JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

BUG SEVERITY PREDICTION USINGDEEP LEARNING

[1] Prof. Chetna Patil, [2] Kalyani Javlekar, [4] Namrata Mandhare, [4]Gauri Shelke,

[1] Professor, [2 3 4] Student,

[1234]Department of Computer Engineering, Shivajirao S. Jondhale College of Engineering, Dombivli, Maharashtra, India.

Abstract: *Software maintenance is an essential phase of software development. Developers employ issue tracking systems to collect bugs for software improvement. Users submit bugs through such issue tracking systems and decide the severity of reported bugs. The severity is an important attribute of a bug that decides how quickly it should be solved. It helps developers to solve important bugs on time. Organizations are spending lots of time and money on testing the product. It takes plenty of time for an individual to go through all testing reports and decide the Severity of the Bug.*

Large systems may have loads of errors and bugs. Every bug needs to be evaluated, monitored, and prioritized for debugging. Our proposed system will significantly reduce the time required and streamline the process. This system can be used to track bugs during a project and automate the process of deciding the severity of bugs. Small Developer Teams can use this application to manage bugs in their project. More time can be spent testing the application and making quick releases while fixing major bugs on time.

Keywords- Bug tracking system, Machine learning, Severity prediction, KNN algorithm, Naive Bayes classifier

I. INTRODUCTION

A bug tracking system is a software application that allows software development teams to keep track of issues, or bugs, that are found during the development and testing process. These systems typically allow team members to report and track bugs, assign them to specific team members for resolution, and keep track of their progress until they are fixed.

Software maintenance is an essential phase of software development. One of the most important tasks of the maintenance process is bug resolution, where bug reports represent the bugs that users encountered while using released software systems. Developers are interested to collect such bugs for the improvement of software systems. They utilize bug tracking systems that help developers to manage bug reporting and bug triaging. Users often submit bug reports through issue tracking systems. A bug report describes the situation under which a software bug occurred and contains information of bug regeneration. A typical bug report contains multiple attributes: bug-id, submission date, status, priority, severity, summary, and description. However, the severity is an important attribute of a bug report that decides how quickly it should be resolved. severity of a bug report may varies from trivial, minor, normal, major, critical to blocker.

To implement severity prediction in a bug tracking system, data about past bugs and their severity levels are collected and used to train machine learning models. These models can then be used to predict the severity of new bugs based on various factors, such as the type of bug, the affected software component, and the circumstances under which the bug was discovered.

II. LITERATURE SURVEY

Bug reports are generally classified as critical, major, minor, and trivial bugs in which critical bugs are more severe and trivial bugs are just inconveniencing to users. To prioritize the severity of the bug reports, several machine learning approaches have been proposed. Most of the studies focus on traditional classification algorithms [1].

This research aims to provide a solution to maintaining software systems and contribute to this context by applying natural language processing (NLP), machine learning (ML), and text mining techniques to predict the bug types automatically. Once the model identifies the bug type, it accelerates the maintenance phase with bug localization techniques rather than doing this timeconsuming task manually [2].

Researchers have addressed various aspects of automated software bug management, classification and prioritization. These include automation of bug assignment, duplicate or similar bug detection, bug fixing time prediction, bug localization, bug categorization, bug severity and priority predictions etc. [3].

In real world software development, this progress is most prevalent in the application of ante-mortem approaches that aim at preventing the introduction of bugs, as for example bug prediction, static checking, and automated testing [4].

This paper aims to present an automated heuristic approach combined with fuzzy multi-criteria decision-making for bug triaging. To date, studies lack consideration of multi-criteria inputs to gather decisive and explicit knowledge of bug reports. The proposed approach builds a bug priority queue using the multi-criteria fuzzy Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) method and combines it with Bacterial Foraging Optimization Algorithm (BFOA) and Bar Systems (BAR) optimization to select developers. A relative threshold value is computed, and categorization of developers is performed using hybrid optimization techniques to make a distinction between active, inactive, or new developers for bug allocation [5].

Bug tracking systems use repositories to keep track of the bugs to improve software quality. A manual analysis of each bug and classifying it according to its severity is an unmanageable job. Text mining techniques have the potential to analyze such massive databases of the textual description of bug reports to classify bug severity levels adequately. Word embedding is a state-of-art text mining technique that captures the semantics of the text and group the words according to their relevance in a document. Words are embedded into real-valued vectors through word-embedding modelsWord2vec is a word embedding model, proven to be effective in representing word meanings [6].

