INDEX

1. ABOUT THE PROJECT

1.1. ABSTRACT :

The "Bug Tracking System" was created to overcome the flaws that existed in the previous manual system. This programme is supported in order to eliminate and, in certain situations, decrease the difficulties encountered by the current system. Furthermore, this system is tailored to the company's specific requirements for smooth and efficient operations. This application has been kept as simple as possible in order to avoid data entry errors. It also displays an error notice when entering invalid data. The user does not require any formal knowledge to use this system. Thus, all of this demonstrates that it is user-friendly. As previously explained, a bug tracking system can lead to an error-free, secure, dependable, and rapid management system. It might help the user focus on their other tasks rather than record keeping. As a result, it will assist organisations in making better use of their resources. Every software organisation, large or little, faces issues in handling Project, Bug, Tester, Bug Category, and Bug Type information.

Bug tracking and management may be a exceptionally imperative and basic portion of the software improvement lifecycle not particularly within the testing stage but in all stages as issues emerge in all stages of the software improvement lifecycle and not only within the testing stage how the issue can be identified within the collection of necessities in terms of lost necessities, equivocal and conflicting within the usage stage in terms of lost comes about, lost utilitarian characteristics, etc., there are so numerous apparatuses accessible on the advertise to screen and oversee specialized bugs so that all issues can be overseen accurately and accurately consideration to each issue depends on its seriousness and need, a few apparatuses are restrictive apparatuses such that Microsoft Group Establishment Server and Hewlett Packard and a few are Open Source computer program such as JIRA, Bugzilla, Mentis, Bug Tracker, Bug Genie, iTracker , Web issues, etc. Productive bug detailing and sorting is basic for any program extend II. BACKGROUND Utilizing as it were normal dialect data may not identify a few common mistake messages due to the assortment of normal dialect employments. In this case, the execution data may be more dependable. In any case, utilizing as it were data approximately the execution may have its disadvantages: not one or the other data within the normal dialect, nor data almost the usage is continuously predominant to others in all cases. In specific, considering both sorts of data may contain the taking after points of interest. To begin with, the common dialect data gotten from the blunder portrayal is likely to speak to the behavior of the outside blunder that the mistake detailing pro watches, whereas the comparing execution data may record inside irregular behavior. Subsequently, the utilize of both sorts of data can permit for both outside and inside behavior when detailing a blunder report.

1.2. MODULES :

1. Authenticate User :

The Bug Tracking System first activates the login form. Here the user enters the User name and password and our system starts the authentication process in whichthe username and password are matched with the existing username and passwordin the database. If the password matches then it is allowed to the main page else it warns the user for Invalid User name and password.After successful authentication the system activates menus. The activity logalso prepared for failures and security.

2. Products:

● List Of Products:

After successful authentication the user is provided with the listexisting products. Here the user can view the details of productsand can modify the existing products. This project even providesthe facility of adding new projects.

● Product Versions:

All the products are maintained in several versions. As it is notpossible to complete the whole project in a single versionFeatures required for the product are categorized into severalversion with dead lines. And the versions are completedaccording to their dead line dates. Here the user can add newversions to a product or can modify the existing details ofversion.

●Product Users :

In order to complete the project each product is allotted with Resources or users. First all the employees with their names and qualifications are stored in the database. Each user is allotted to the product based on their rating, Qualification and designation. For each user Effective date is stored which specifies the total period a user is valid for that product.

3. Bug Details:

● Bug Details

In this module the user is provided with the facility for addingbugs or updating the existing bugs. As the number of bugs for aproduct can be very large this system is provided with efficientfiltering. The user

can filter the bugs based on the priority,database, operating system and status. After the user appliesfilter the list of bugs are displayed from the database.

● Bug History

Here the bug history is maintained. All the solutions given forthe bug resolution by various users are stored. As the bug needsseveral techniques or methods for resolution it is important tostore the history of the bug.

● Bug Assignee

This displays the list of users for whom the bug is assigned forresolution. As the bug need to be resolved for completing theproduct several user are assigned to find a solution for the bug.The user can add this bug to a new user or he can modify theexisting user details.

● Bug Attachments

This gives a list of attachments for a particular bug. The bug canbe of any type it can be a database bug or a GUI bug. So whileyou add a bug you need to provide with the details of bug. Sothe file attachments can be a document, database file or animage file. All then attachments are stored in a location alongwith the size and type of the file. Here the user can add a newattachment or can change the details of existing files.

4. Bug Tracking

●Track Hierarchy

All the bugs saved in the database will have a particular hierarchy. There might be bugs which can be related to the earlier bugs saved in the database so our system is provided with a hierarchy. And user can add child nodes in this hierarchy or he can modify the existing values of the nodes. This hierarchy is based on the parent child relation ship between the bugs.

●Track Resolution

This displays a list of all solutions provided by the users allotted to a bug. This stores the action type and the necessary resolution provided by the user.

●Track Resources

This displays list of resources allotted to the project. As the bugs need to be resolved resources are provided for the bugs. These Resources will be the resources allotted to the project. The resources are allotted based on the rating of the employee.

5. View

●Product Bu g Hierarchy

This module is just for displaying the hierarchy for the easy Look ofthe bugs. Here the bugs are displayed in the form of parent childnodes. As it is difficult for the user to look at the vast number of bugsin the database. And one cannot easily access the relation between thebugs.

●Product User Hierarchy

This module if for displaying the users allotted to the bug. The usersalong with their name and designation are displayed in this module.Even in the allotment of resources there can be hierarchy between theemployees depending on their designation. So this module simplifiesthe hierarchy among the employees.

6. Search

Our system provides with the feature of advanced search technique. Generally Number of bugs for a project increased tremendously so if we want to know about a particular bug It takes much amount of time. With the search screen provided one can filter the bug's base on priority, product, severity, database and type of operating system. He can also list the bugs between particular time based on the start date and end date. After Searching it displays a list of bugs. From this list the user can modify the existing bugs or can add a new bug.

