Prediction of the Patch Release Date

Submitted By

Priya Vasu

16MCEC29

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

May 2018

Prediction of the Patch Release Date

Major Project

Submitted in fulfillment of the requirements

for the degree of

Master of Technology in Computer Science & Engineering

Submitted By

Priya Vasu

(16MCEC29)

Guided By

Prof. Monika Shah

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD-382481

May 2018

Certificate

This is to certify that the major project entitled **"Prediction of the Patch Release Date"** submitted by **Priya Vasu (16MCEC29)**, towards the partial fulfillment of the requirements for the award of degree of Master of Technology in Computer Science and Engineering of Nirma University, Ahmedabad, is the record of work carried out by her under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Prof. Monika Shah
Guide & Asst. Professor,
CSE Department,
Institute of Technology,
Nirma University, Ahmedabad.

Dr. Priyanka Sharma
Professor,
Coordinator M.Tech - CSE
Institute of Technology,
Nirma University, Ahmedabad

Dr. Sanjay Garg
Professor and Head,
CSE Department,
Institute of Technology,
Nirma University, Ahmedabad.

Dr Alka Mahajan
Director,
Institute of Technology,
Nirma University, Ahmedabad

Statement of Originality

I, **Priya Vasu**, **16MCEC29**, give undertaking that the Major Project entitled "**Prediction of the Patch Release Date**" submitted by me, towards the partial fulfillment of the requirements for the degree of Master of Technology in **Computer Science & Engineering** of Institute of Technology, Nirma University, Ahmedabad, contains no material that has been awarded for any degree or diploma in any university or school in any territory to the best of my knowledge. It is the original work carried out by me and I give assurance that no attempt of plagiarism has been made.It contains no material that is previously published or written, except where reference has been made. I understand that in the event of any similarity found subsequently with any published work or any dissertation work elsewhere; it will result in severe disciplinary action.

Signature of Student
Date:
Place:Ahmedabad

Endorsed by
Prof. Monika Shah
(Signature of Guide)

Acknowledgements

It gives me immense pleasure in expressing thanks and profound gratitude to **Prof. Monika Shah**, Asst. Professor, Computer Science & Engineering Department, Institute of Technology, Nirma University, Ahmedabad for her valuable guidance and continual encouragement throughout this work. The appreciation and continual support she has imparted has been a great motivation to me in reaching a higher goal. Her guidance has triggered and nourished my intellectual maturity that I will benefit from, for a long time to come.

It gives me an immense pleasure to thank **Dr. Sanjay Garg**, Hon'ble Head of Computer Engineering Department, Institute of Technology, Nirma University, Ahmedabad for his kind support and providing basic infrastructure and healthy research environment.

A special thank you is expressed wholeheartedly to **Dr. Alka Mahajan**, Hon'ble Director, Institute of Technology, Nirma University, Ahmedabad for the unmentionable motivation she has extended throughout course of this work.

I would also thank the Institution, all faculty members of Computer Engineering Department, Nirma University, Ahmedabad for their special attention and suggestions towards the project work.

- Priya Vasu

16MCEC29

Abstract

For a large and advancing Software System, the project group could get numerous bug reports over a long stretch of time. It is critical to accomplish a quantitative comprehension of bug fixing time. The capacity to predict bug fixing time can enable an project group better estimate programming support endeavors and better manage software programming ventures. In addition this time will be used to predict the release date of an minor version i.e Patch. In industry when a client raises any bug , the undertaking supervisor needs to give them a date till which the minor variant will be released. The procedure for estimating the release date of the version needs to go through numerous stages like bug fixing time, smoke testing time lastly regression testing time. So we exhibit an effort that consequently predicts the fixing time. Our procedure uses existing issue following frameworks i.e when a new bug report is generated the title and the description is extracted from the report that the bug with the similar title and description is searched from the database and here we have used Lucene framework for finding the bugs that have text similarity with the new bug report and used their time for prediction. In this approach, we have used Support vector Machine technique to query the database of resolved issues for textually similar reports. We also increase the reliability of our predictions by extending the SVM approach to explicitly state when there are no similar issues. Here this approach helps us for the early estimation of the bug fixing time, better assignment of the issues and predicting and scheduling the stable releases. Here we have assessed our approach utilizing the information from the RPAS (Retail Predictive Application Server) Project of an Oracle. Given an adequate number of issues reports, our programmed predictions are near the real exertion.

Abbreviations

RPAS	Retail Predective Application Server.
ADF	Application Development Framework.
CT	ConfigTool.
BR	BugReport.
BugDB	Bug Database
SVM	Support Vector Machine.

Contents

List of Figures

Chapter 1

Introduction

Prediction of when a specific programming development task will be finished has dependably been troublesome. The time it takes to fix an defect is especially challenging to forecast. Why is that so? As contrast to programming, which is a development procedure, debugging is an inquiry procedure a search which can include the greater part of the program's code, its runs, its states, or even its history. Debugging is especially terrible on the grounds that the first suspicions of the program's creators can't be trusted. Once the defect is recognized, fixing it is again a programming action, however the prior push to seek typically far exceeds the effort taken for correcting the same. However in industry predicting the release date of the minor version is extremely critical task as it is the date given to the client as an agreement so the supervisor must be twofold sure before giving the date of releasing the fix (patch) ,in light of the fact that once the date is given regardless of what happens the fix must be discharged on that date otherwise the client's trust would be lost

1.1 Introduction to area of work

Retail industry is highly dependent on consumers, their demands and their choices. Along with all the different retail outlets at different locations, from remote small towns to major cities, consumers all over the world have a huge array of options from where they can buy all their stuffs. With the advent of online shopping, their options have only increased. This has increasingly made the competition among various retailers all the more fierce, as they now need to predict the consumers future demands, all the while, being in touch with current market trends.

This means that retailers all over the world need to gauge the market trends and plan accordingly. They need to plan for the profits they may generate in the future, by analyzing the current demands of the consumers, to predict their future demands.This allows to the retailers to maintain their stock inventory, by changing supplies of different products, depending upon market trends, season and consumer interest.Optimizing their supply according to consumer demands further adds to their bottom line. Planning and optimizing helps the retailers to make profitable decisions based on future insight and allows effective collaboration among their different departments and their suppliers.Having the right localized products by planning ahead, at different local stores across the world, helps to sell more of certain products.In order to provide superior customer experience, they have to utilize all their data and plan accordingly to make price, inventory and assortment decisions.

1.2 Present Day Scenario and Proposed Approach

The quantity of bug reports in complex programming increments drastically. Since bugs are still triaged physically, bug triage or task is a work concentrated and tedious undertaking. Without learning about the structure of the product, analyzers frequently indicate the part of another bug erroneously. In the mean time, it is troublesome for triage's to decide the part of the bug just by its portrayal. For example, we uncover the parts of bugs from the RPAS bug venture, which have been determined erroneously and adjusted at any rate once, and demonstrated that these bugs must be reassigned what's more, the procedure of bug settling must be deferred. The normal time of fixing erroneously indicated bugs is longer than that of effectively indicated ones. So we have used the Below approach that works effectively.

Here, we have taken in to consideration the issue of evaluating the time it will take to solve the bug and fix the issue from a novel point of view. Our approach depends on utilizing the experience from prior issuesor, more mundane, to extract bugs reports from databases and to utilize their property to make forecasts for new, comparable issues. We have used this approach to deal with predicting the fixing effortthat is, the effort (in person-hours) it takes to settle a specific issue. These assessments are key to extend directors, since they permit to design the cost and time of future releases.