III. Existing System

There are various approaches to guarantee quality in programming, for example, code surveys and thorough testing with the goal that bugs can be expelled as ahead of schedule as conceivable to avoid the misfortune it might cause. Programming bug is generally used to portray the event of a shortcoming in a product framework which results it to act uniquely in contrast to its determination bug following frameworks are one of the significant stores among all accessible programming storehouses. Many open-source programming tool have an open bug vault that permits the two engineers and clients to submit imperfections or issues in the product, propose potential improve and remark on existing bug reports.

Furthermore, the rapid proliferation of NFTs in the marketplace can sometimes lead to misinformation and misrepresented assets, causing confusion and mistrust among users. The sheer volume of NFTs being created and traded can make it challenging to verify the authenticity and value of each digital asset, leading to concerns about trust and credibility within the marketplace.

Disadvantages:

1. Another issue basic to these frameworks is that the majority of the information is unstructured. Explicit to [PITS] is that the database fields in [PITS] continue changing, yet the idea of the unstructured content stays steady.

2. Many open-source programming ventures have an open bug archive that enables the two designers and clients to submit imperfections or issues in the product, recommend potential upgrades, and remark on existing bug reports.

3. As contrasted with more up to date frameworks, the issue with [PITS] is that there is an absence of consistency in how every one of the undertakings gathered issue information.

IV. Problem Statement And Objectives

To build a Web application that can be used on an enterprise level to track the bug and using Machine Learning algorithms to determine the most accurate algorithm from the results, based on different mathematical factors and predict bug severity. Bug reports are very different from such showcase NLP problems. The information required for correct classification may not be contained in a bug report: Bug reports are textual descriptions of complicated behaviors, states, and outcomes to communicate a problem in a complex system. Such reports are authored by people occupying different roles, functions, and positions in relation to the project.

Adverting to word2vec as a word-embedding technique to capture the semantics and context of the words, and bugs as the defects or problems raised by users in natural language, (Walden et al., 2014) performed an empirical analysis to compare relative performances of text mining-based techniques and software metrics-based mechanisms to predict vulnerabilities and concluded the preponderance of text mining-based techniques in terms of better recall values.

Objectives:

1. To provide a structured process for reporting bugs in software.
2. To provide a centralized location for tracking bugs throughout the software development lifecycle.
3. To prioritize bugs based on their severity and impact on the software’s functionality.
4. To provide insights into the frequency and patterns of bugs that are discovered.

V. PROPOSED SYSTEM

A. INTRODUCTION:

This diagram offers a high-level view of a web application bug tracking system that incorporates machine learning for severity prediction. Administrators upload bug datasets, which the system then processes. This processing involves several steps including data pre-processing, feature extraction, and TF-IDF scoring/ranking. A core function is the prediction of bug severity using a Hybrid KNN-NBM algorithm. Finally, the system analyzes its own performance.

From a user perspective, the system allows them to add bugs and view the predicted severity level. This DFD provides a valuable starting point for understanding the core functionalities, but additional details would be needed for a complete picture. For instance, the DFD doesn't show how the system handles different data formats or how users interact with the system to add bugs and view predictions.

There are use two types of Algorithms as following:

KNN: K closest neighbors is a straightforward calculation that stores every single accessible case and characterizes new cases dependent on a closeness measure (e.g., separation capacities). KNN has been utilized in measurable estimation and example acknowledgment as of now in the start of 1970's as a non-parametric strategy. A case is classified by a majority vote of its neighbors, with the case being assigned to the class most common amongst its K nearest neighbors measured by a distance function, If K = 1, then the case is simply assigned to the class of its nearest neighbor as shown

$$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$ Euclidean

$$\sum_{i=1}^{k}|x_i - y_i|$$ Manhattan

$$\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$$ Minkowski

NBM: Bayes theorem provides a way of calculating the posterior probability, P(c—x), from P(c), P(x), and P(x—c). Naive Bayes classifier assume that the effect of the value of a predictor (x) on a given class (c) is independent of the values of other predictors. This assumption is called class conditional independence.