7. Admin

●UsersAll the users of this system are displayed in this module. One can addnew user or can update the details of an existing user. Here thepassword provided by the user is encrypted before saving them to thedatabase for proper security. This module saves the details likeaddress, phone and email.

●ConfigurationAll the Values that we are using in this system are configurable. Valueslike status, priority and others can be added dynamically on thescreen. Suppose if we limit these fields by hot coding them and if theuser wants to add a new value again he has to come to the developerof the product. So In order to avoid this it is provided with the featureof adding values from the screen. And the user can change the statusto In Active whenever he wants.

●Log ViewIn order for the efficient Tracking of the system logs are maintained.As the logs will be in vast it will be a problem for user for checking thedatabase. The Log View module can be searched based on the userand Records between a start date and end date.

8. Logout

In this once the user clicks on Log out First the session variable is killed and then the system is redirected to the login page.

9. Prepare Logs

At all the stages, whenever user performs an operation by clicking a button,automatically the Bug Tracking System logs the activity.

1.3. COMPANY PROFILE:

TechCiti, established in 2013, is a prominent player in the realm of end-to-end IT infrastructure solutions, catering to businesses of all sizes. With a footprint in over 12 major Indian cities, we have positioned ourselves as a trusted partner for corporations seeking customized technology solutions. Our journey began with a commitment to providing comprehensive services that address the unique needs of each client. Our free consultation service stands as the first step in understanding and defining the requirements of our clients, enabling us to tailor solutions that align with their business objectives.

At TechCiti, we recognize the importance of guiding our clients through the complexities of technology integration. Our dedicated team assists in product selection, configuration, and installation, ensuring a seamless onboarding process. We take pride in our consultative approach, prioritizing value demonstration by helping clients make informed decisions. Whether you are a startup or an established enterprise, our focus remains unwavering — bringing competitive pricing, predictability in execution, and exceptional post-sales support to the table.

Driven by a commitment to excellence, TechCiti has successfully forged long-lasting relationships with a diverse range of corporate customers. Our emphasis on understanding the unique challenges faced by businesses sets us apart. By staying at the forefront of technological advancements, we empower our clients with the tools they need to thrive in an ever-evolving digital landscape. TechCiti is not just a service provider; it's a partner dedicated to the success and growth of the businesses we serve.

Beyond being a service provider, TechCiti is a strategic ally invested in the success of the businesses we serve. Our post-sales support is characterized by excellence, ensuring that our clients can rely on us beyond the initial implementation. In a rapidly evolving digital landscape, we remain committed to staying at the forefront of technology, empowering our clients with the tools and insights needed to navigate and thrive in the dynamic world of IT infrastructure. TechCiti is not just a solution provider; it's a pioneering force dedicated to shaping the future of IT services in India.

2. LITERATURE SURVEY

2.1 INITIAL INVESTIGATION:

1. An Eye Tracking Research on Debugging Strategies towards Different Types of Bugs

AUTHORS: Fei Peng; Chunyu Li; Xiaohan Song; Wei Hu; Guihuan Feng

YEAR:2020

Debugging is one of the important links in software quality assurance. Generally, the debugging performance of people adopting different debugging strategies varies enormously. Although there are studies discussing debugging strategies, little research analyzes the impact of these strategies towards different types of bugs. In this paper, the experiments conducted on 20 participants suggest that there do exist differences on the eye movement data of those successful and failed debugging samples. Specifically, concerning data flow bugs, it is beneficial to pay attention to the changes of variables, Nevertheless, it is more important to watch the code and understand their logical structure when dealing with control flow bugs. We believe it can help programmers find defects more efficiently by combining this conclusion and the error message provided by the compiler.

2. Bug Tracking Process Smells In Practice

AUTHORS: Erdem Tuna; Vladimir Kovalenko; Eray Tüzün

YEAR:2022

Software teams use bug tracking (BT) tools to report and manage bugs. Each record in a bug tracking system (BTS) is a reporting entity consisting of several information fields. The contents of the reports are similar across different tracking tools, though not the same. The variation in the workflow between teams prevents defining an ideal process of running BTS. Nevertheless, there are best practices reported both in white and gray literature. Developer teams may not adopt the best practices in their BT process. This study investigates the non-compliance of developers with best practices, so-called smells, in the BT process. We mine bug reports of four projects in the BTS of JetBrains, a software company, to observe the prevalence of BT smells in an industrial setting. Also, we survey developers to see (1) if they recognize the smells, (2) their perception of the severity of the smells, and (3) the potential benefits of a BT process smell detection tool. We found that (1) smells occur, and their detection requires a solid understanding of the BT practices of the projects, (2) smell severity perception varies across smell types,

and (3) developers considered that a smell detection tool would be useful for six out of the 12 smell categories.

3. Feature Ranking and Aggregation for Bug Triaging in Open-Source Issue Tracking Systems

AUTHORS: Anjali Goyal; Neetu Sardana

YEAR:2021

The increasing complexity and team-based projects have lead to the rise of various project management tools and techniques. One of the important components of open- source project management is the usage of bug tracking systems. In the last few decades, software projects have experienced an inescapable appearance of bug reports. One of the main challenges in handling these incoming bugs is triaging of bug reports. Bug triaging can be considered as a mechanism for the election of a suitable software developer for a reported bug who will work towards resolving bug in a timely fashion. There exist several semi and fully automated bug triaging techniques in the existing literature. These techniques often consider varied bug parameters for prominent developer selection. Past researchers have concluded different parameters to be possessing prime importance in the optimal developer selection task. However, a common ranking scale depicting the importance among different bug parameters for bug triaging is not available. This paper presents a methodology to rank the non-textual bug parameters using feature ranking and aggregation techniques. The presented methodology has been evaluated on four open-source systems, namely, Mozilla Firefox, Eclipse, GNome and Open Office. From the experimental evaluation, it has been observed that the ranking of bug parameters is consistent among the different open-source projects of Bugzilla repository.

