: Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE).
- Results: SVM outperformed others, with SVM showing the highest accuracy.

#### 4. User Interaction

- Service Provider Module: For training, testing models, and analyzing results.
- Remote User Module: Allowed users to predict car prices based on input attributes.

#### 5. Deployment

- A user-friendly interface enables real-time predictions and model management, ensuring the system is accessible and effective.

## 4.2 ALGORITHMS

The core of the project relies on machine learning algorithms to predict the nature of bug report type. The following algorithms were implemented and tested:

### 1.Support Vector Machine (SVM)

**Purpose:** SVM is a supervised learning algorithm used for classification and regression. It is ideal for datasets with clear margins of separation and is effective in high-dimensional spaces.

**Implementation:**

- Transform data to a higher dimension using a kernel function if necessary.
- Find a hyperplane that maximizes the margin between classes.
- Use the hyperplane to classify new data points.

### 2.Logistic Regression

**Purpose:** Logistic regression is used for binary classification (or multiclass using extensions). It models the probability of a target class using a sigmoid function.

**Implementation:**

- Calculate the weighted sum of input features.
- Apply the sigmoid function to map outputs to a probability range (0, 1).
- Classify based on a threshold (e.g., 0.5).

### 3.Decision Tree Classifier

**Purpose:** Decision trees are used for classification and regression tasks. They model decisions and their possible consequences as a tree structure, making them interpretable.

**Implementation:**

- Start with the entire dataset at the root.
- Split the data based on the feature that provides the highest information gain or lowest Gini impurity.
- Repeat splits recursively until stopping criteria are met.

### 4.Random Forest (RF)

**Purpose:** An ensemble method that improves classification and regression accuracy by combining multiple decision trees.



### Implementation:

- It can handle the data set containing continuous variables, as in the case of regression, and categorical variables, as in the case of classification.
- It performs better for classification and regression tasks.
- It also provides an exceptionally high level of precision.

### 5.K-Nearest Neighbors (KNN)

**Purpose:** Classifies data based on the majority class among its nearest neighbors, useful in pattern recognition.

### Implementation:

- It classifies or predicts new data points based on their similarity to existing data points.
- Classification decision rule and confusion matrix.
- Feature transformation.
- Performance assessment with cross-validation.

### Advantages:

#### 1. Automation of Complex Tasks

Machine learning (ML) automates tasks that are too complex for traditional programming, such as image recognition, speech processing, and anomaly detection.

#### 2.Improved Accuracy and Predictions

ML algorithms improve with data. They can provide highly accurate predictions and classifications, especially in tasks like fraud detection, recommendation systems, or medical diagnostics.

#### 3.Scalability

Once trained, ML models can handle vast amounts of data, making them suitable for projects requiring real-time processing or large-scale deployment.

#### 4.Versatility

ML algorithms are versatile and applicable to diverse fields like finance, healthcare, marketing, and robotics.

### **5.Data Insights**

ML helps uncover hidden patterns and relationships in data, enabling informed decision- making and innovation.

### **6.Customization**

Algorithms can be tailored to specific project needs, ensuring adaptability to varying scenarios and data types.

## **Disadvantages**

### **1.High Data Dependency**

ML algorithms require large, high-quality datasets for training. Poor or insufficient data can lead to unreliable models.

### **2.Complexity and Interpretability**

Many ML models, like deep learning, are black-box models, making them difficult to interpret and explain, especially in critical applications like healthcare or law.

### **3.Resource Intensive**

Training ML models can be computationally expensive and time-consuming, requiring powerful hardware and optimization.

### **4.Overfitting or Underfitting**

If not tuned properly, models may overfit (memorize the training data) or underfit (fail to learn adequately), leading to poor generalization.

### **5.Maintenance Challenges**

ML models require regular updates and retraining as data and conditions evolve, increasing maintenance overhead.

### **6.Ethical and Bias Issues**

ML models can perpetuate biases present in the training data, leading to unfair outcomes and ethical concerns in sensitive domains.

### **7. Dependency on Expertise**

Implementing and optimizing ML algorithms requires specialized knowledge, which can limit accessibility for non-experts.

## 5. TESTING

### 5.1 TESTING METHODS

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

The following are the Testing Methodologies:

- o **Unit Testing.**
- o **Integration Testing.**
- o **User Acceptance Testing.**
- o **Output Testing.**
- o **Validation Testing.**

#### 5.1.1 Unit Testing

Unit testing focuses verification effort on the smallest unit of Software design that is the module. Unit testing exercises specific paths in a module's control structure to ensure complete coverage and maximum error detection. This test focuses on each module individually, ensuring that it functions properly as a unit. Hence, the naming is Unit Testing.

During this testing, each module is tested individually and the module interfaces are verified for the consistency with design specification. All important processing paths are tested for the expected results. All error handling paths are also tested.

#### 5.1.2 Integration Testing

Integration testing addresses the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of high order tests are conducted. The main objective in this testing process is to take unit tested modules and build a program structure that has been dictated by design.