$$P(C_k|X) = \frac{P(X|C_k) \cdot P(C_k)}{P(X)}$$

where: P(Ck|X) is the posterior probability of class Ck given the input X,P(X|Ck)

is the likelihood of observing input X given class Ck, P(Ck) is the prior

probability of class Ck,

P(X) is the probability of observing input X (evidence)

B. ARCHITETURE:

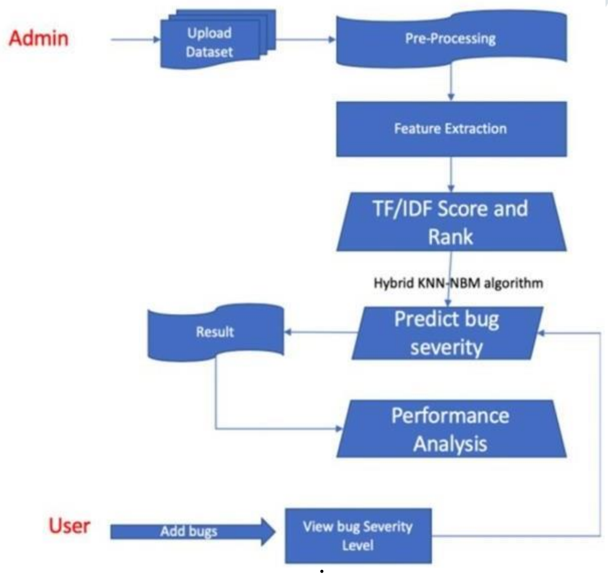


Fig.1. Architecture of Proposed System

C. DATA FLOW DIAGRAM:

I. Level 0 DFD:

This DFD provides a blueprint for the core functionalities of the bug tracking system. Our focus would be on building the data structures to store project information, team members, and bug details (including severity level and status). The interaction points between the system and the users (project manager and team members) would need to be translated into user interfaces (UI) most likely involving forms and menus for adding team members, reporting bugs, and updating bug information. This DFD would be a starting point to define the programming tasks needed to implement these functionalities and data management aspects of the application.

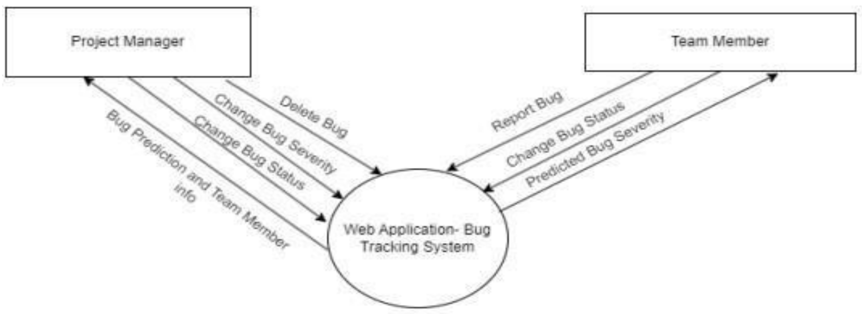


Fig 2. DFD Level 0

II. Level 1 DFD:

This level 1 DFD outlines a web application bug tracking system's core function of adding team members. Project managers input team member data, which the system stores in a dedicated database. While this DFD focuses on adding team members, the system likely allows additional functionalities like reporting bugs and updating their status, which would be captured in separateDFDs.

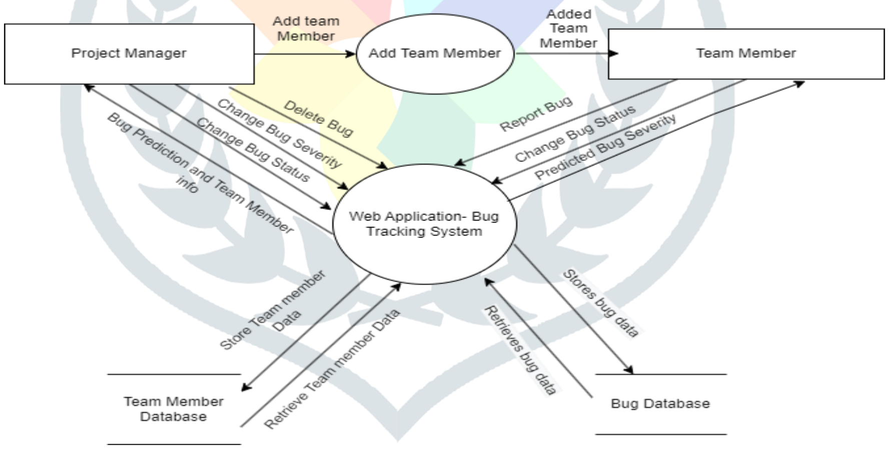


Fig 3. Level 1 DFD **III.**

Level 2 DFD:

This level 2 DFD delves deeper into the "Change Bug Status" process from level 1 by showing how the system interacts with the Bug Database and a Severity Prediction Model. A team member can initiate the process by reporting a bug, which triggers the system to retrieve bug data and predicted severity from the database and model, respectively. The team member can then change the bug status, and the system updates the bug data in the Bug Database.

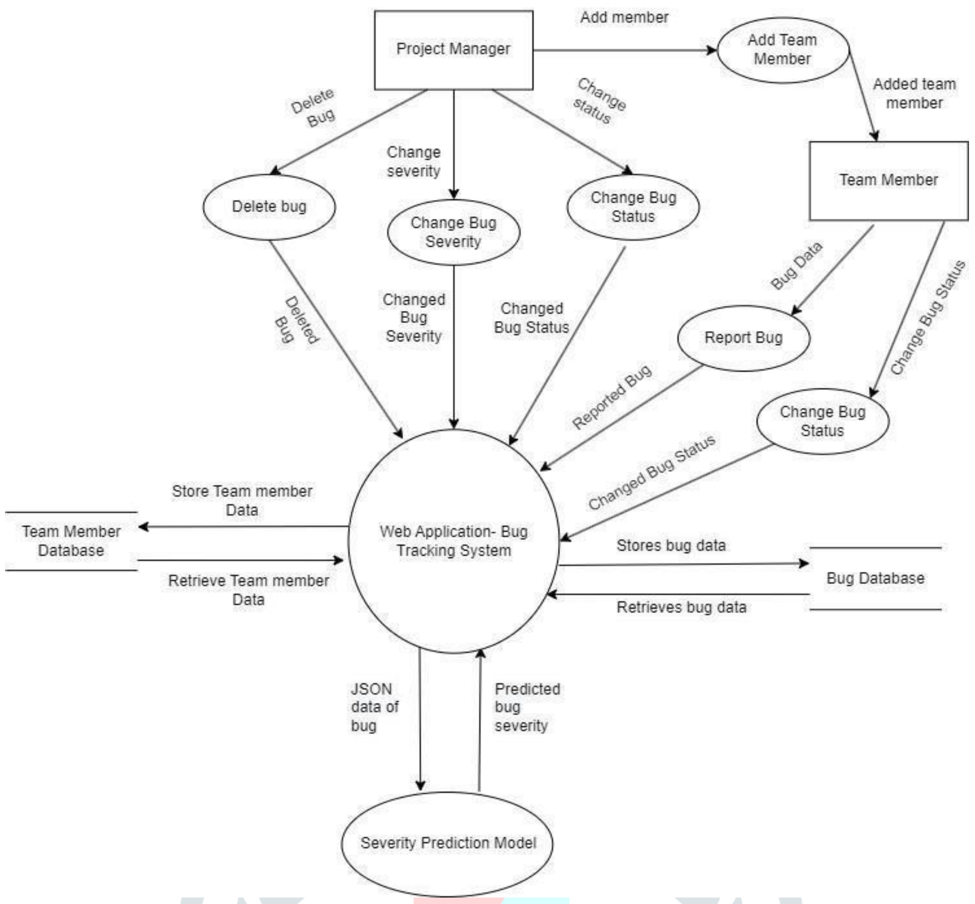
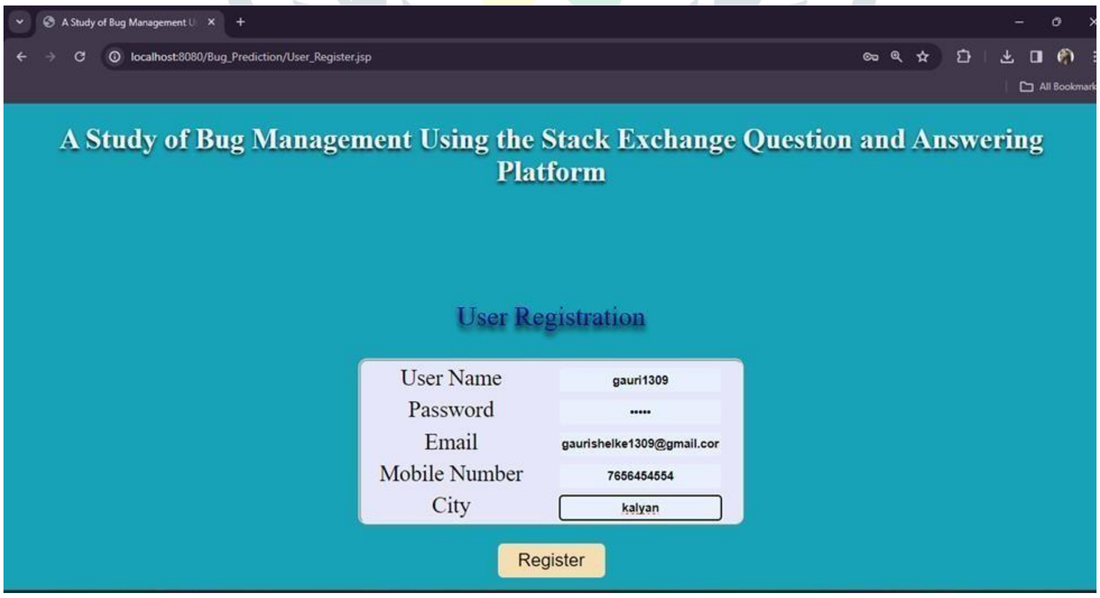