4. An Expert System Framework for Bug Tracking and Management

AUTHORS: Abdul Wahab Khan; Sanjay Kumar

YEAR:2020

Open Source software wanders, fair as closed software exercises and firms, utilize the Issue tracking framework. The data of Issue tracking frameworks is uncommonly critical for distinctive sorts of investigate moreover within the correct Program Building. For Open Source Software ventures a noteworthy degree of this data is available transparently and with approximately unhindered get to. Intentionally dismembered, this data can appear a awesome bargain and deliver unused bits of information around an Open Source ventures - for occasion with regard to the task’s region or its progression. The amassed information from an colossal degree of Issue tracking frameworks can allow

choices approximately Open Source software progression practices as a run the show. Along these lines, it may be a critical test to form the information accessible and discharge its potential for exploratory investigate. This investigate will display a utilization that enables us to standardize and utilize the information of a colossal degree of Issue tracking.

5. Bugtrac – A New Improved Bug Tracking System

AUTHORS: Madhwaraj Kango Gopal; Govindaraj M; Paramita Chandra; Prathiksha Shetty; Sunny Raj

YEAR:2022

Software testing is a concept that is associated with the identification and detection of bugs. Most software bugs emerge from mix-ups and mistakes made by individuals in either a program's source code or in its technical design. Bugs are found in a variety of forms while using software applications and therefore it becomes difficult to detect and manage them properly for an individual/user of an application. In order to resolve this issue, bug tracking systems have been introduced to monitor and keep track of the reported bugs in an application. There exist many proprietary and open-source bug tracking systems today and some new systems are also under development to cater to the variety of bugs that keep originating due to the changing requirements of software. There is a need for a tool that will help in fixing and tracking the progress of bug fixes with the ever-changing types of software applications that keep evolving every day. In this paper, a comparative study has been performed between five defect tracking systems, a combination of both open source and proprietary. A new defect tracking system “Bugtrac” has been proposed taking into consideration all types of software applications and the defects that may originate from these applications. The proposed system is expected to become a more promising defect tracking system when compared with the already existing ones.

2.2 PROBLEM DEFINITION:

• No use of Web Services or Remote Access.

• The risk of mismanagement and data loss while the project is in development.

• Reduced security.

• Inadequate coordination between various applications and users.

2.3. EVALUATION OF EXISTING SYSTEM:

Bugs discovered by testers during the software testing process are reported to the Project Manager and developer using simple shared lists and emails. The majority of businesses provide this information via a document known as a "defect report." Because there is no specific tracking system in place, this technique is prone to errors, and there is a good probability that some defects will go unfixed and disregarded. Each bug reported must be tracked by the team involved in the software development life cycle. The current system does not meet this need, which has an impact on team productivity and responsibility.

2.4. PROPOSED SYSTEM:

The following actions are included in the creation of the new system, which attempt to automate the entire process while keeping the database integration method in mind.

1. The application provides user friendliness through many controls.

2. The system simplifies and adapts entire project management.

3. There is no chance of data mismanagement at any stage when the project is being developed.

4. It offers a high level of security with many levels of authentication.

FEASIBILITY STUDY

All projects are feasible given unlimited resources and infinite time. Unfortunately the development of computer-based system in many cases is more likely to be plagued by scarcity of resources and delivery date. Hence, we have made use the concept of reusability that is what Object Oriented Programming (OOPS) is all about. The feasibility report of the project holds the advantages and flexibility of the project. This is divided into three sections: ¬ Economical Feasibility ¬ Technical Feasibility ¬ Behavioral Feasibility

- **ECONOMIC FEASIBILITY:**

Economic analysis is the most frequently used method for evaluating the effectiveness of the candidate system. More commonly known as cost/benefit analysis, the procedure is to be determining the benefits and savings that are expected from a candidate and compare them with costs. If benefits outweigh costs, then the decision is made to design and implement the system. A systems financial benefit must exceed the cost of developing that system. i.e. a new system being developed should be a good investment for the organization. Economic feasibility considers the following i. The cost to conduct a full system investigation. ii. The cost of hardware and software for the class of application. iii. The benefits in the form of reduced

cost or fewer costly errors. iv. The cost if nothing changes (i.e. the proposed system is not developed). The proposed Movie Recommendation System is economically feasible because the system requires very less time factors. ii. The system will provide fast and efficient automated environment instead of slow and error prone manual system, thus reducing both time and man power spent in running the system. iii. The system will have GUI interface and very less user-training is required to learn it. iv. The system will provide service to view various information for proper managerial decision making.

- **TECHNICAL FEASIBILITY:**

Technical feasibility centers around the existing computer system (Hardware and Software etc.) and to what extend it support the proposed addition. For example, if the current computer is operating at 80 percent capacity - an arbitrary ceiling - then running another application could overload the system or require additional Hardware. This involves financial considerations to accommodate technical enhancements. If the budget is a serious constraint, then the project is judged not feasible. In this project, all the necessary cautions have been taken care to make it technically feasible. Using a key the display of text/object is very fast. Also, the tools, operating system and programming language used in this localization process is compatible with the existing one.

- **BEHAVIORAL FEASIBILITY:**

People are inherently resistant to change, and computers have been known to facilitate change. An estimate should be made of how strong a reaction the user staff is likely to have toward the development of a computerized system. Therefore it is understandable that the introduction of a candidate system requires special efforts to educate and train the staff. The software that is being developed is user friendly and easy to learn. In this way, the developed software is truly efficient and can work on any circumstances, tradition, locales. Behavioral study strives on ensuring that the equilibrium of the organization and status quo in the organization are not disturbed and changes are readily accepted by the users.

2.5 SOFTWARE SELECTION:

2.5.1 SOFTWARE TOOLS:

1. ANOCONDA PROMPT:

working on a Django project and using Anaconda Prompt for managing your Python environment, Below are some key points you might want to cover when describing how you use Anaconda Prompt for running a Django project in your report:

1. Introduction to Anaconda Prompt:

- Briefly explain what Anaconda is and why you chose to use it for your project.

- Mention that Anaconda Prompt is a command-line interface provided by Anaconda for managing Python environments.