**The following are the types of Integration Testing:**

- **Top Down Integration**

This method is an incremental approach to the construction of program structure. Modules are

integrated by moving downward through the control hierarchy, beginning with the main program module. The module subordinates to the main program module are incorporated into the structure in either a depth first or breadth first manner.

In this method, the software is tested from main module and individual stubs are replaced when the test proceeds downwards.

### **• Bottom-up Integration**

This method begins the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from the bottom up, processing required for modules subordinate to a given level is always available and the need for stubs is eliminated. The bottom up integration strategy may be implemented with the following steps:

- The low-level modules are combined into clusters into clusters that perform a specific Software sub-function.
- A driver (i.e.) the control program for testing is written to coordinate test case input and output.
- The cluster is tested.
- Drivers are removed and clusters are combined moving upward in the program structure

The bottom up approaches tests each module individually and then each module is module is integrated with a main module and tested for functionality.

### **5.1.3 User Acceptance Testing**

User Acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes wherever required. The system developed provides a friendly user interface that can easily be understood even by a person who is new to the system.

### **5.1.4 Output Testing**

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the outputs generated or displayed by the system under consideration. Hence the output format is considered in 2 ways – one is on screen and another in printed format.

### 5.1.5 Validation Testing

Validation checks are performed on the following fields:

#### **Text Field:**

The text field can contain only the number of characters lesser than or equal to its size. The text fields are alphanumeric in some tables and alphabetic in other tables. Incorrect entry always flashes and error message.

#### **Numeric Field:**

The numeric field can contain only numbers from 0 to 9. An entry of any character flashes an error messages. The individual modules are checked for accuracy and what it has to perform. Each module is subjected to test run along with sample data. The individually tested modules are integrated into a single system. Testing involves executing the real data information is used in the program the existence of any program defect is inferred from the output. The testing should be planned so that all the requirements are individually tested.

A successful test is one that gives out the defects for the inappropriate data and produces and output revealing the errors in the system.

#### **Preparation of Test Data**

Taking various kinds of test data does the above testing. Preparation of test data plays a vital role in the system testing. After preparing the test data the system under study is tested using that test data. While testing the system by using test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

#### **Using Live Test Data:**

Live test data are those that are actually extracted from organization files. After a system is partially constructed, programmers or analysts often ask users to key in a set of data from their normal activities. Then, the systems person uses this data as a way to partially test the system. In other instances, programmers or analysts extract a set of live data from the files and have them entered themselves.

It is difficult to obtain live data in sufficient amounts to conduct extensive testing. And, although it is realistic data that will show how the system will perform for the typical processing requirement, assuming that the live data entered are in fact typical, such data generally will not test all combinations or formats that can enter the system. This bias toward typical values then

does not provide a true systems test and in fact ignores the cases most likely to cause system failure.

### **Using Artificial Test Data:**

Artificial test data are created solely for test purposes, since they can be generated to test all combinations of formats and values. In other words, the artificial data, which can quickly be prepared by a data generating utility program in the information systems department, make possible the testing of all login and control paths through the program.

The most effective test programs use artificial test data generated by persons other than those who wrote the programs. Often, an independent team of testers formulates a testing plan, using the systems specifications.

The package “Virtual Private Network” has satisfied all the requirements specified as per software requirement specification and was accepted.

## **5.2 USER TRAINING**

Whenever a new system is developed, user training is required to educate them about the working of the system so that it can be put to efficient use by those for whom the system has been primarily designed. For this purpose, the normal working of the project was demonstrated to the prospective users. Its working is easily understandable and since the expected users are people who have good knowledge of computers, the use of this system is very easy.

## **5.3 MAINTAINENCE**

This covers a wide range of activities including correcting code and design errors. To reduce the need for maintenance in the long run, we have more accurately defined the user’s requirements during the process of system development. Depending on the requirements, this system has been developed to satisfy the needs to the largest possible extent. With development in technology, it may be possible to add many more features based on the requirements in future. The coding and designing is simple and easy to understand which will make maintenance easier.

### **TESTING STRATEGY:**

A strategy for system testing integrates system test cases and design techniques into a well planned series of steps that results in the successful construction of software. The testing strategy must co-operate test planning, test case design, test execution, and the resultant data collection and

evaluation. A strategy for software testing must accommodate low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high level tests that validate major system functions against user requirements.

Software testing is a critical element of software quality assurance and represents the ultimate review of specification design and coding. Testing represents an interesting anomaly for the software. Thus, a series of testing are performed for the proposed system before the system is ready for user acceptance testing.

### **SYSTEM TESTING:**

Software once validated must be combined with other system elements (e.g. Hardware, people, database). System testing verifies that all the elements are proper and that overall system function performance is achieved. It also tests to find discrepancies between the system and its original objective, current specifications and system documentation.