Whenever the new bug is raised ,the new issue report r is entered into the bug database

, and the following approach is used to predict the release date of the patch.

- We look for the existing issue reports which have a same title and depiction that is most like r.
- We than compute the aggregate time taken in days to fix that comparative defect.(The time can be figured from the reported date and the updated date)
- Finally we include some steady days for smoke testing and regression testing and an overhead day(in case there happens some unforeseen circumstance) and the final output will be the release date of the fix (I.e the minor version)..

Chapter 2

Literature Survey

2.1 Retail Predective Application Server (RPAS)

The RPAS is a basement for multiple optimization and planning applications. The different solutions which RPAS provides support are Advanced Inventory Planning (AIP), Merchandise Financial Planning (MFP) and Retail Demand Forecasting (RDF). RPAS is used to forecast the current market trends with the data already present. In retail industry, RPAS is utilized to gauge the present market trends and to bring in light the information definitely known. It is a stage which gives the framework expected to handle and create data in light of the info given by the retailer. It helps in arranging the stock by breaking down the client needs and conduct and limit the stock wastage because of ill-advised arranging.

With a tested adaptability for advancement of arrangements which depend on multidimensional forecasting and planning, RPAS is a configurable platform. Multidimensional structure of database, on web and batch processing, slice-and-dice UI and calculation engine which are configurable, client's protection and application capacities like uploading and exporting, are the capacities given by using RPAS platform, every certainly one of them on an environment that is fairly adaptable and specialized, that may be conveyed on range of hardware.

2.2 Introduction to BugDB

A BugDB is the tool used for tracking the bug , issue tracking and management of the project.In Oracle all the bugs are raised in the BugDB. The report created while raising

the bug is called the bug report. BugDB I.e bug database isn't an open asset - not even for clients with a help contract. Oracle Support (formally known as Meta link) gives data on bugs that are related with help tickets opened by contract holders. These bugs are found in different records all through Oracle Support, however there is no open database as exists for Open Source ventures like Apache. Just Oracle workers approach their inner bug database, and I am sure that the database is exclusive and not something like Bugzilla.

Figure 2.1 shows the main page of BugDB tool.Here on the left side we have all the functionality like Create an request for the bug i.e raise the bug, Escalate the bugs etc.

To raise an new bug in bugdb we need to select create an new option from the inbuilt functionality of the bugdb.To create a new bug report the reporter has to fill Following details in the report which is mandatory as this details stored in the database and can be used in for future purpose.The details that need to be entered are as follows:

- Bug No : Each bug in the database has an unique bug number.If the user try to enter duplicate number than the tool will give an error saying " This bug number already exists please enter some other unique identifier".
- BUG Title and Description: The title and Description of the Bug.
- Filed by and Updated By: Name of the bug filer and the one who has updated the bug.
- Reported Date and the Updated date of the bug.
- Other information like Status,Fixed version,Fix by Tag, steps to reproduce etc.

Figure 2.2 shows the RPAS bug's issue report in BugDB.At the Top we see the BUG No and the Title of the Bug. i.e BUG 27569879: Text Access keys do not wok for menus on chrome and IE11.At the bottom we have the detailed description i.e when the user press Alt+F the file menu is not selected similarly ctrl+T is not opening the new tab. Here we also have the filed and the updated date of the bug.This date will later be used to estimate the time it has taken to fix the bug.Here in this report we also have othet details like the the type of bug is internal, the status of the bug is 40 , filed by is MONISHR , fix by tag is 16.0.3.10 etc.BUGDB also allow us to query the database and to only use the details that is relevent to us.

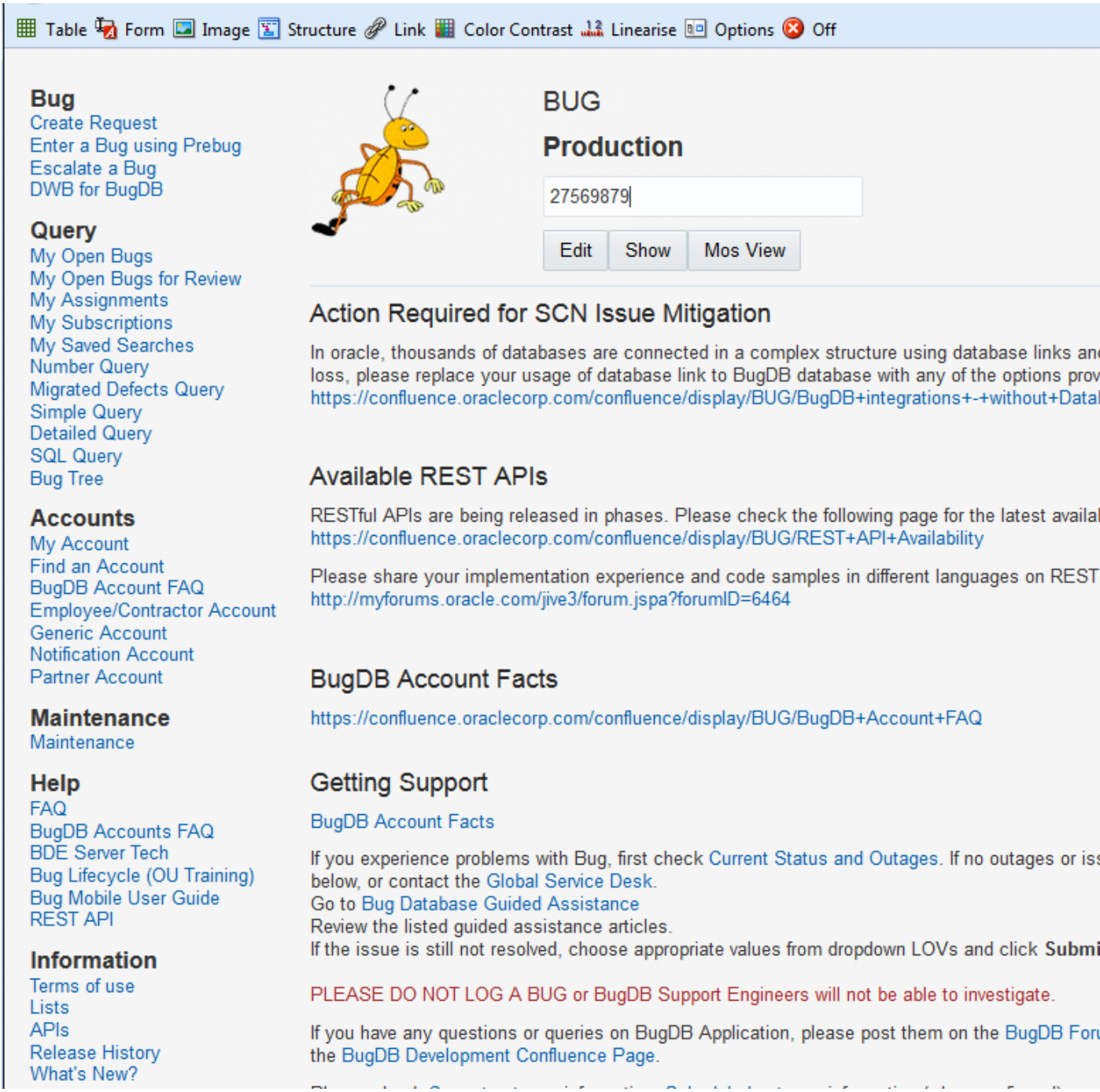


Figure 2.1: BugDB Main Page

Figure 2.3 shows the database that we have used in the this project.Here we have fire the detailed query in the bug database.We have not taken all the fields form the bug report because it is not needed in the project.