Fig 4. Level 2 DFD

VI. RESULT


A Study of Bug Management U
localhost:8080/Bug_Prediction/User_Register.jsp
All Bookmark

A Study of Bug Management Using the Stack Exchange Question and Answering Platform

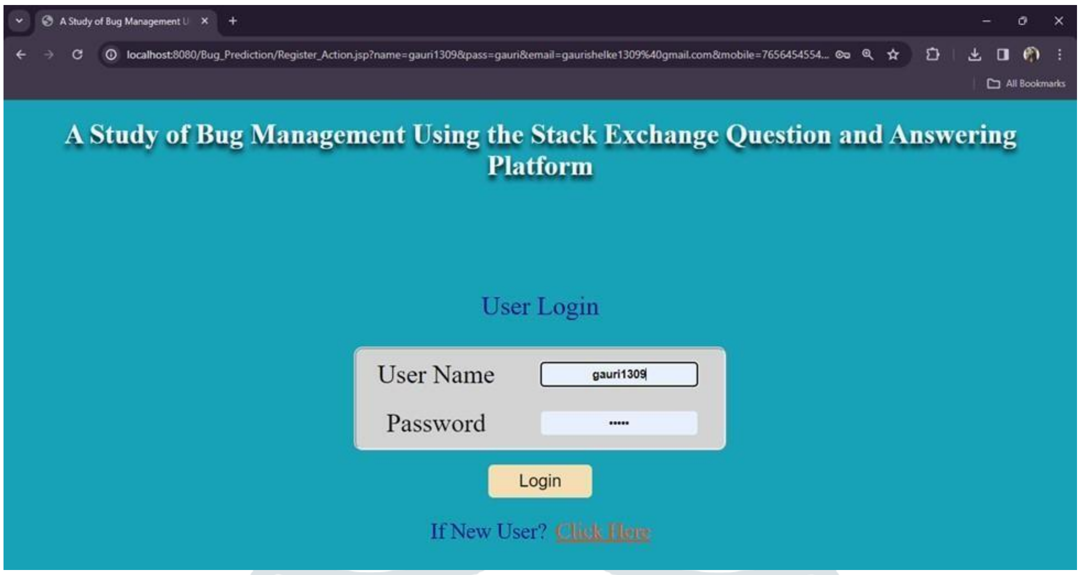
User Registration

User Name	gauri1309
Password	•••••
Email	gaurishelke1309@gmail.cor
Mobile Number	7656454554
City	kalyan

Register

Fig.1. User Registration

Any user can register their credentials to create an account on the platform. The registration form typically includes fields such as username, email address, password, and possibly additional information depending on the platform's requirements.