### **UNIT TESTING:**

In unit testing different are modules are tested against the specifications produced during the design for the modules. Unit testing is essential for verification of the code produced during the coding phase, and hence the goals to test the internal logic of the modules. Using the detailed design description as a guide, important Conrail paths are tested to uncover errors within the boundary of the modules. This testing is carried out during the programming stage itself. In this type of testing step, each module was found to be working satisfactorily as regards to the expected output from the module.

In Due Course, latest technology advancements will be taken into consideration. As part of technical build-up many components of the networking system will be generic in nature so that future projects can either use or interact with this. The future holds a lot to offer to the development and refinement of this project.



## 6. RESULTS

A large number of experiments have been conducted in order to find the best network structure and the best parameters for the neural network. We found that a neural network with 1 hidden layer and 2 nodes produced the smallest mean absolute error among various neural network structures that were experimented with. However, we found that Support Vector Regression and a multilayer perceptron with back-propagation produced slightly better predictions than linear regression while the k-Nearest Neighbor algorithm had the worst accuracy among these four approaches. All experiments were performed with a cross-validation value of 10 folds. The results are summarized in Table 6.1 below:

Sl. no	Model Type	Accuracy
1.	SVM	71.68%
2.	Logistic Regression	75.32%
3.	Decision Tree Classifier	71.20%
4.	KNeighbors Classifier	75.63%
5.	Random Forest	77.22%

**Table 6.1: Model Type and Accuracy**

Table 6.1 presents the accuracy scores of four different machine learning models applied to a specific dataset. Random Forest achieved the highest accuracy of 77.22%, followed by K Neighbors Classifier(75.63%), Logistic Regression (51.12%), Support Vector Machine(SVM) (71.68%), and Decision Tree Classifier (71.20%). These results suggest that Random Forest is the most suitable model for this particular task, but further analysis may be necessary to confirm its superiority and identify potential areas for improvement.

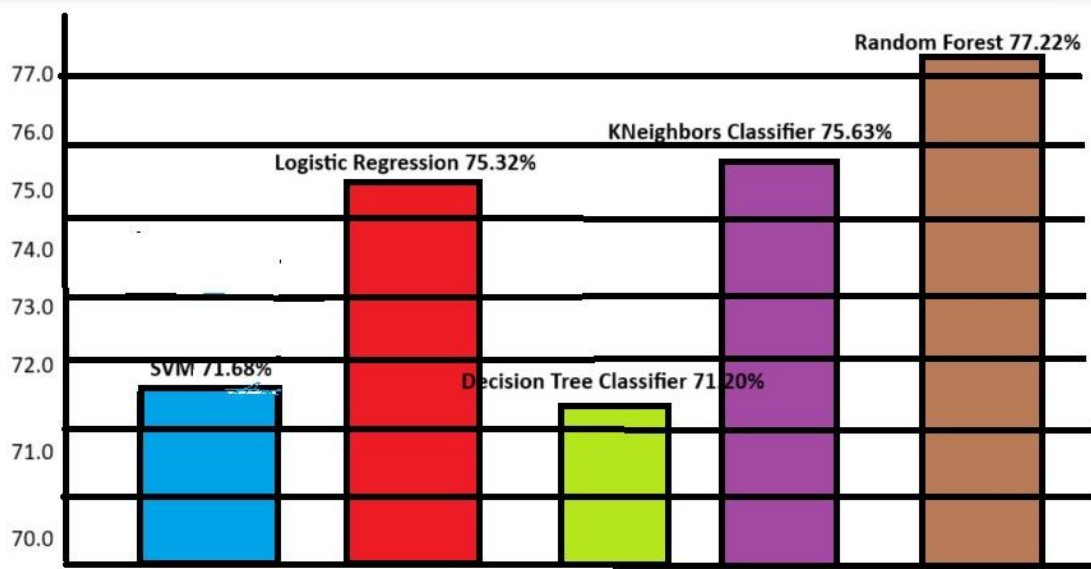


Fig 6.1 Model vs Accuracy Graph

The figure shows the variation in the accuracies in different machine learning models. As we can see the support vector machine model performed well against various models. The data in graph is fairly accurate and can be relied upon in many cases.

## **7. CONCLUSION**

The proposed nature-based prediction model of bug reports on the basis of ensemble machine learning effectively enhances the accuracy and effectiveness of bug classification. Compared to existing methodologies, the system classifies bugs automatically, reducing the effort required and improving software maintenance processes.

This model takes four base machine learning algorithms—Random Forest, Support Vector Classification, Logistic Regression, and Multinomial Naïve Bayes—with an accuracy of 90.42%. Additionally, using a text augmentation technique, the model accuracy is also improved to 96.72%, which indicates its stability in dealing with diverse bug reports.

As opposed to the conventional approach where the central focus is on severity and priority classification, the proposed system offers a nature-based classification technique that classifies bugs into six categories: Program Anomaly, GUI, Network or Security, Configuration, Performance, and Test-Code. The enhanced classification facilitates enhanced defect handling through a detailed structured classification framework in bug reports.

Overall, this model provides a promising direction towards automated bug triaging, resulting in more accurate, efficient, and intelligent defect management for software development and maintenance.