The Output of the query is the the following fields :

- Bug No : Unique identifier of the bug
- Subject:The title of the Bug

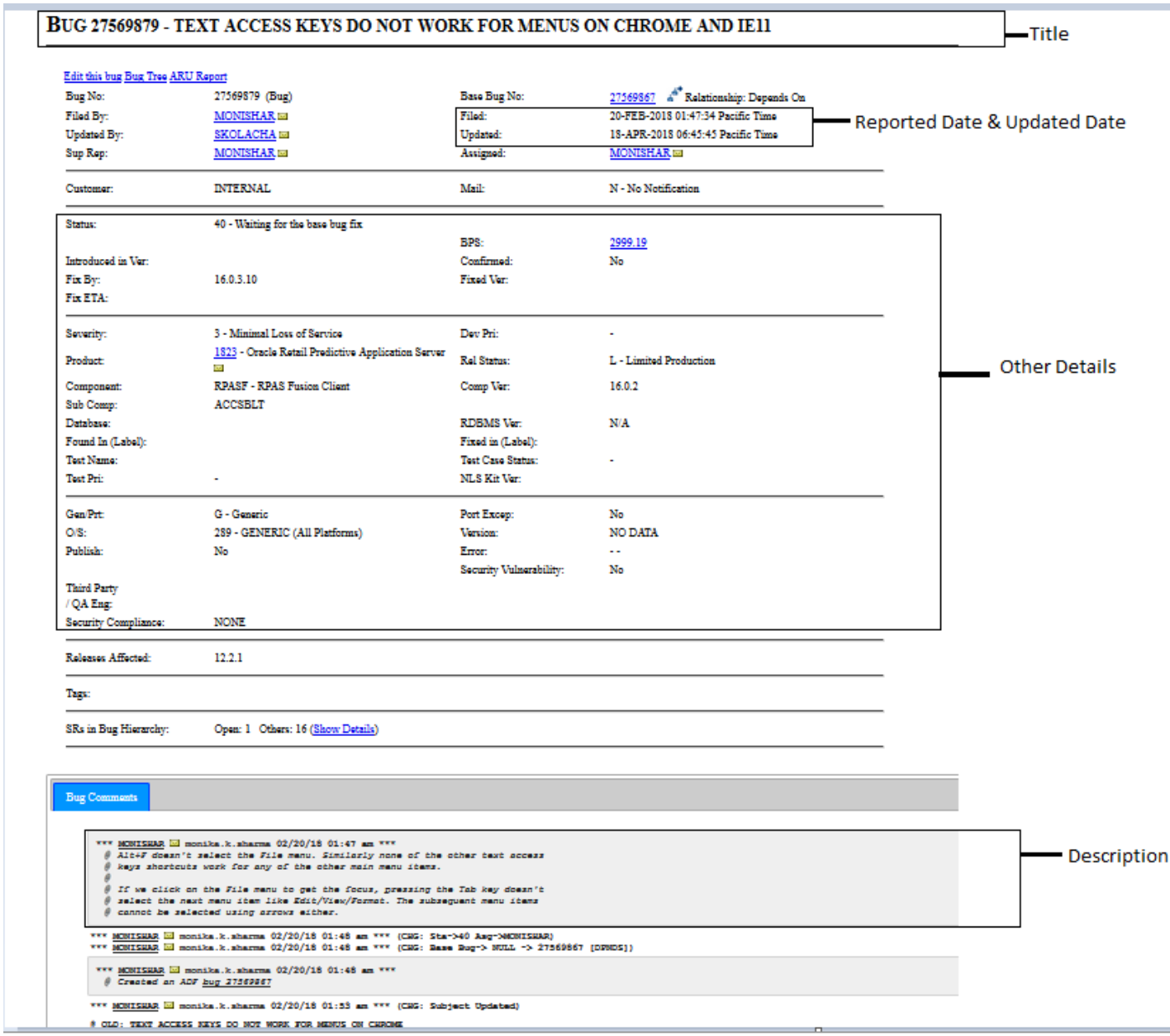


Figure 2.2: Bug Report

Sl No.		Num	Show Bug	Subject	Description	St	Sev	Assignee	Reported	Closed
1.	☑	28012078	∞	CUSTOM JAVA EXPRESSIONS ARE FAILING DUE TO BASE CODE CHANGE TO JAVA CLASS WRAPPE	AS THE BASE CODE HAS CHANGED TO THE JAVA CLASS THE CUSTOM JAVA EXPRESSION IS FAILING	11	2	LRODDENB	17-APR-18	07-MAY-18
2.	☑	28011640	∞	ALLOW DEFAULT SETTINGS FOR QUICK EXPORT BUTTON TO BE CONFIGURABLE	ALLOW DEFAULT SETTINGS FOR QUICK EXPORT BUTTON TO BE CONFIGURABLE	15	2	SKOLACHA	17-APR-18	08-MAY-18
3.	☑	28007646	∞	DYNHERREFRESH THROWS EXCEPTION INSTEAD OF RETURNING	DYNHERREFRESH THROWS EXCEPTION INSTEAD OF RETURNING	11	2	SKOLACHA	05-APR-18	06-MAY-18
4.	☑	28006410	∞	SUBLOGS FOLDER NOT GENERATED IN PCGD DOMAIN WHILE RUNNING OAT TASK	SUBLOGS FOLDER NOT GENERATED IN PCGD DOMAIN WHILE RUNNING OAT TASK	11	3	SKOLACHA	04-APR-18	13-APR-18
5.	☑	28001794	∞	ADD SUPPORT FOR GROUPS AND USERS TO COPY FORMAT TASK	ADD SUPPORT FOR GROUPS AND USERS TO COPY FORMAT TASK	15	2	SKOLACHA	04-APR-18	04-APR-18
6.	☑	27999621	∞	ERROR WHILE DOING 'SUBMIT AN ADMIN TASK'	ERROR WHILE DOING 'SUBMIT AN ADMIN TASK'	11	3	RALVA	30-MAR-18	13-APR-18
7.	☑	27996713	∞	COPYDOMAIN UPDATES FNHBI MEASURE ARRAY HEADERS INCORRECTLY	COPYDOMAIN UPDATES FNHBI MEASURE ARRAY HEADERS INCORRECTLY	11	2	SKOLACHA	29-MAR-18	03-APR-18
8.	☑	27992676	∞	IMAGES SHOW UP ONLY ON Z-AXIS BUT NOT ON X OR Y AXIS	IMAGES SHOW UP ONLY ON Z-AXIS BUT NOT ON X OR Y AXIS	60	3	NKULKAR	28-MAR-18	28-MAR-18
9.	☑	27986648	∞	RPAS-16.0.3.10 INSTALLER FAILS ON AIX	RPAS-16.0.3.10 INSTALLER FAILS ON AIX	11	2	SKOLACHA	19-MAR-18	10-APR-18
10.	☑	27986461	∞	COMMAND LINE ERROR RUNNING CONFIGTOOLS.BAT	COMMAND LINE ERROR RUNNING CONFIGTOOLS.BAT	11	3	ANOMOHAN	19-MAR-18	03-APR-18

Figure 2.3: BugDB Database Schema

- Description:The Detailed description of the bug
- Status: The status of the bug

- Sev:Severity of the Bug
- Assignee:The person whom the bug is assigned
- Reported:The Date on which the Bug is Reported
- Closed:The Date on which the Bug is Fixed i.e Closed

2.3 Introduction to smoke testing and Regression Testing

Smoke testing covers a large portion of the significant elements of the product yet none of them top to bottom. The after effect of this test is utilized to choose whether to continue with additionally testing. On the off chance that the smoke test passes, proceed with additionally testing. On the off chance that it comes up short, stop additionally tests and request another form with the required fixes. In the event that an application is gravely broken, definite testing may be an exercise in futility and exertion.

