**Objective**

Nowadays, everyone is going online, so if there is a problem, everyone needs an online application to get the solution right away, and everyone can easily access the software through the internet. This project is also available online for convenience. This system will be beneficial for all users to have access to. This Bug Tracking system is primarily used to identify project bugs. For example, stack overflow is quite popular these days because all the developers, students, and everyone will get the solution from that site, and now most of the people use it. Similarly, this project will provide a solution to the problem. This project will be worked on internally and will include three modules: tester, developer, and administrator. The tester will be the person who can report bugs in the project. The tester must be aware of the entire project flow and must prepare a test report. The test report will be updated to the developer, who will be able to monitor the bugs and be the authorised person to put the solutions in place. The administrator will be the only person authorised to monitor the developer's and tester's activities. And if anyone requires the bug report, they must request it from the administrator.

**Chapter 1**

**INTRODUCTION**

**1.1 Project Description**

Throughout this prototype, we are developing a Software platform that is elevated, extensible, and secure. We created three modules in this: admin, developer, and tester. The admin module manages all of the users' data, allows them to view the solution, and modifies all of the users' data. Developer modules for adding projects, finding solutions, and tracking bugs. Similarly, the tester module tracks bugs, adds projects, and finds solutions. All data is saved in Mysql by default, and PHP provides a database connection. And we are using the http request method with cross-site request forgery (csrf token) to protect the data. We also have the bug modules. When we post the project, we will include a project description, such as whether the bug is in the front end or the back end. Based on that, we can track bugs and find solutions for bugs in the front end or backend. We are providing authentication to admin in order to determine whether or not the users are valid. If we know whether the user is valid or invalid, we will be able to determine whether the user is a valid or invalid person. After that, we can easily track the bug based on the project and find a solution.

Current technologies are not only becoming more sophisticated and large, but they are also being updated regularly under the agile approach. It is common for a software to have tens of components created by a large number of developers. As a result, it is difficult to guarantee that every version of a high-end programme is bug-free.

Even though the usage of defect monitoring systems greatly aids the entire process of problem reporting to bug resolving, there are still certain issues to overcome in terms of identical bug record identification. Bug reporting systems aid in the collecting of bug reports, however they no longer allow automatic duplication identification. As a result, the triager must manually analyse each bug report to determine whether or not it is a duplicate. This approach not only takes time and is prone to errors, but it also raises the cost of software maintenance. Furthermore, because there are so many duplicate bug reports, it is impracticable for a triager to personally handle by them. As a consequence, automatic identification of identical bug reports is critical.

In order to achieve this goal, some studies are devoted to automating the detection process, which mainly include two ways. The first one is employing the traditional NaturalLanguage Processing (NLP) and Information Retrieval techniques to help with duplicate detection. However, in such a way, when a new bug report is received, it needs to choose key words to search in bug tracking systems. In way to attain this aim, several research are being conducted to automate the detection procedure, which consists mostly of two methods. It should be to use standard Natural Language Processing (NLP) and Information Extraction methods to aid with duplication identification. However, when an unique defect record is generated, it must select key phrases to scan in defect monitoring systems. The alternative is to do categorization by using a criteria to evaluate whether or not the newly-reported document is identical. Since these techniques can help relieve some of the strain on triagers, overall precision of duplication identification seems far from acceptable. However, neither of the above techniques completely accounts for the spatial data in defect record summaries.

**Limitation**

Whenever an issue is identified, numerous actions such as research, implementation, test, and troubleshooting are performed across the process. Because numerous flaws are frequently discovered, managing problems with a project for recording every finding in documents is fairly inconvenient. While developers are the ones who generate bugs, the QA team will be the one who analyzes the program and software to determine the specific process or collection of procedures that results an issue.

The communication between the system QA team and the system developers will have to be simplified. According to research, attempting to keep these bugs in your head or in a single document such as excel spreadsheet results in instant catastrophe because you won't be able to have a conversation with each other or with the event team and thus won't be prepared to contribute to increasing product quality. As a project expands, the most common issue is that only one person may make changes to the document at a time. He also believes that information is an essential factor in the success of a software development. Despite of how technically proficient the members of the team are, if communication breaks down, the entire team may fail.

This section we are going to see about existing system and proposed system of bug tracking project.

**2.1.1 Existing System**

In the current structure, project managers delegate project-related tasks to developers. The developers' work must be centred on client pleasure. In the project, the project manager allocates all tasks to developers and testers. If a project fails to function correctly during testing, it is tough to bridge with the programmer, and the project is then assigned to the tester department. It takes a long time to trace out the issue and discover remedies in this case.