## 8. REFERENCES

- [1] M. A. Jamil, M. Arif, N. S. A. Abubakar, and A. Ahmad, “Software testing techniques: A literature review,” in Proc. 6th Int. Conf. Inf. Commun. Technol. Muslim World (ICT4M), Nov. 2016, pp. 177–182.
- [2] W. Y. Ramay, Q. Umer, X. C. Yin, C. Zhu, and I. Illahi, “Deep neural network-based severity prediction of bug reports,” IEEE Access, vol. 7, pp. 46846–46857, 2019.
- [3] W. Wen, “Using natural language processing and machine learning techniques to characterize configuration bug reports: A study,” M.S. thesis, College Eng., Univ. Kentucky, Lexington, KY, USA, 2017.
- [4] J. Polpinij, “A method of non-bug report identification from bug report repository,” Artif. Life Robot., vol. 26, no. 3, pp. 318–328, Aug. 2021.
- [5] S. Adhikarla, “Automated bug classification.: Bug report routing,” M.S. thesis, Fac. Arts Sci., Dept. Comput. Inf. Sci., Linköping Univ., Linköping, Sweden, 2020.
- [6] K. C. Youm, J. Ahn, and E. Lee, “Improved bug localization based on code change histories and bug reports,” Inf. Softw. Technol., vol. 82, pp. 177–192, Feb. 2017.
- [7] N. Safdari, H. Alrubaye, W. Aljedaani, B. B. Baez, A. DiStasi, and M. W. Mkaouer, “Learning to rank faulty source files for dependent bug reports,” in Proc. SPIE, vol. 10989, 2019, Art. no. 109890B.
- [8] A. Kukkar, R. Mohana, A. Nayyar, J. Kim, B.-G. Kang, and N. Chilamkurti, “A novel deep-learning-based bug severity classification technique using convolutional neural networks and random forest with boosting,” Sensors, vol. 19, no. 13, p. 2964, Jul. 2019.
- [9] A. Aggarwal. (May 2020). Types of Bugs in Software Testing: 3 Classifications With Examples. [Online]. Available: <https://www.scnsoft.com/software->

testing/types-of-bugs

- [10] A. Kukkar and R. Mohana, “A supervised bug report classification with incorporate and textual field knowledge,” *Proc. Comput. Sci.*, vol. 132, pp. 352–361, Jan. 2018.
- [11] A. F. Otoom, S. Al-jdaeh, and M. Hammad, “Automated classification of software bug reports,” in *Proc. 9th Int. Conf. Inf. Commun. Manage.*, Aug. 2019, pp. 17–21.
- [12] P. J. Morrison, R. Pandita, X. Xiao, R. Chillarege, and L. Williams, “Are vulnerabilities discovered and resolved like other defects?” *Empirical Softw. Eng.*, vol. 23, no. 3, pp. 1383–1421, Jun. 2018.
- [13] F. Lopes, J. Agnelo, C. A. Teixeira, N. Laranjeiro, and J. Bernardino, “Automating orthogonal defect classification using machine learning algorithms,” *Future Gener. Comput. Syst.*, vol. 102, pp. 932–947, Jan. 2020.
- [14] T. Hirsch and B. Hofer, “Root cause prediction based on bug reports,” in *Proc. IEEE Int. Symp. Softw. Rel. Eng. Workshops (ISSREW)*, Oct. 2020, pp. 171–176.
- [15] Q. Umer, H. Liu, and I. Illahi, “CNN-based automatic prioritization of bug reports,” *IEEE Trans. Rel.*, vol. 69, no. 4, pp. 1341–1354, Dec. 2020.
- [16] H. Bani-Salameh, M. Sallam, and B. Al Shboul, “A deep-learning-based bug priority prediction using RNN-LSTM neural,” *e-Inform. Softw. Eng. J.*, vol. 15, no. 1, pp. 1–17, 2021.
- [17] Ö. Köksal and B. Tekinerdogan, “Automated classification of unstructured bilingual software bug reports: An industrial case study research,” *Appl. Sci.*, vol. 12, no. 1, p. 338, Dec. 2021.
- [18] B. Alkhazi, A. DiStasi, W. Aljedaani, H. Alrubaye, X. Ye, and M. W. Mkaouer, “Learning to rank developers for bug report assignment,” *Appl. Soft Comput.*, vol. 95, Oct. 2020, Art. no. 106667.
- [19] L. Jonsson, M. Borg, D. Broman, K. Sandahl, S. Eldh, and P. Runeson,

“Automated bug assignment: Ensemble-based machine learning in large scale industrial contexts,” *Empirical Softw. Eng.*, vol. 21, no. 4, pp. 1533–1578, Aug. 2016.

[20] X. Ye, R. Bunescu, and C. Liu, “Learning to rank relevant files for bug reports using domain knowledge,” in *Proc. 22nd ACM SIGSOFT Int. Symp. Found. Softw. Eng.*, Nov. 2014, pp. 689–699.