A Study of Bug Management U
localhost:8080/Bug_Prediction/Register_Action.jsp?name=gauri1309&pass=gauri&email=gaurishelke1309%40gmail.com&mobile=7656454554...
All Bookmarks

A Study of Bug Management Using the Stack Exchange Question and Answering Platform

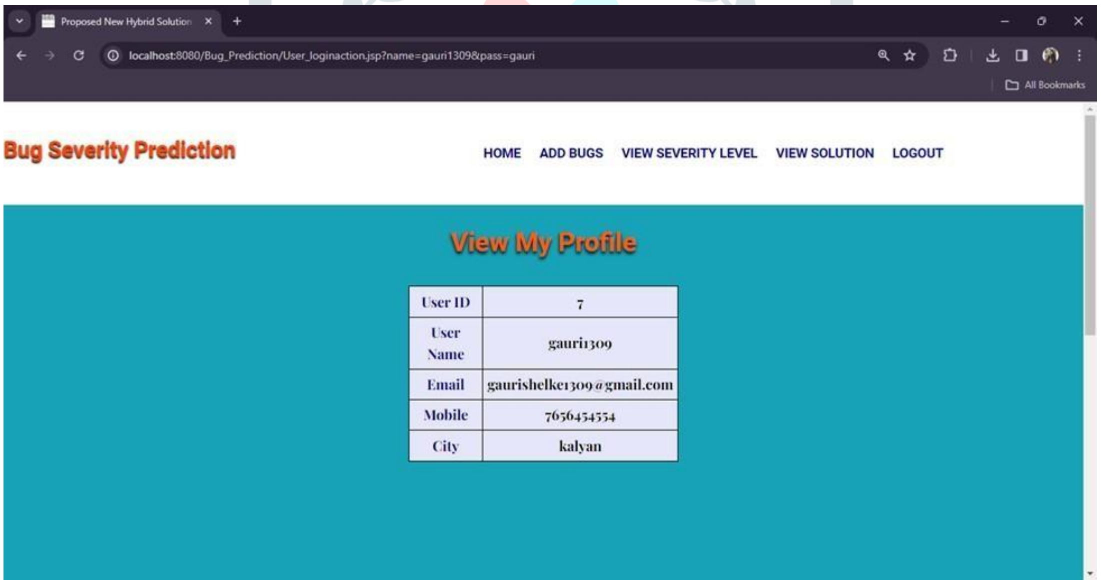
User Login

User Name	gauri1309
Password	•••••

Login
If New User? Click Here

Fig.2. User Login

Users can log in to their accounts by entering their username and password. Once entered, the platform verifies the credentials against its database.