Smoke test helps in uncovering combination and significant issues ahead of schedule in the cycle. It can be directed on both recently made programming and upgraded programming. Smoke test is performed physically or with the assistance of mechanization apparatuses/contents. In the event that manufactures are arranged every now and again, it is best to mechanize smoke testing.[1]

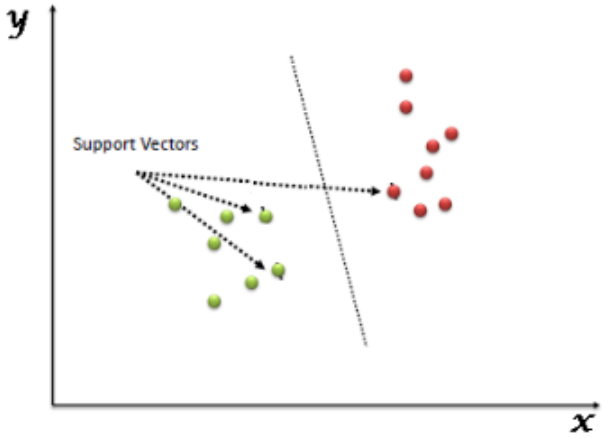
As and when an application ends up develop, with expansion of more functionality and so forth, the smoke test should be made more broad. In some cases, all that's needed is one inaccurate character in the code to render a whole application futile.

Regression testing is the mode toward testing modification to PC projects to guarantee that the more constituted computer programming still works with the new alteration. Regression testing is an ordinary portion of the program betterment procedure and, in larger administration, is processed by code investigation masters. Test office coders makes code test ascertain and activities that will test new units of code after they have been constitute. These scientific research frame what turns into the test pail. Earlier another adjustment of a product item is discharged, the old scientific research are keep running against the brand-new form to ensure that all the old power still work. The explanation they won't not work is on the evidence that ever-changing or adding new code to a

program can without much of a expanse bring blunders into code that isn't anticipated to be varied.[2]

2.4 Support Vector Machine

"Support Vector Machine" (SVM) is a regulated machine learning computing which can be utilized for both command or reverting challenges. In whatsoever suit, it is for the most portion utilized as a part of characterization issues. In this calculation, we plot every datum thing as a factor in n-dimensional space (where n is number of highlights you have) with the approximation of each component being the estimation of a particular form. At that constituent, we perform arrangement by uncovering the hyper-plane that detached the two classes exceedingly well as shown in the figure 2.4.[3]



*Source:*Understanding Support Vector Machine algorithm from examples (along with code) [4]

Figure 2.4: Support Vector

SVM models have functional form to neural systems and outspread premise capacities, both prominent information mining methods. Be that as it may, neither of these calculations has the all around established hypothetical way to deal with regularization that structures the premise of SVM. The nature of speculation and simplicity of preparing of SVM is a long ways past the limits of these more customary strategies.

SVM can show mind boggling, true issues, for example, content and picture characterization, hand-composing acknowledgment, and bioinformatics and biosequence examination.

SVM performs well on informational collections that have numerous traits, regardless

of whether there are not very many cases on which to prepare the model. There is no maximum point of confinement on the quantity of characteristics; the main limitations are those forced by equipment. Conventional neural nets don't perform well under these conditions.

Chapter 3

A Bug's Life

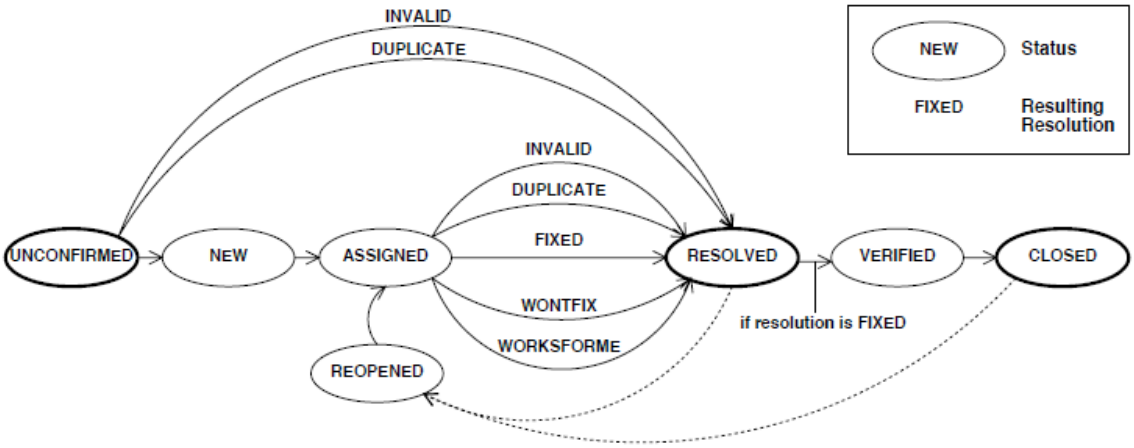
3.1 Overview

Most development groups sort out their work around a bug database. Basically, a bug database goes about as a major list of issuesmonitoring I.e maintaining track of almost every bugs, highlight demands, furthermore, assignments that must be tended to amid the undertaking. Bug databases scale up to big number of developers , clients and issues.

An individual single record in a bug database is known as an issue report; it is otherwise called problem report or ticket. An Bug report gives has an fields such as description (what causes the issue, and the steps to reproduce the defect), a title (a one-line abstract of the description), and additionally a priority (what is the priority of the bug form point of view of the developer), The severity (I.e how severe the bug is.Basically more the severity , higher the priority and less time to fix the bug).The seriousness can extend from "upgrade" (i.e. a component request) over "typical" and "basic" to "blocker" (an issue that stops promote development). These fields are ordinarily given by the first submitter.