The drawbacks of the current method are as follows:

• It takes a long time to transfer the project developer department to the tester department;

• Some errors may arise during the transition.

Manually, the project manager goes over all of the projects involving by the developers and testers.

**2.1.2 Proposed System**

Web apps are used to build web applications in the suggested system. We are developing three modules: admin, developer, and tester. Admin module for managing all developers and users, assigning projects, tracking issues, and finding solutions. Developer module for adding projects, viewing projects, finding issues, and adding solutions. Tester module to add projects linked to testing data bugs and discover solutions for specific projects. As a result, it is the simplest approach to trace down the issue and discover a remedy.

## The benefits of the suggested system are as follows:

## • Less work time

## • Less manual labour

## • Data protection

## • Efficient method

## • User friendliness

**Literature Survey**

**3.1 Towards Effective Troubleshooting with Data Truncation**

Towards Effective Troubleshooting with Data Truncation deals with reducing the data present in the bug repository and improve the quality of data then reduce time and cost of bug triaging, it represent an automatic approach to predict a developer with relevant experience to solve the new coming report.. The bug data sets are obtained and techniques such as instance selection feature selection are applied simultaneously. The top k pruning is applied for improving results of data reduction quality, obtaining domain wise bug solution. Instance selection is for obtaining a subset of relevant instances (i.e., bug reports in bug data) .It is used to Remove noise and redundant instances,Remove non-representative instances. Feature selection which aims to obtain a subset of relevant features (i.e., words in bug data) ,Sorting of words according to feature values .Top-K pruning algorithm for improving results of data reduction quality.

**3.2 Technique to Combine Feature Selection with Instance Selection for Effective Bug triage**

A Technique to Combine Feature Selection with Instance Selection for Effective Bug Triage .It addresses the issue of data reduction for bug triage by text classification techniques. Conventional software analysis is not totally suitable for the large-scale and complex data in software repositories. Data mining has developed as a promising means to handle software data. There are two difficulties related to bug data that may influence the effective use of bug repositories in software development tasks, namely the huge scale and the low quality. Therefore unfixed bugs are deleted from the bug repositories.

**3.3 Automatic Bug Triage using SemiSupervised Text Classification Automatic Bug Triage using Semi-Supervised**

Text Classification propose a semi-supervised text classification approach for bug triage to avoid the deficiency of labeled bug reports in existing supervised approaches. This approach combines naive bayes classifier and expectation maximization to take advantage of both labeled and unlabeled bug reports. Then the approach iteratively labels numerous unlabeled bug reports and trains a new classifier with labels of all the bug reports. Then it employs a weighted recommendation list to boost the performance by imposing the weights of multiple developers in training the classifier. Before training a supervised classifier for bug triage, a necessary step is to collect numerous labeled bug reports, which are bug reports marked with their relevant developers.The semisupervised text classification approach to improve the classification accuracy of bug triage. This semisupervised approach enhances a NB classifier by applying expectation-maximization (EM) based on the combination of unlabeled and labeled bug reports. First, this semi-supervised approach trains a classifier with labeled bug reports. Then, the approach iteratively labels the unlabeled bug reports and trains a new classifier with labels of all the bug reports. To adjust bug triage, we update a semi-supervised approach with a weighted recommendation list (WRL) to augment the effectiveness of unlabeled bug reports. This WRL is employed to probabilistically label an unlabeled bug report with multiple relevant developers instead of a single relevant developer.

**3.4 Reducing Features to Improve Bug Prediction**

Recently, machine learning classifiers have emerged as a way to predict the existence of a bug in a change made to a source code file. The classifier is first trained on software history data, and then used to predict bugs. Two drawbacks of existing classifier-based bug prediction are potentially insufficient accuracy for practical use, and use of a large number of features. These large numbers of features adversely impact scalability and accuracy of the approach. Reducing Features to Improve Bug Prediction aims in classifier to first trained on software history data, and then used to predict bugs. The disadvantage of the traditional method is that, classifier-based bug predictions are potentially insufficient accuracy for practical use, and use of a large number of features. The system uses Naive Bayes and Support Vector Machine (SVM).The system mainly Gain Ratio for feature selection, along with the characterization of bug prediction results achieved when using feature selection. This paper proposes a feature selection technique applicable to classification-based bug prediction. This technique is applied to predict bugs in software changes, and performance of Naıve Bayes and Support Vector Machine classifiers is characterized. . These features include everything separated by whitespace, in the code added or deleted in a change. This leads to a large number of features, in the thousands, and low tens of thousands. For larger project histories which span thousand revisions or more, this can stretch into hundreds of thousands of features. The addition of many non-useful features reduces a classifier’s accuracy. Additionally, the time required to perform classification increases with the number of features, rising to several seconds per classification for tens of thousands of features, and minutes for large project histories. A standardapproach (in the machine learning literature) for handling large feature sets is to perform a feature selection process to identify that subset of features providing the best classification results. This paper introduces a feature selection process that discards features with lowest gain ratio until optimal classification performance is reached for a given performance measure.