2. Setting up a Conda Environment:

- Describe how you created a dedicated Conda environment for your Django project. This helps in managing dependencies and avoiding conflicts with other projects.

```
>>> conda create --name mydjangoenv python=3.8
>>> conda activate mydjangoenv
```

3. Installing Django:

- Explain how you used Anaconda Prompt to install Django within your project's Conda environment.

```
>>> conda install django
```

4. Running the Django Development Server:

- Provide the command you use to start the Django development server.

```
>>> python manage.py runserver
```

- Explain any additional parameters you use, such as specifying a different host or port.

```
>>> python manage.py runserver 0.0.0.0:8000
```

5. Database Migrations:

- If applicable, describe how you handle database migrations using Django commands in Anaconda Prompt.

```
>>> python manage.py makemigrations
>>> python manage.py migrate
```

6. Running Tests:

- If you've written tests for your Django application, mention how you run them using the Anaconda Prompt.

```
>>> python manage.py test
```

2. VISUAL STUDIO CODE:

1. Introduction to Visual Studio Code

- Provide an overview of Visual Studio Code, emphasizing its popularity, open-source nature, and its cross-platform support (Windows, macOS, Linux).

- Mention key features such as a lightweight yet powerful editor, built-in Git support, extensions, and a wide range of language support.

2. Installation and Setup:

- Explain the process of installing Visual Studio Code on your chosen operating system.

- Briefly describe the initial setup steps, including configuring preferences and extensions.

3. Workspace and Project Management:

- Describe how Visual Studio Code handles workspaces and projects.

- Highlight features like the ability to open multiple folders in a single window, the use of workspace settings, and the handling of project-specific configurations.

4. Integrated Terminal:

- Discuss the benefits of the integrated terminal within VS Code for running commands and scripts directly from the editor.

- Showcase examples of using the terminal for tasks such as package installation, running scripts, or managing version control.

5. Extensions:

- Emphasize the extensibility of Visual Studio Code through extensions.
- Mention any specific extensions you find particularly useful for your development workflow. For example, extensions for languages, frameworks, or tools related to your project.

6. Source Control Integration:

- Highlight the built-in Git integration in VS Code.
- Explain how you manage version control tasks such as committing changes, viewing diffs, and pushing/pulling from remote repositories.

7. Code Editing Features:

- Discuss the intelligent code editing features of VS Code, such as IntelliSense, code navigation, and code snippets.
- Provide examples of how these features enhance your coding experience.

8. Debugging Capabilities:

- Explain how Visual Studio Code supports debugging for various languages and frameworks.
- Provide examples of setting breakpoints, inspecting variables, and using the debugger to troubleshoot issues in your code.

9. Task Automation:

- If applicable, mention how you use the built-in task automation features or external task runners to streamline repetitive tasks.

10. Collaboration and Live Share:

- If relevant, discuss how Visual Studio Code facilitates collaboration through features like Live Share, allowing multiple developers to collaborate in real-time.

3. PYTHON:

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

Install Python Step-by-Step in Windows and Mac :

Python a versatile programming language doesn't come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high-level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace.

The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

How to Install Python on Windows and Mac :

There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Before you start with the installation process of Python. First, you need to know about your **System Requirements**. Based on your system type i.e. operating system and based processor, you must download the python version. My system type is a **Windows 64-bit operating system**. So the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. Download the Python Cheatsheet here.The steps on how to install Python on Windows 10, 8 and 7 are **divided into 4 parts** to help understand better.

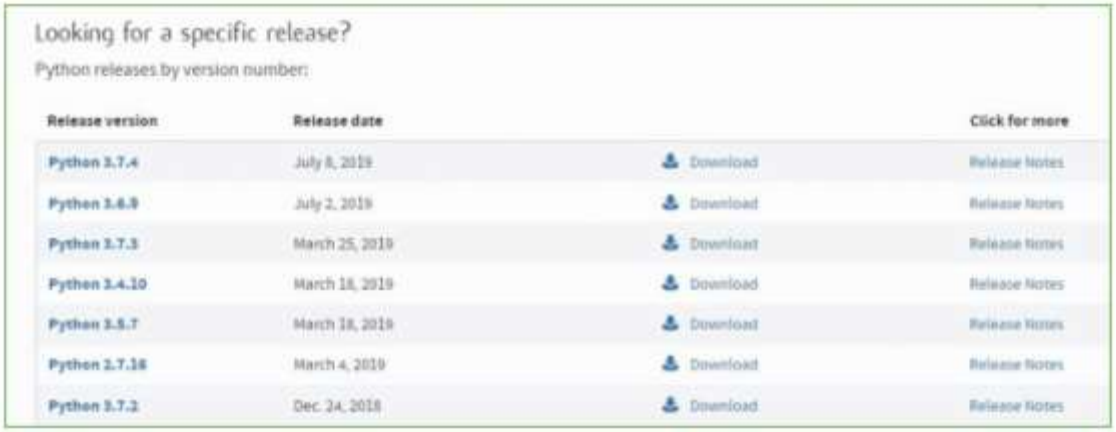
Download the Correct version into the system

Step 1: Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: **https://www.python.org**

Now, check for the latest and the correct version for your operating system.

Step 2: Click on the Download Tab.

Step 3: You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4



Looking for a specific release?

Python releases by version number:

Release version	Release date		Click for more
Python 3.7.4	July 8, 2019	Download	Release Notes
Python 3.6.9	July 2, 2019	Download	Release Notes
Python 3.7.3	March 25, 2019	Download	Release Notes
Python 3.4.10	March 18, 2019	Download	Release Notes
Python 3.5.7	March 18, 2019	Download	Release Notes
Python 2.7.16	March 4, 2019	Download	Release Notes
Python 3.7.2	Dec. 24, 2018	Download	Release Notes

Step 4: Scroll down the page until you find the Files option.

Step 5: Here you see a different version of python along with the operating system.