At the instant an bug report is submitted, it gets a unique identifier by that it will be brought up in additional communication.Consider an scenario that someone has just entered the bug report in to the bug database. Along with handling the issue , the report goes through an life cycle of an bug 3.1.The situation in the life cycle is dictated by the condition of the issue report. At first, each and every issue report has a province of UNCONFIRMED. It is then checked for legitimacy and uniqueness; in the event that it passes these checks, it turns out to be NEW. Now, the issue report is likewise relegated a needthe higher the need, the sooner it will be tended to.Commonly, the need mirrors the

hazard or potentially harm of the issue/bug. In Bugs report need and state are appeared in the subtle elements section on the left.[5]



Source: Predicting Bugs Components via Mining Bug Reports [6]

Figure 3.1: Life Cycle of Bug

Priority and evaluation I.e estimation are urgent in planning fixesand in assessing at the point when a steady state will be come to.In the long run, the issue report is doled out to a person designerits state is then changed to ASSIGNED. The designer now chips away at the issue, in some cases coming about in extra remarks, inquiries, and re-assignments, all put away in the bug database. In the end, the designer comes up with a determination. This determination can be FIXED, which means that the issue is comprehended, yet additionally WONTFIX (which means the issue isn't considered accordingly) or WORKSFORME (implying that the issue couldn't be recreated). With this determination, the state winds up RESOLVED.

As the issue is presently settled, two more advances remain: the analyzers must affirm the achievement of the fix (bringing about Confirmed state), lastly, the fix must be conveyed as a fix or another discharge, shutting the issue report (CLOSED) what's more, in this manner finishing the issue's life, unless one day, it gets Revived. The bug database consequently is at the focal point of the improvement process. Engineers question bug databases to discover their errands, and also to find out about the undertaking history. Supervisors utilize bug databases to question, plan, and appoint the venture's errands. In the event that the bug database is freely open, clients check it to see the improvement on the bugs they submitted.

As the bug database develops, it turns into an undertaking memory of the gathering-posting every one of the issues as they happened previously, and how they were tended to. As we appear in this paper, this memory can be a significant asset when it comes to survey the undertaking's future.

Chapter 4

Prediction of Patch Release Date

4.1 Architecture

The architecture of the Prediction of Patch Release date is shown in the Fig.**??**.When an new issue is raised The Bug Report of the issue is also generated.Now from that newly generated Report various details like the title, description is extracted.After that the text mining is done on the title and the database is searched.If the similar title is found in the database than the description is extracted and now the text mining is done on the description and now if the description is found to be similar to the report which is there in the database.Than the Reported date and Closed date is extracted and the total no of days is calculated from the difference between the above two dates.Now some days is added and the Final Date is the output that is the date for the release of Patch. Here we have done the following contribution.

- We use existing bug databases to consequently gauge exertion for new issues.
- We utilize content similitude systems to recognize those issue reports which are most firmly related.
- Given an adequate number of issue reports to learn from, our expectations are near the genuine exertion, particularly for issues that are bugs.

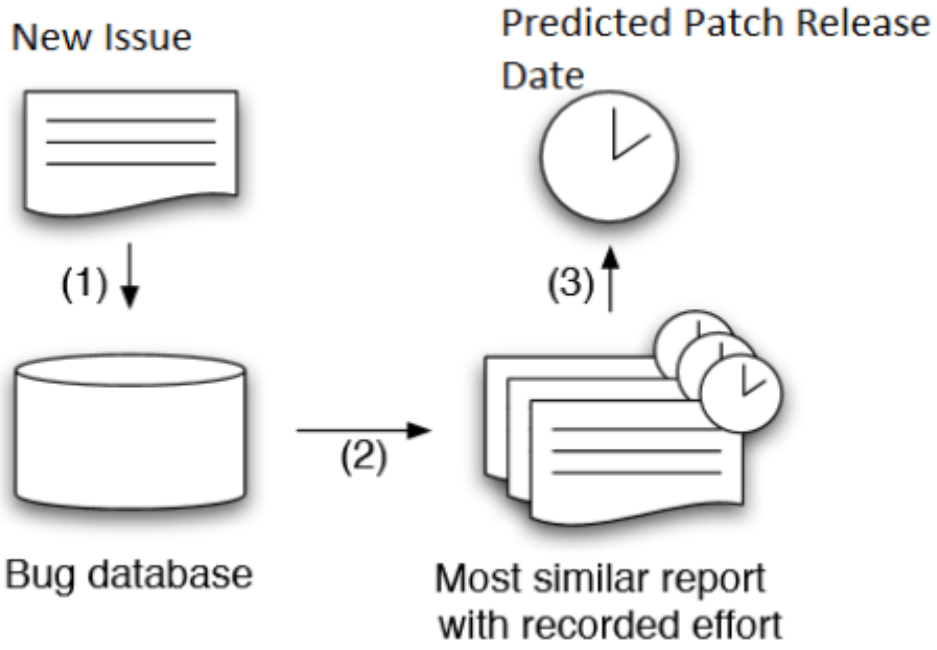


Figure 4.1: Architecture

Let us understand the flow in more detail with the help of flow diagram shown in figure 4.2

Whenever a new bug is raised in the BugDB a bug report is created with a unique id. This bug report has the following details. Bug Number, Bug Title, Severity, Priority, Description , steps to reproduce the bug, Assignee, Status, Reported Date and Updated Date.

4.1.1 CASE 1

Now First the title of the bug is extracted from the bug report. Now using SVN algorithm , it is checked whether the database has the similar type of bug Title. If the similarity in the title is up to 0.80 score than the description is extracted from the report and the similarity of the newly raised description is calculated form the one whose title is similar to the newly raised bug. Now if the similarity of the Description is also matched to the 0.65 percent than the new bug will be consider as similar to the old one. And now the time taken by the bug which is in database is calculated from the Reported Date and the Closed Date by using following formula. Total time (in days) = Updated Date Reported Date. Than extra days for smoke testing,regression testing an overhead is added and the

final output is the release date of the patch.