[21] Y. Tian, D. Wijedasa, D. Lo, and C. Le Goues, “Learning to rank for bug report assignee recommendation,” in *Proc. IEEE 24th Int. Conf. Program Comprehension (ICPC)*, May 2016, pp. 1–10.

[22] D. Devaiya, *Castr: A Web-Based Tool for Creating Bug Report Assignment Recommenders*. Lethbridge, AB, Canada: Univ. Lethbridge, 2019.

[23] M. Alenezi, S. Banitaan, and M. Zarour, “Using categorical features in mining bug tracking systems to assign bug reports,” 2018, arXiv:1804.07803.

[24] H. A. Ahmed, N. Z. Bawany, and J. A. Shamsi, “CaPBug-a framework for automatic bug categorization and prioritization using NLP and machine learning algorithms,” *IEEE Access*, vol. 9, pp. 50496–50512, 2021.

[25] R.-M. Karampatsis and C. Sutton, “How often do single-statement bugs occur?: The ManySStuBs4J dataset,” in *Proc. 17th Int. Conf. Mining Softw. Repositories*, Jun. 2020, pp. 573–577, doi: 10.1145/3379597.3387491.

[26] X. Han, T. Yu, and D. Lo, “PerfLearner: Learning from bug reports to understand and generate performance test frames,” in *Proc. 33<sup>rd</sup> IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE)*, Sep. 2018, pp. 17–28.

[27] P. E. Strandberg, T. J. Ostrand, E. J. Weyuker, W. Afzal, and D. Sundmark, “Intermittently failing tests in the embedded systems domain,” in *Proc. 29th ACM SIGSOFT Int. Symp. Softw. Test. Anal.*, Jul. 2020, pp. 337–348, doi: 10.1145/3395363.3397359.