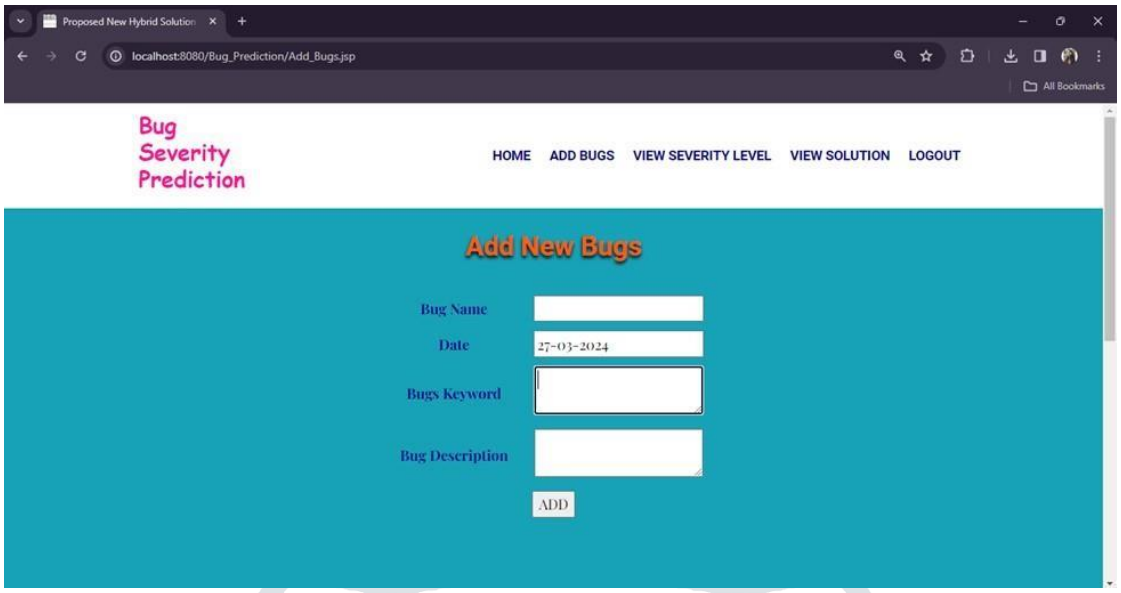

Proposed New Hybrid Solution
localhost:8080/Bug_Prediction/User_loginaction.jsp?name=gauri1309&pass=gauri
All Bookmarks
Bug Severity Prediction
HOME
ADD BUGS
VIEW SEVERITY LEVEL
VIEW SOLUTION
LOGOUT

View My Profile

User ID	7
User Name	gauri1309
Email	gaurishelke1309@gmail.com
Mobile	7656454554
City	kalyan

Fig.3. View User Profile

Users can view their created profile, which displays all the details they entered during registration. This functionality allows usersto review and verify the information associated with their account.

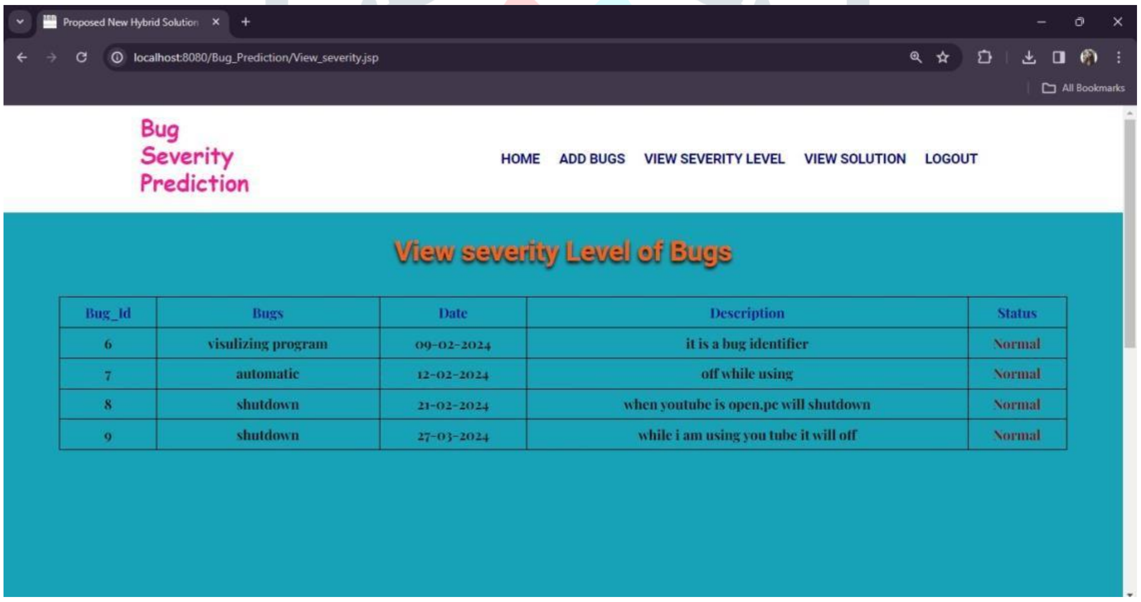


Add New Bugs

Bug Name	
Date	27-03-2024
Bugs Keyword	
Bug Description	

Fig.4. Add Bug

Users can add newly detected bugs to the database along with their corresponding severity levels. By adding new bugs to the database, users contribute to enriching the platform's repository of information, enhancing its effectiveness in bug tracking and resolution.