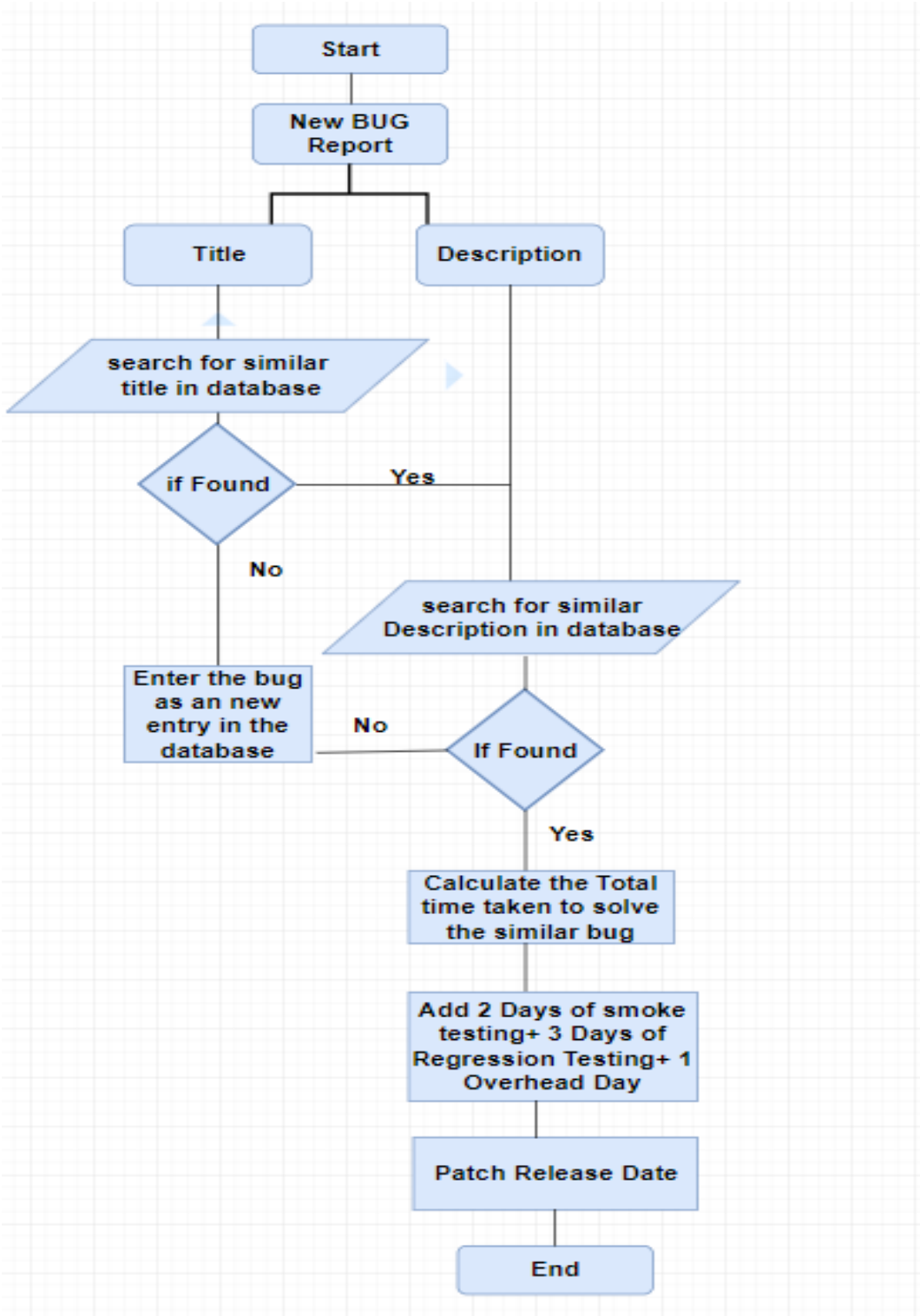


Figure 4.2: Flowchart of Prediction of Patch Release Date

4.1.2 CASE 2

The title from the newly raised bug report is extracted and its similarity is checked from the database. If no similarity (i.e no same functionality is found in the database) than the bud will be entered as an new data in the database. If the title is matched to up to 0.80 score but the description is not matched upto 0.65 score than also the bug will not be consider as an separate new entry.

4.2 Evolution Method

Our approach builds a managed classifier prepared on recorded BRs I.e the historical record of bug report to predict the segment of bug that has newly came. It comprises of two process training process and predicting process, as appeared in Fig.3. In the preparation procedure, we extract the title, description and remarks of a bug as its content. At that point we change over the content into bag of words also, ascertain their TF-IDF (Term Frequency-Inverse Document Frequency) weighting esteems, including stop words separating, word stemming and feature choice (x^2 statistics).Last we apply SVM classifier. In the predicting procedure, we simply separate the synopsis and description of another bug and shows to it as a feature vector, and afterward we foresee the time take to settle the comparative bug by the direction Total No of Days=Closed Date Reported Date.

Figure 4.3 shows the work flow of finding the text similarity of the title and the description of the new bug with the one that was already there in the database.Once the text similarity is found than we need to just calculate the time taken to fix the similar bug and add few days for other testing process and the final output will be the Release Date of the Patch.Here the output will be in number of days and we have provided the field of the starting date i.e when we want to start the process of the fixing the bugs.Than the number of days will be added to the starting date and the final output is the release date of the patch.[7]

4.2.1 Preprocessing

Despite the fact that a bug report contains a considerable measure of data, just piece of the report is valuable for the development of classifiers. We extract Bug number, title , severity,priority and description from each bug report. Keeping in mind the end goal

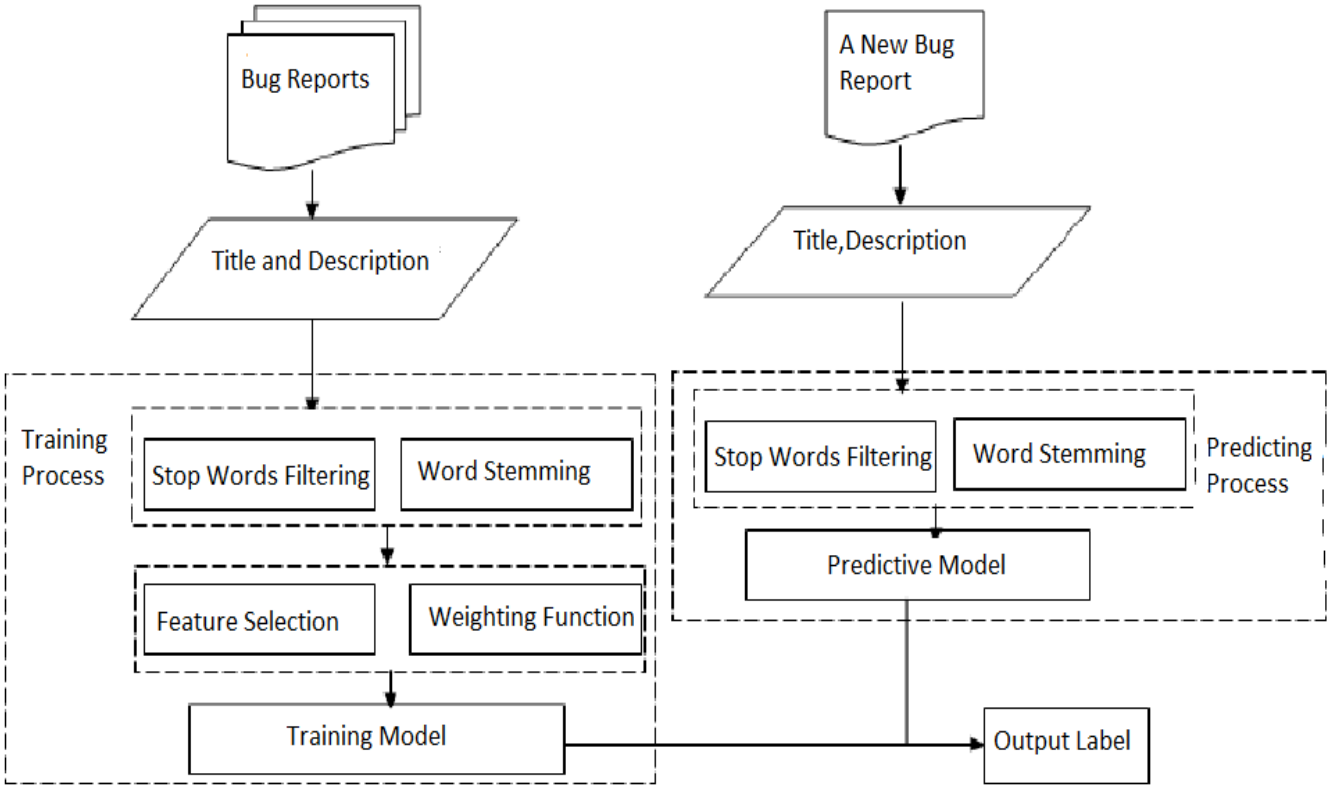


Figure 4.3: Work flow of finding Text Similarity

to describe a bug report, each bug report is changed over into an feature vector. Filtering of stop words, word stemming are likewise acquainted with idealize the component vector. Stop words are exceptionally regular words that are futile in content order. For instance, typically articles,conjunction and relational words are stop words. Stop words are exceptionally normal words that are pointless in classification of text. For instance, typically articles,conjunction and relational words are stop words.[8]

An example of the effects of Preprocessing is shown in Table 4.1 .