Files

Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		6811171e502db4aef7bfab019f9f9be	23017663	SIG
XZ compressed source tarball	Source release		d33e4aae66097051c2eca45ee3604803	17131432	SIG
macOS 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	6428b4fa75f3daff1a442cbabcee08e6	34898416	SIG
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	5db605c38217a457738f5e4e938b243f	28082845	SIG
Windows help file	Windows		d639985173a2c062ac56cade6b4f7cd2	8131761	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	9b00cdcf0df9ec0b6abe8319ae45725a2	7504391	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	a7f02b4b0ad71debdb3041af83e563400	26480368	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	28c51c008bbd71aebe53a38d3513b4bd2	1362904	SIG
Windows x86 embeddable zip file	Windows		9fab3b81f884187f9da94133574139d8	6741626	SIG
Windows x86 executable installer	Windows		33cc6029420544e46a3d64516763947f9	25863848	SIG
Windows x86 web-based installer	Windows		1b670cfa5d117df92c3f983ea371d87c	1324608	SIG

• To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.

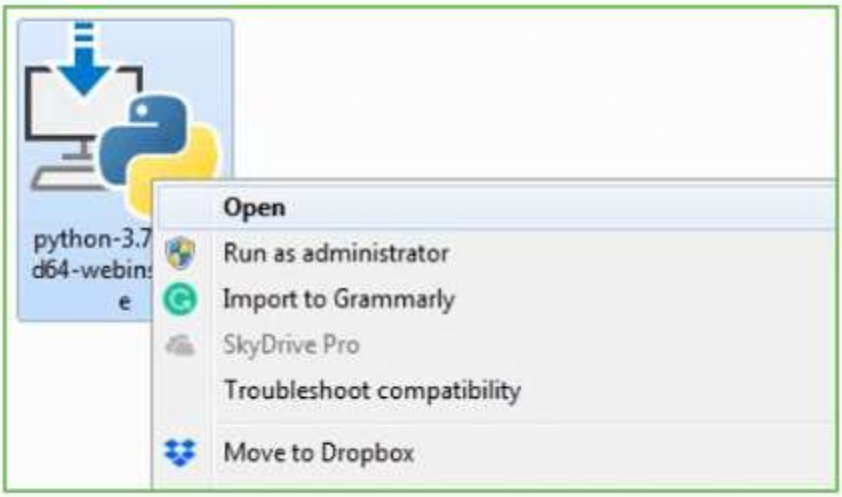
•To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e. Installation

Note: To know the changes or updates that are made in the version you can click on the Release Note Option.

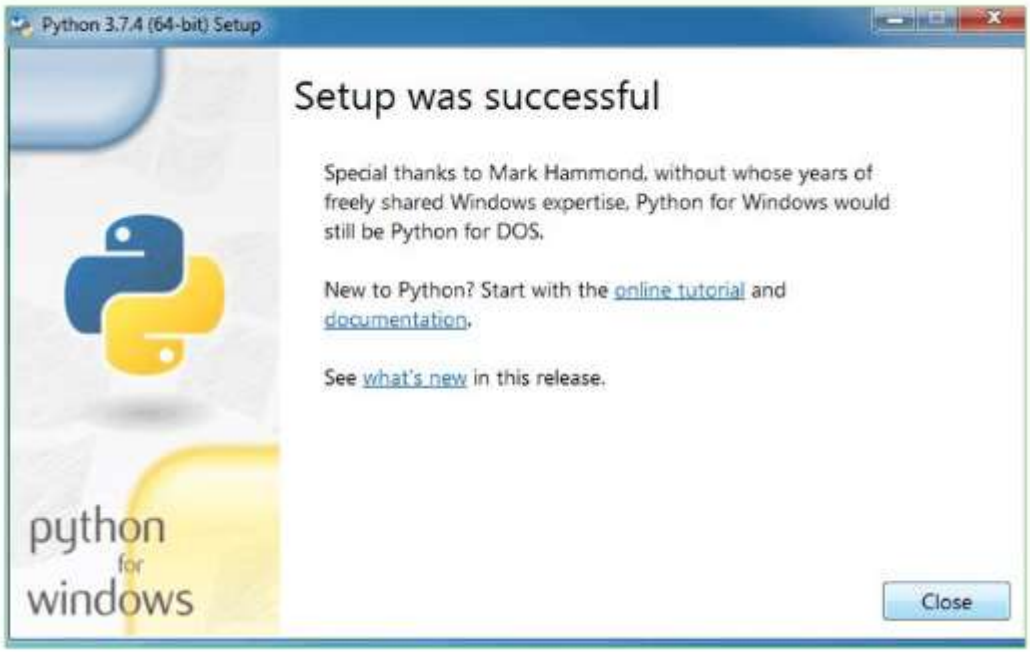
Installation of Python

Step 1: Go to Download and Open the downloaded python version to carry out the installation process.



Step 2: Before you click on Install Now, Make sure to put a tick on Add Python 3.7 to PATH.

Step 3: Click on Install NOW After the installation is successful. Click on Close.



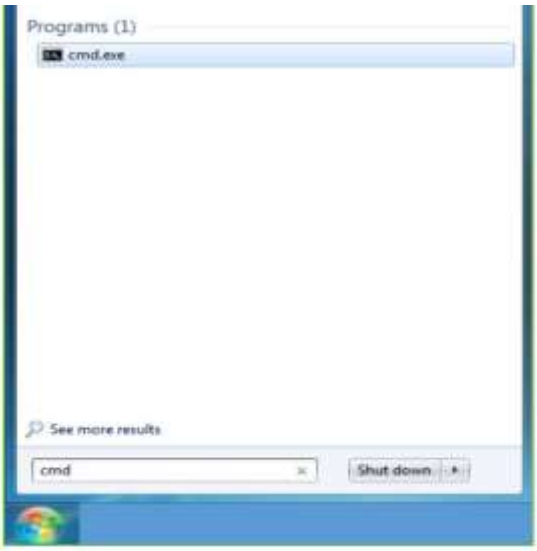
With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

Note: The installation process might take a couple of minutes.