View severity Level of Bugs

Bug_Id	Bugs	Date	Description	Status
6	visulizing program	09-02-2024	it is a bug identifier	Normal
7	automatic	12-02-2024	off while using	Normal
8	shutdown	21-02-2024	when youtube is open.pc will shutdown	Normal
9	shutdown	27-03-2024	while i am using you tube it will off	Normal

Fig. View Severity Level

Upon entering a bug into the tool, the severity of the bug is automatically detected and displayed. This feature enables users to promptly ascertain the severity level of the bug as soon as it is logged into the system.

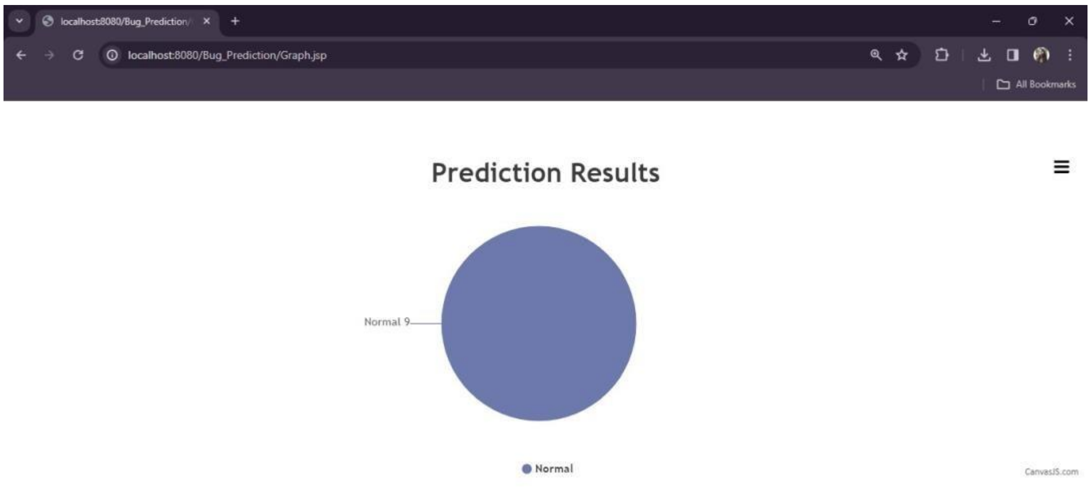


Fig Prediction Results

The output of our tool displays the severity level of the collected bug set. Prediction results are categorized into various levels suchas critical, enhancement, trivial, major, minor, and normal,

VII. Conclusion And Future Work

Team members can successfully add a bug into the system. Admin can assign a bug to a team member. Admin can delete a bug and can update the status of an existing bug. Admin can also delete an existing project or a user who is assigned to a project. Team members can change their profiles, change the status of an existing bug. The future scope of this project can include that the system can be automated further to reduce the manual effort required in bug tracking and severity prediction. This can be achieved by using machine learning algorithms and natural language processing techniques. This paper proposed a nature-based bug prediction component using an ensemble machine learning algorithm that consists of four base machine learning algorithms, Random Forest, Support Vector Classification, Logistic Regression, and Multinomial Naïve Bayes.

Bug reports play a crucial role in software development and maintenance activities. They allow software developers quality assurance team, and customers to identify and report related issues. These reports entail extensive details, hence manual extraction of this information is infeasible due to large time complexity.

VIII. References

[1]. Xu Cheng Yin, Chao Zhu, Qasim Umer, Waheed Yousuf Ramay, Inam Illahi, “Deep Neural Network-Based SeverityPrediction of Bug”, Feb 2019.
[2]. Shatha Abed Alsaedi, Amin Yousef Noaman, Ahmed A. A. Gad-Elrab, Fathy Elbouraey Eassa, “Nature-Based PredictionModel of Bug”, Dec 2023.
[3]. Hafiza Anisa Ahmed, Haemeen Zakaria Bawany, Jawwad Ahmed Shamsi, “CaPBug-A Framework for Automatic Bug Categorization and Prioritization Using NLP”, March 2021.
[4]. Thomas Hirsch, Birgit Hofer, “Using textual bug reports to predict the fault category of software bugs”, May 2022. [5]. Chetna Gupta, Pedro R.M. Inácio , Mário M Freire, “Improving software maintenance with improved bug triaging”, Feb
2022.
[6]. Thomas Hirsch, Birgit Hofer, “Using textual bug reports to predict the fault category of software bugs”, Dec 2021.