**3.5 Efficient Bug Triaging Using Text Mining**

Efficient Bug Triaging Using Text Mining aims for an automatic approach to predict a developer with relevant experience to solve the new coming report. The techniques used are five term selection method. Term selection methods are used to reduce the high dimensionality of term space by selecting the most discriminating terms for the classification task. The methods give a weight for each term in which terms with higher weights are assumed to contribute more for the classification task than terms with lower weights. The goal of bug triaging is to assign potentially experienced developers to new-coming bug reports. To reduce time and cost of bug triaging, we present an automatic approach to predict a developer with relevant experience to solve the new coming report. It investigate the use of five term selection methods on the accuracy of bug assignment. In addition, it re-balance the load between developers based on their experience. It conduct experiments on four real datasets. To reduce the time spent triaging, it present an approach for automatic triaging by recommending one experienced developer for each new bug report. This information can help to manage the progress of these projects. In the last decade, practitioners have analyzed and mined these software repositories to support software development and evolution. One of the important software repositories is the bug tracking system (BTS). Many open source software projects have an open bug repository that allows both developers and users to submit defects or issues in the software, suggest possible enhancements, and comment on existing bug reports. It formulate the bug triaging process as a classification task where instances represent bug reports, features represent the terms of the report, and the class label represents the developer who fixed this report. This approach can help the triage process in two ways: 1) it may allow a triager to process a bug more quickly, and 2) it may allow a triager with less knowledge about systems and developers to perform bug assignments more accurately.

**References**

[1] B Jifeng Xuan, He Jiang, Yan Hu, Zhilei Ren, Weiqin Zou, Zhongxuan Luo, and Xindong Wu,” Towards Effective Bug Triage with Software Data Reduction Techniques” ieee transactions on knowledge and data engineering, vol. 27, no. 1, january 2015.

[2] karishmaMusale, Gorakshanath Gagare, “Towards Effective Troubleshooting With Data Truncation” , “International Journal of Advanced Research in Computer and Communication EngineeringVol. 4, Issue 11, November 2015”

[3] Ashwini Jadhava, Komal Jadhava, “A Survey on Software Data Reduction Techniques for Effective Bug Triaging”, “International Journal of Computer Science and Information Technologies, Vol. 6 (5) , 2015, 4611-4612”.

[4] JifengXuan, He Jiang, “Automatic Bug Triage using Semi-Supervised Text Classification”, “Chinese Academy of Sciences, Beijing, 100190 China”.

[5] Suvarnaa Kale, Ajay Kumar Gupta, “A Technique to Combine Feature Selection with Instance Selection for Effective Bug Triage”, “International Journal of Science and Research (IJSR) ISSN (Online): 2319- 7064”.

[6] C. Sun, D. Lo, S. C. Khoo, and J. Jiang, “Towards more accurate retrieval of duplicate bug reports,” in Proc. 26th IEEE/ACM Int. Conf. Automated Softw. Eng.,2012

[7] J. Anvik, L. Hiew, and G. C. Murphy, “Who should fix this bug?,” Proc. Intl. Conf. Softwarre Engineering (ICSE 06), ACM, 2006, pp. 361-370.

[8] N. Betten burg, R. Prem raj, T. Zimmermann, and S. Kim, “Reducing Features to Improve Bug Prediction,” Proc. IEEE Conf. Software Maintenance (ICSM 08), IEEE Computer Society, Sep. 2008, pp. 337-345.

[9] J. Anvik , “Automatic bug reporting assignment, ” in Proc 28th International Conference on Software Engineering. ACM, 2006, pp. 937–940.

[10] J. Anvik , and G. C. Murphy , “Reducing the effort of bug report triage: Recommenders for development-oriented decisions,” ACMTrans. Soft. Eng. Methodol., vol. 20, no. 3, article 10, Aug. 2011.