Verify the Python Installation

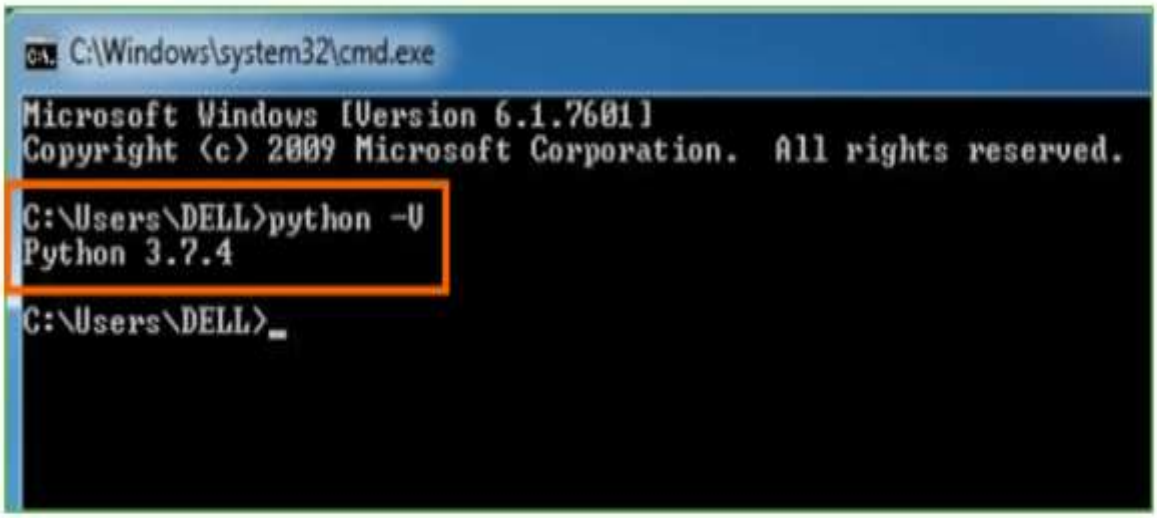
Step 1: Click on Start

Step 2: In the Windows Run Command, type "cmd".



Step 3: Open the Command prompt option.

Step 4: Let us test whether the python is correctly installed. Type **python –V** and press Enter.



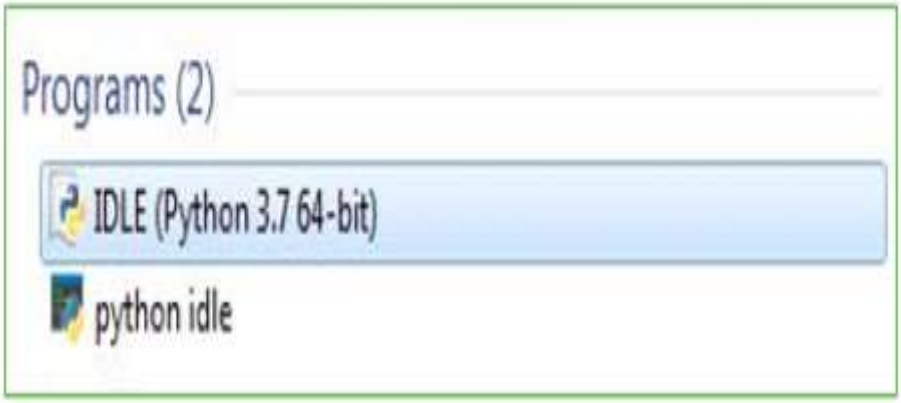
Step 5: You will get the answer as 3.7.4

Note: If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

Check how the Python IDLE works

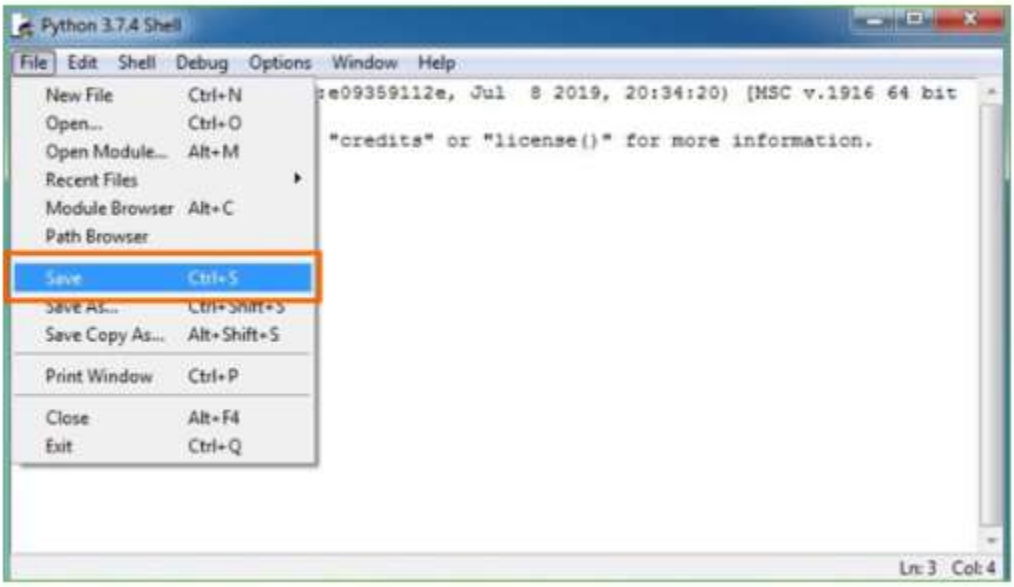
Step 1: Click on Start

Step 2: In the Windows Run command, type "python idle".



Step 3: Click on IDLE (Python 3.7 64-bit) and launch the program

Step 4: To go ahead with working in IDLE you must first save the file. **Click on File > Click on Save**



Step 5: Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World.

Step 6: Now for e.g. **enter print**.

2.5.2. OPERATING SYSTEM :

❖ System Architecture:

- Explain the system architecture supported by Windows, emphasizing whether it uses a monolithic kernel.