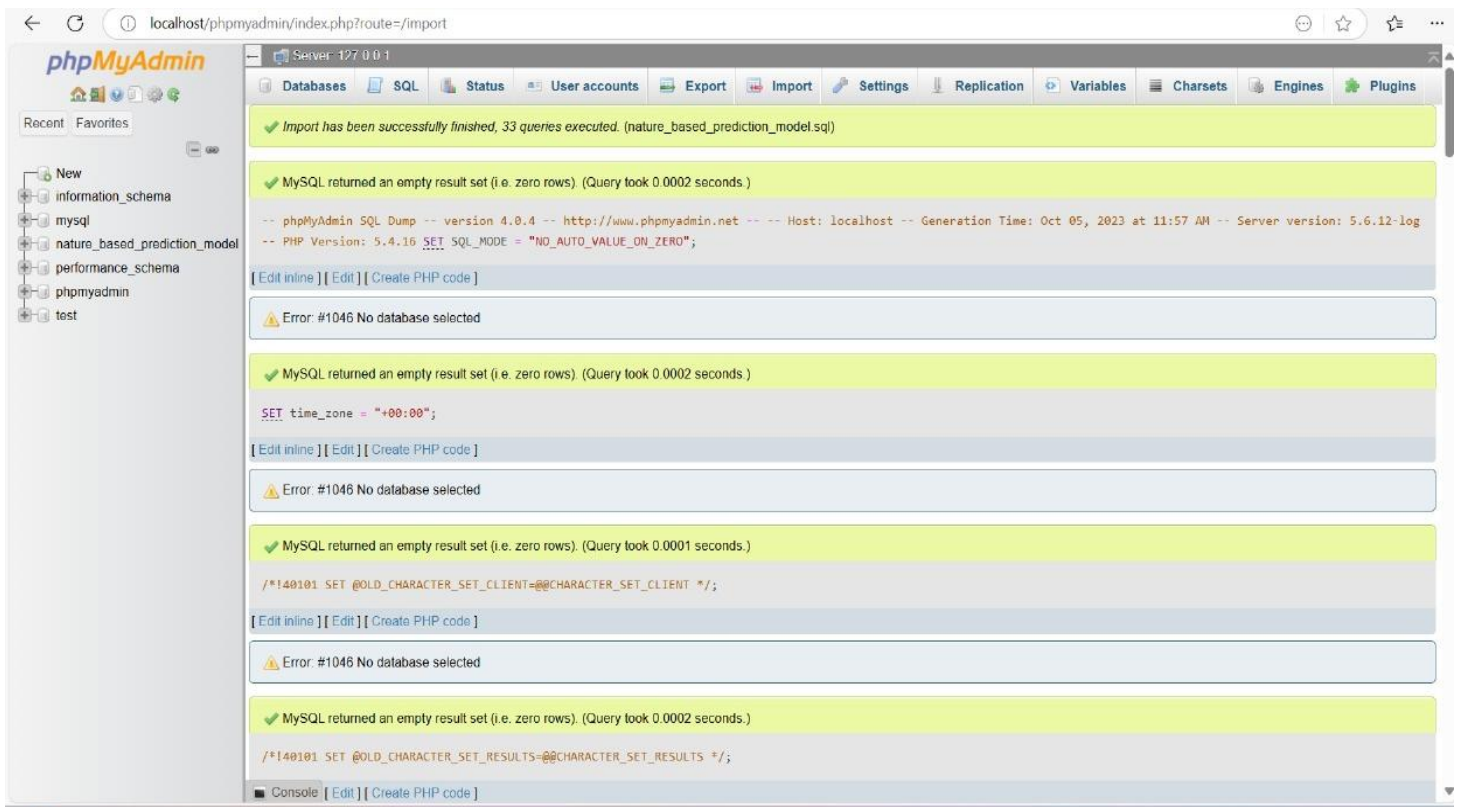
## APPENDIX I – SCREENSHOTS

	A	B	C	D	E	F	G	H	I	J	K	L
1	bug_id	creation_c	componen	product_n	short_desc	long_desc	reporter_r	resolution	status_cat	bug_fix_tir	severity_category	
2	FREEDESK	30-09-09	New Acco	FREEDESK	New acco	Created	amaasikas	fixed	resolved	0	normal	
3	XORG-205	#####	Document	XORG	XChangePr	The	matthieu.f	fixed	resolved	2	normal	
4	XORG-799	13-06-22	Driver/inte	XORG	intel-virtu	Created	main.haar	fixed	resolved	0	normal	
5	LIBREOFFI	26-01-12	Writer	LIBREOFFI	: The way	Problem	blackjay	fixed	closed	70	major	
6	POLICYKIT	26-08-22	daemon	POLICYKIT	[patch] pk	Google's	hanno	fixed	resolved	1	normal	
7	PKG-CONF	22-12-10	src	PKG-CONF	c99ism in	Version	hauke	fixed	resolved	120	normal	
8	MESA-863	16-11-22	Drivers/Ga	MESA	[RadeonSi	Created	madcatx	fixed	resolved	259	normal	
9	POPPLER-7	20-02-22	utils	POPPLER	pdftops 0.	pdftops -	williambac	fixed	resolved	23	normal	
10	XORG-270	#####	Driver/inte	XORG	Switch to c	Using the	sndirsch	fixed	closed	4	normal	
11	XORG-150	16-03-08	App/other	XORG	xorg/app/	Created	pcpa	fixed	resolved	256	normal	
12	POPPLER-(	#####	glib fronte	POPPLER	mem leak	font_info	carlosgc	fixed	resolved	11	normal	
13	XORG-212	15-04-09	Input/syna	XORG	The synapt	For some	peter.hutt	fixed	resolved	19	normal	
14	LIBREOFFI	21-01-11	Spreadshe	LIBREOFFI	[FORMAT	When I	mistresssil	fixed	resolved	322	normal	
15	XORG-238	#####	Driver/geo	XORG	xf86-video	Doing a	memsize	fixed	resolved	216	normal	
16	SWFDEC-1	25-12-07	library	SWFDEC	Wrong url	Version:	riccardo.m	fixed	resolved	31	normal	
17	TELEPATH'	19-07-11	tp-glib	TELEPATH'	TP_CONTA	I noticed	will	fixed	resolved	8	normal	
18	LIBREOFFI	20-10-10	Libreoffice	LIBREOFFI	LibreOffice	While	PeterSHan	fixed	closed	13	normal	
19	SYSTEMD-	21-08-13	general	SYSTEMD	systemd-n	systemd-	maxposed	fixed	resolved	77	normal	
20	LIBREOFFI	#####	Presentati	LIBREOFFI	SLIDESHOW	Libreoffic	wmzhere	fixed	resolved	60	major	
21	TELEPATH'	4/5/2010	tp-glib	TELEPATH'	telepathy-	Subject	smcv	fixed	resolved	0	blocker	
22	MESA-592	#####	Drivers/Ga	MESA	Undefined	Mesa	tdroste	fixed	resolved	1	blocker	
23	XORG-120	27-08-04	Lib/Xlib	XORG	multiple de	in make	ago	fixed	resolved	4	blocker	
24	CAIRO-831	17-09-06	pdf backer	CAIRO	cairo_pdf	as a	hans	fixed	resolved	1	normal	
25	LIBREOFFI	#####	WWW	LIBREOFFI	Bugzilla Ve	We need	LibreOffice	fixed	resolved	9	normal	