Table 4.1: Effecte of Preprocessing

Preprocessing Actions	Result
Original Description	Evolution crashes trying to open Calendar
After stop-words removal	Evolution crashes open calendar
After stemming	Evolut crash open calendar

Preprocessing consists of Three steps. Tokenization,Stop-word-removal and Stemming.

4.2.2 Tokenization

The procedure of tokenization comprises of separating a vast textual string into an arrangement of tokens where a solitary token compares to a solitary term. This progression likewise incorporates sifting through every trivial image like accentuations and commas, on the grounds that these images don't add to the grouping assignment. Likewise, all promoted characters are supplanted by their lower-cased ones.

4.2.3 Stop-word-Removal

Human dialects regularly make utilization of useful terms like conjunctions, verb modifiers, relational words and other dialect structures to develop sentences. Terms like "the", "in" and "that" otherwise called stop-words don't convey much particular data with regards to a bug report. Furthermore, these footing show up much of the time in the portrayals of the bug reports and subsequently increment the magnitude of the information which in twist could diminish the precision of arrangement calculations. This is some of the time likewise alluded as the scourge of dimensionality. Subsequently, all prevent words are expelled from the arrangement of tokens in light of a rundown of known stop-words. [9]

4.2.4 Stemming

The stemming step goes for lessening each term showing up in the depictions into its essential shape. Each single term can be communicated in various structures yet at the same time convey a similar particular data. For instance, the expressions "mechanized", "modernize" and "calculation" all offer the same morphological base:"computer". A stemming calculation like the watchman stemmer [11] changes each term to its fundamental frame.

4.3 TF-IDF Weighting Function

Once the preprocessing of Bug Report is completed, each bug report is changed over into an arrangement of catchphrases. We connected CHI to choose highlights from our BRs

corpus. The CHI estimation of between a term t and a class c is characterized to be

$$x^2(t,c) = \frac{N * (AD - CB)^2}{(A+C) * (B+D) * (A+B) * (C+D)} \tag{4.1}$$

In the above equation A is the circumstances (the total times of) t c co-happen, B is the circumstances t happens without c, C is the quantity times c happens without t, D is the circumstances neither c nor t happens, and N is the aggregate number of archives. We figure for every classification the x^2 measurement between every interesting term in the preparation corpus and that classification, and after that entirety the class exceptional scores of each term into one grade.

$$x^2_{avg}(t) = \sum_{i=1}^{m} P(c_i)x^2(t_i, c_i) \tag{4.2}$$

In the above Equation $P(c_i)$ parallels the quantity of reports in class $c_i$isolated by the aggregate number of reports, m is the aggregate number of classifications. Arranging terms by the estimation of x^2_{avg} in opposite to the forward sorting order,we select the first K terms feature.

The capacity of term weight is utilized to assess the weight of term t in report d after element determination. A number of term weighting capacities and their variations have been proposed in content characterization. TF-IDF weighting capacity is a factual measure for assessing how vital a word is to an archive in a corpus. In our approach, we utilize the best term weighting equation to calculate the weight of one term. The formula is defined to be

$$w_{ij} = \frac{tfidf(t_i, d_j)}{\sqrt{\sum_{k=1}^{v}[tfidf(t_k, d_j]^2)}} \tag{4.3}$$

In the above Equation w_{ij} is the value of the weight of the i term in the database.The weight of all the data is calculated and the title whose weight is more similar to the one is consider as the base of the new bug and the details of that bug is used to calculate the fixing time of the new one.

4.4 Results of the Prediction of Patch Release Date

4.4.1 Login

Figure 4.4 shows the Login Page of the Prediction Of Patch Release Date.

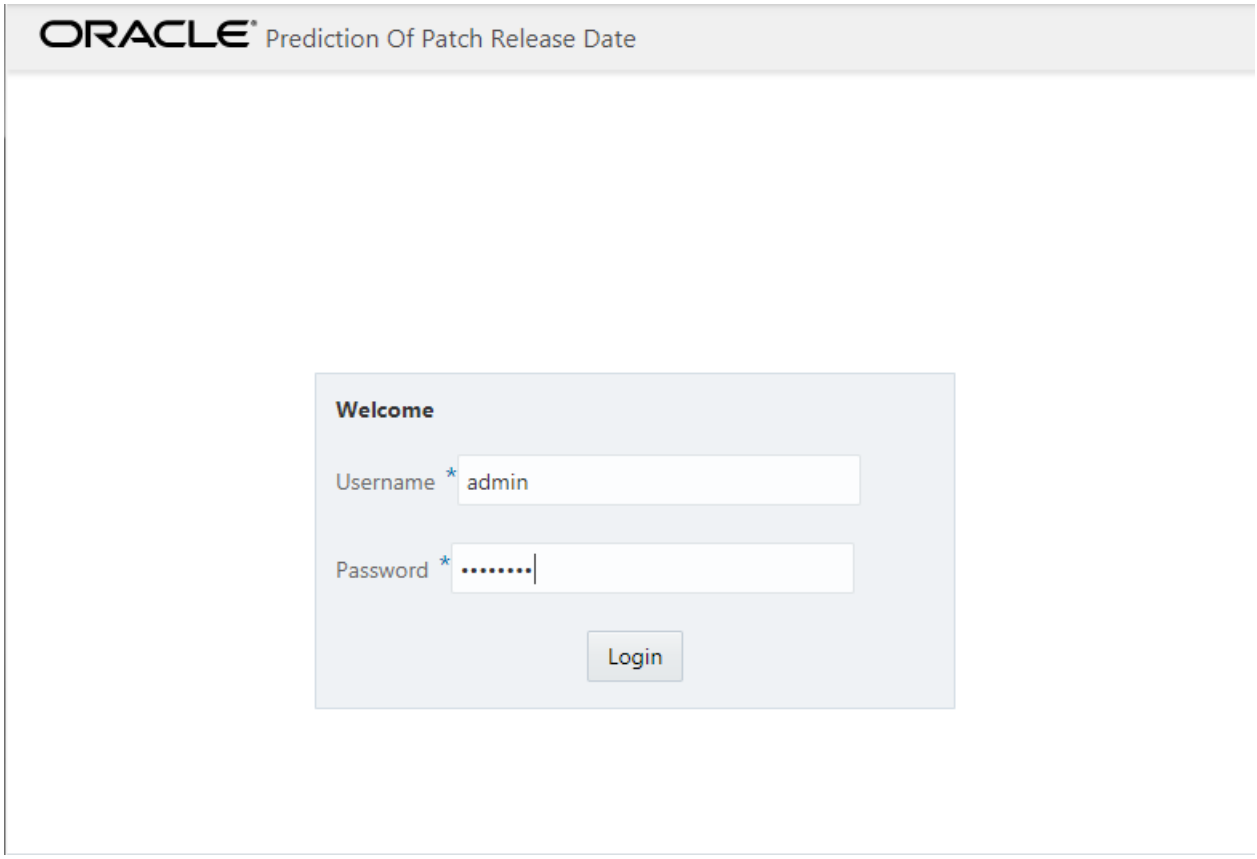


Figure 4.4: Login Page

The Prediction of an Patch Release Date has an Differennt Tabs in the Main Page as Shown in Figure 4.5