- Discuss compatibility with different hardware architectures, such as x86 and ARM.

❖ User Interface:

- Describe the graphical user interface (GUI) provided by Windows, including the Start menu, taskbar, and other key elements.

- Mention any unique features like the Windows Aero interface, Live Tiles, and the Action Center.

❖ File System:

- Discuss the file system structure used by Windows, typically NTFS (New Technology File System).

- Highlight features related to file organization, permissions, and security.

❖ Process and Memory Management:

- Provide an overview of how Windows manages processes and memory.
- Discuss features like the Task Manager, memory protection, and process scheduling.

❖ Security Features:

- Discuss security mechanisms, including user authentication, access control, and encryption.
- Highlight built-in security tools like Windows Defender for antivirus and Windows Firewall.

❖ Networking Capabilities:

- Describe the networking capabilities of Windows, including support for various network protocols and network configuration tools.
- Discuss features like HomeGroup, network discovery, and Remote Desktop.

❖ Software Compatibility:

- Discuss Windows' broad compatibility with software applications, emphasizing its support for a wide range of programming languages and development frameworks.
- Mention compatibility layers, virtualization options, and Windows Subsystem for Linux (WSL).

❖ Updates and Maintenance:

- Discuss how Windows handles updates, patches, and maintenance.
- Explain the Windows Update service, including update frequency and the importance of keeping the system up to date.

❖ User Support and Documentation:

- Mention the availability of user support resources, such as official documentation, forums, and customer support channels.

- Evaluate the quality of available documentation and the user-friendliness of support options.

❖ Performance and Resource Utilization:

- Provide insights into the performance characteristics of Windows, including system requirements for different versions.

- Discuss resource utilization and any performance optimization features.

2.5.3. DATABASE:

When writing a report about SQL databases, it's important to cover various aspects to provide a comprehensive understanding. Here's a structured outline you can follow:

I. Introduction to SQL Databases

1. Definition:

- Explain what SQL databases are.

- Highlight their role in managing and organizing data.

2. Key Characteristics:

- Discuss the characteristics of SQL databases, such as structured data, ACID properties (Atomicity, Consistency, Isolation, Durability), and the use of tables.

II. Basic SQL Concepts

1. Tables and Fields:

- Explain the concept of tables and fields (columns).

- Discuss data types and how they are defined for each field.

2. Queries:

- Introduce SQL queries for data retrieval.

- Include SELECT statements and examples.

3. Data Modification:

- Discuss SQL statements for modifying data.

- Include INSERT, UPDATE, and DELETE statements.

III. Database Design

1. Normalization:

- Explain the concept of normalization and its importance in database design.

- Discuss different normalization forms.

2. Indexes:

- Discuss the role of indexes in optimizing database performance.

- Explain how to create and use indexes.

IV. Database Management Systems (DBMS)

1. Popular DBMS:

- Mention popular relational database management systems like MySQL, PostgreSQL, SQLite, and Microsoft SQL Server.

2. DBMS Features:

- Discuss additional features provided by DBMS, such as user management, security, and backup/restore options.

VI. Challenges and Best Practices

1. Concurrency and Locking:

- Discuss challenges related to concurrent access and locking mechanisms.

- Highlight best practices to manage concurrency.

2. Security:

- Discuss common security considerations, including authentication and authorization.

- Mention best practices for securing databas

3 SOFTWARE AND HARDWARE REQUIREMENTS

3.1. HARDWARE REQUIRMENTS:

- Processor: Pentium-III (or) Higher
- Ram: 64MB (or) Higher
- Hard disk: 80GB (or) Higher

3.2. SOFTWARE REQUIREMENTS:

- Technology: Python Django
- IDE : Pycharm/Atom
- Client Side Technologies: HTML, CSS, JavaScript , Bootstrap
- Server Side Technologies: Python
- Data Base Server: Sqlite
- Operating System: Microsoft Windows/Linux

4. SOFTWARE REQUIREMENT SPECIFICATION

4.1. PROJECT SCOPE:

The project has a wide scope, as it is not intended to a particular organization. This project is going to develop generic software, which can be applied by any businesses organization. More over it provides facility to its users. Also the software is going to provide a huge amount of summary data.

4.2. PROJECT PERSPECTIVE:

This project is aimed at developing a web based Bug Tracking System, which is the perfect or unique solution to track the bugs of a solution, product or an application. Bug tracking system admits single or set of developers to continue track of not finished bugs in their product successfully. Bug tracking system can increases a lot, the accountability and productivity of single employees by giving a positive feedback and back up the workflow. The bug tracking software allows or group of testers or individual testers to keep path of unfixed bugs in their software successfully. The bug tracking software can track bugs, can handle code changes, can share information with teammates, submit and review connects and control standard assurance.

4.3. USER CLASSES AND CHARACTERISTICS:

- Login Class: Used for performing all the operations of the login functionality.
- Page Class: Class for managing all the operations of the page.
- Traffic Class: Class for managing the traffic of the website.
- IP Class: It has been used for storing all the IPs which hits the website.
- Users Class: Class for managing all the user operations.
- Permission Class: This class has been used for managing all the permissionslevel opeations.

4.4. FUNCTIONAL REQUIREMENTS :

- MySQL database has been used for storing the data of the website
- HTML has been used for creating the layout of the web application
- CSS has been used for creating the designing of the webpages
- JavaScript scripting language has been implemented on the system forperforming all of the Client Side Server Validation.
- Should install django in anoconda prompt.

Admin: This module has complete control over all other modules. The administrator establishes the project and assigns it to the newly established manager, adding members to the managers and assigning bugs depending on importance.

Manager: The manager has complete access to the allocated project and is responsible for fixing the assigned bug. The manager's bug list can be viewed by the developer.

Tester: The tester has access to the project or bugs allocated by the management, can see the project, add a new bug to the list, and report the bug to the manager. The tester can access the allocated project list by logging into the system.

Reports: Both the administrator and the manager can access this module and generate reports based on their needs.