**Fig 1: The dataset of Bug Reports**

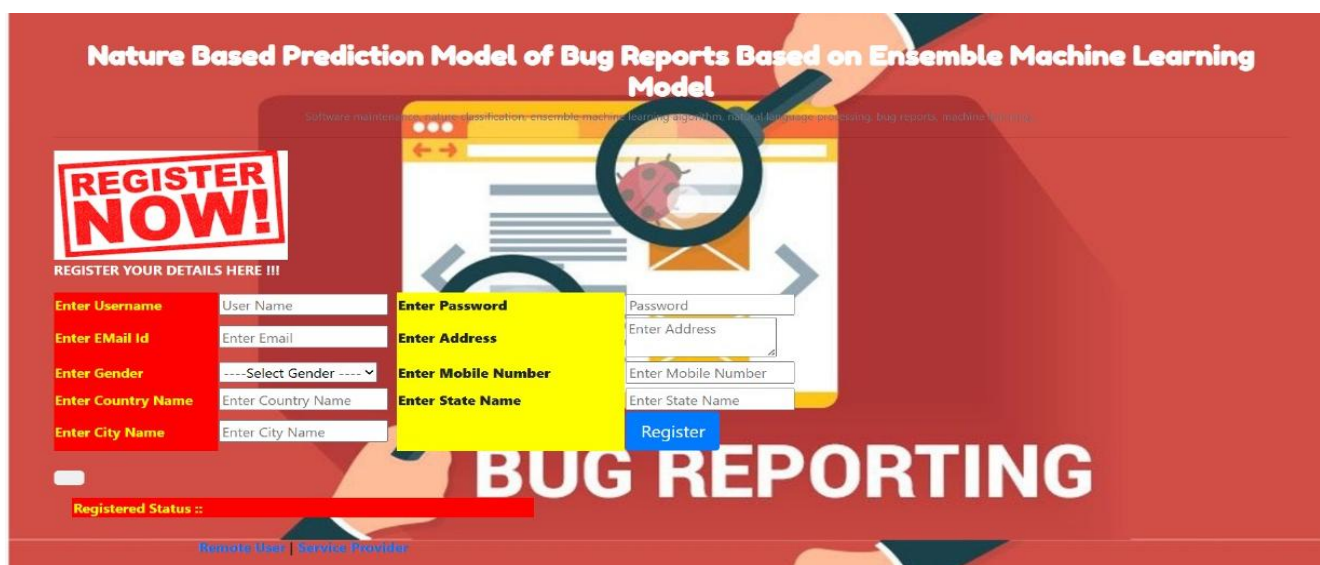
The dataset which consists of information about bug reports collected from various sources, including websites and newspapers. It contains key features such as the bug id, creation date, component name, product name, short description, long description, reporter name, resolution category, status category, bud fix time and severity category. This dataset serves as the foundation for building and training the machine learning models used in the project to predict nature of bugs accurately.

## Nature-Based Prediction Model of Bug Reports Based on Ensemble Machine Learning Model



**Fig 2: Upload the Database**

XAMPP is used to upload the bug dataset into phpMyAdmin, a web-based tool for managing MySQL databases. The data is stored and organized within a MySQL database, allowing efficient access and management. This setup enables seamless integration between the database and the machine learning models used for bug nature prediction.



**Fig 3: User Registration Page**



## Nature-Based Prediction Model of Bug Reports Based on Ensemble Machine Learning Model

The User Registration Page allows new users to create an account by providing details such as their name, email, and password. This ensures secure access to the system, enabling users to log in and utilize features like entering car details and predicting bug types. All user information is securely stored in the database for authentication and profile management.



**Fig 4: User Login Page**

The user logs into the profile using his credentials. The service provider has to choose the service provider option before login to access the admin profile.

## Nature-Based Prediction Model of Bug Reports Based on Ensemble Machine Learning Model

The screenshot shows a web browser window with the URL `127.0.0.1:8000/Predict_Bug_Report_Type/`. The page has a red header with the title "Nature Based Prediction Model of Bug Reports Based on Ensemble Machine Learning Model". Below the header is a navigation bar with links: "PREDICT BUG REPORT TYPE", "VIEW YOUR PROFILE", and "LOGOUT". The main content area contains a form with two columns of input fields. The left column includes: "Enter bug\_id" (text input with value "FREEDESKTOP.ORG-242"), "Enter component\_name" (text input with value "New Accounts"), "Enter short\_description" (text input with value "development"), "Enter reporter\_name" (text input with value "amaasikas"), and "Enter status\_category" (text input with value "resolved"). The right column includes: "Enter creation\_date" (text input with value "30-09-2009"), "Enter product\_name" (text input with value "FREEDESKTOP.ORG"), "Enter long\_description" (text input with value "Created attachment 29953"), "Enter resolution\_category" (text input with value "fixed"), and "Enter bug\_fix\_time" (text input with value "0"). Below the form is a button labeled "Predict". At the bottom of the form area, there is a red box with the text "PREDICTION OF BUG REPORT TYPE". The browser's taskbar at the bottom shows the system clock as 18:19 on 12-03-2025.