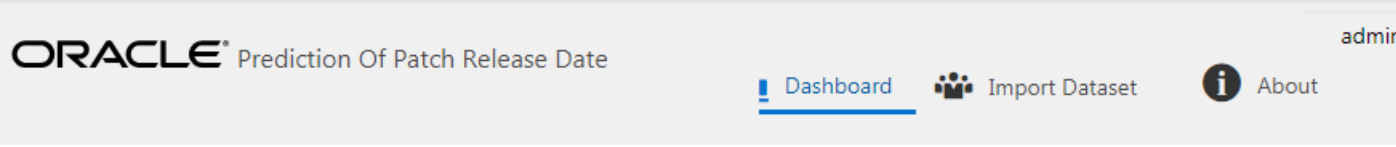


Figure 4.5: Tabs in Main Page

As the New Bug is entered in to the database the Bug Report is Generated now this bug report works as the base of the prediction.Figure 4.6 shows that when the import Dataset Tab is clicked the following page opens up and which allows user to enter the Bug Report.

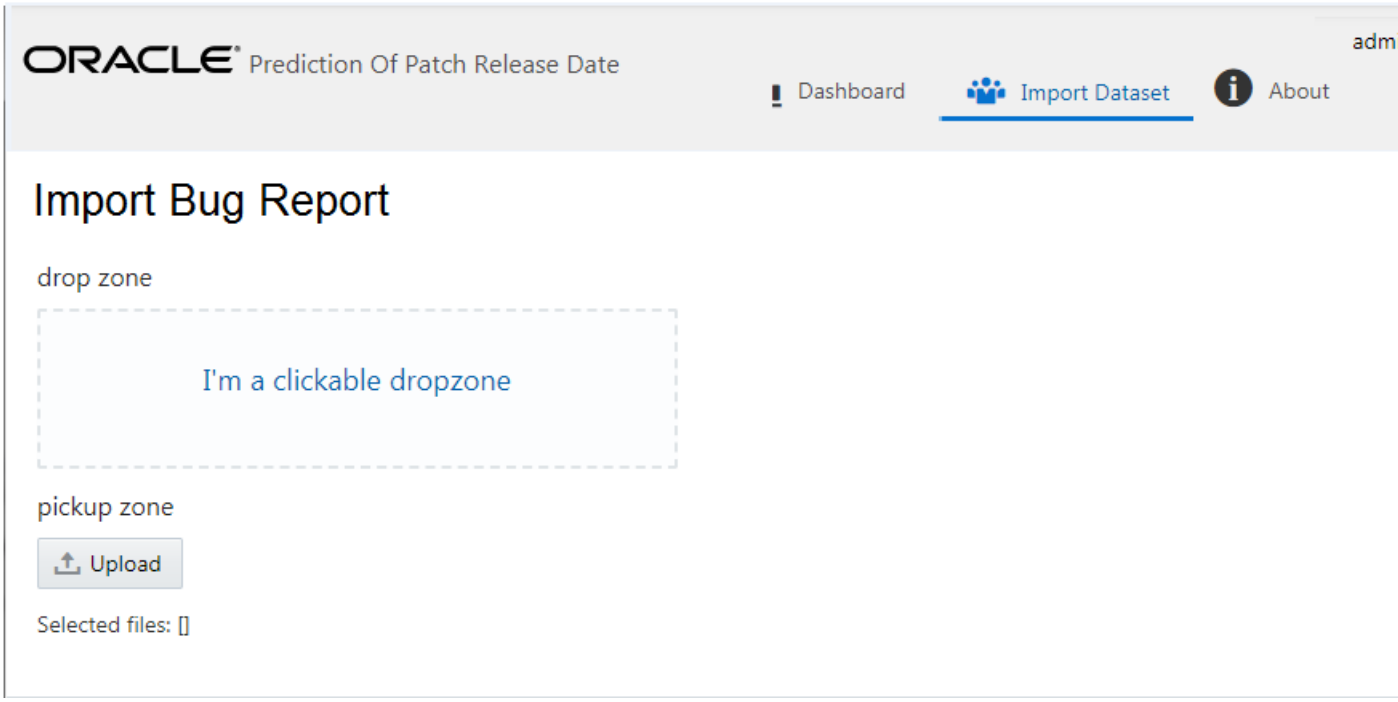


Figure 4.6: Import Bug Report

After the Bug Report is Uploaded than the Title is fetched from the bug report and the similar title is searched in to the database and if it is found than it is checked that whether the bug has the similarity score upto the threshold for description if same is found than the Total days to fix the similar bug is calculated and than additional days is added and the final Output is the No of days required to release the patch.

Figure 4.7: Output

Chapter 5

Conclusion And Future Scope

5.1 Summary Of the Work

In Prediction of the Patch release date we have utilized the verifiable date i.e the current bug reports to discover the comparability with the recently raised bug report and if the likeness is found than the date i.e the time required to complete that comparable bug is figured and this will function as a base for the forecast of an opportunity to fix this bug. Than few days is included and the last yield is the Date on which the patch(i.e the minor version) will be available to release.

5.2 Conclusion

Given an adequate amount of prior issue reports, our programmed exertion predictor beats the nave approach; specifically, our expectations are very close to bug reports. As an outcome, it is conceivable to predict the effort and the time at the exact minute another bug is accounted for. This gives a big relief to the project manager who have a long line of bug reports holding up to be assessed as well as estimated, and this will be a great help to the managers to decide the allocation of the resources as well as planning the upcoming releases. The Performance of this approach is far more better considering the fact that the effort and time predictor is only relied on the two data points I.e the title of the bug report and the description of the bug report.

5.3 Future Work

The Bug Report contains numerous Field like version data i.e Fix by tag and Fixed version,Steps to reproduce,Assignee,attachments and numerous more.This will require some other feature model form text integrating and will help to predict the time and effort more accurately.In Future we will use this data to track the performance of each of the employee on the premise add up to number of the bugs fixed by them and the time taken for the same.By doing this every single level administrator can track the execution of every worker.

Bibliography

[1] "Smoke testing (url: http://softwaretestingfundamentals.com/smoke-testing/)."

[2] "Regression testing (url: https://smartbear.com/learn/automated-testing/what-is-regression-testing/)."

[3] "Regression testing (url: https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/)."

[4] "Oracle retail predictive application server guide (url: http://docs.oracle.com/)."

[5] "J. anvik, l. hiew, and g. c. murphy. coping with an open bug repository. in proc. of the oopsla workshop on eclipse technology exchange, pages 3539, 2005.."

[6] D. Wang, H. Zhang, R. Liu, M. Lin, and W. Wu, "Predicting bugs' components via mining bug reports.," *JSW*, vol. 7, no. 5, pp. 1149–1154, 2012.

[7] "G. salton, c. buckley, term-weighting approaches in automatic text retrieval, information processing management, 24 (5), pp. 513523.."

[8] "Meera sharma , and madhu kumari,the way ahead for bug-fix time prediction."

[9] "S. kim and j. e. whitehead, how long did it take to fix bugs?, int. workshop mining software repositories. new york, ny, usa, acm, pp. 173174, 2006."