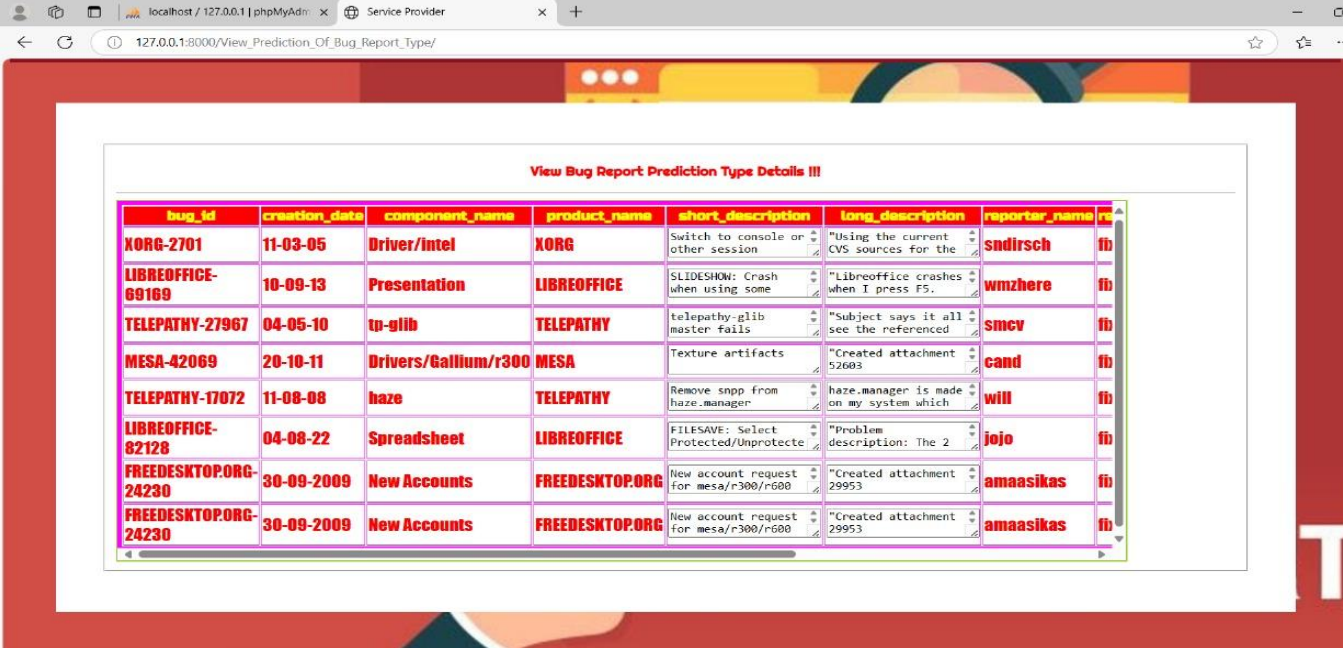
**Fig 5: Details of Bug Report**

To Predict the nature of bug report users must provide key details about the one bug report including bug id, creation date, component name, product name, short description, long description, reporter name, resolution category, status category, and bug fix time. These features are essential inputs for the machine learning model to generate an accurate risk prediction tailored to bug type specifications.



**Fig 6: Prediction of bug report type**

## Nature-Based Prediction Model of Bug Reports Based on Ensemble Machine Learning Model



The screenshot shows a web browser window with the address bar displaying 'localhost / 127.0.0.1 | phpMyAdmin x Service Provider' and the URL '127.0.0.1:8000/View\_Prediction\_Of\_Bug\_Report\_Type/'. The main content area features a table titled 'View Bug Report Prediction Type Details !!!'. The table has 8 columns: bug\_id, creation\_date, component\_name, product\_name, short\_description, long\_description, reporter\_name, and a final column with initials. The table contains 8 rows of data, each representing a different bug report.

bug_id	creation_date	component_name	product_name	short_description	long_description	reporter_name	
XORG-2701	11-03-05	Driver/Intel	XORG	Switch to console or other session	"Using the current CVS sources for the	sndirsch	fb
LIBREOFFICE-69169	10-09-13	Presentation	LIBREOFFICE	SLIDESHOW: Crash when using some	"Libreoffice crashes when I press F5.	wmzhere	fb
TELEPATHY-27967	04-05-10	tp-glib	TELEPATHY	telepathy-glib master fails	"Subject says it all see the referenced	smcv	fb
MESA-42069	20-10-11	Drivers/Gallium/r300	MESA	Texture artifacts	"Created attachment 52603	cand	fb
TELEPATHY-17072	11-08-08	haze	TELEPATHY	Remove snpp from haze.manager	haze.manager is made on my system which	will	fb
LIBREOFFICE-82128	04-08-22	Spreadsheet	LIBREOFFICE	FILESAVE: Select Protected/Unprotecte	"Problem description: The 2	jojo	fb
FREEDESKTOP.ORG-24230	30-09-2009	New Accounts	FREEDESKTOP.ORG	New account request for mesa/r300/r600	"Created attachment 29953	amaasikas	fb
FREEDESKTOP.ORG-24230	30-09-2009	New Accounts	FREEDESKTOP.ORG	New account request for mesa/r300/r600	"Created attachment 29953	amaasikas	fb

**Fig 7: View Bug Report Type**

The service provider or Admin can view all the history of the bug details the users have predicted using the model. They can view the details about the bug users have entered and also view the predicted nature of the model.

## APPENDIX II – SAMPLE CODE

### Manage.py

```
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys

def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE',
'nature_based_prediction_model.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)

if __name__ == '__main__':
    main()
```