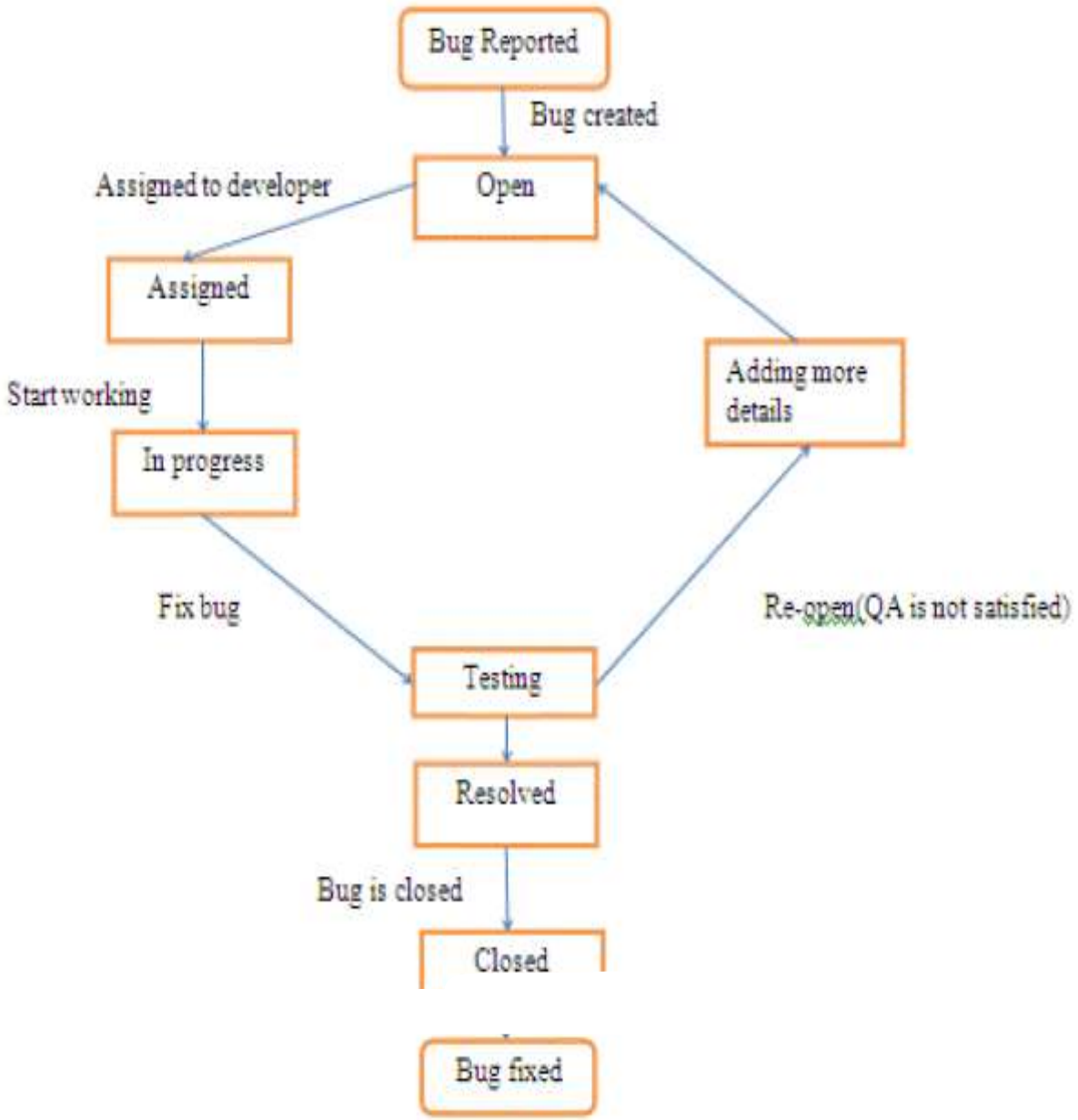
4.5. NON FUNCTIONAL REQUIREMENTS:

NON-FUNCTIONAL REQUIREMENT (NFR) specifies the quality attribute of a software system. They judge the software system based on Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the software system. Example of nonfunctional requirement, *"how fast does the website load?"* Failing to meet non-functional requirements can result in systems that fail to satisfy user needs. Non- functional Requirements allows you to impose constraints or restrictions on the design of the system across the various agile backlogs. Example, the site should load in 3 seconds when the number of simultaneous users are > 10000. Description of non-functional requirements is just as critical as a functional requirement.

- Serviceability requirement
- Manageability requirement
- Recoverability requirement
- Security requirement
- Data Integrity requirement
- Capacity requirement
- Availability requirement
- Scalability requirement
- Interoperability requirement
- Reliability requirement
- Maintainability requirement
- Regulatory requirement
- Environmental requirement

5. SYSTEM DESIGN

5.1. ARCHITECTURAL REPRESENTATION OF PROJECT :



5.2. ARCHITECTURAL DESCRIPTION

1. Client Side (Web Browser):

- The user interacts with the bug tracking system through a web browser.

2. Frontend (Django Templates and JavaScript):

- Django templates are used to render HTML pages.
- JavaScript can be used to enhance the user interface with dynamic features.

3. Web Server (Django Server):

- Handles HTTP requests from the client.
- Routes requests to the appropriate views.

4. Views:

- Process user requests received from the web server.
- Interact with models to fetch or update data.
- Render Django templates for the user interface.

5. Models:

- Define the structure of the database tables.
- Interact with the database to perform CRUD operations.
- Include models for Bug, User, Project, Comment, etc.

6. Database (SQL Database):

- Stores data related to bugs, users, projects, comments, etc.
- Django ORM (Object-Relational Mapping) translates between Python objects and database records.

7. Business Logic:

- Enforces business rules and logic.
- Validates data before saving it to the database.

8. Middleware:

- Additional components that can process requests and responses globally.
- Examples include authentication middleware for user authentication.

9. URL Dispatcher:

- Maps URLs to the appropriate view functions.
- Defines the URL patterns for different pages and actions.

10. Authentication and Authorization:

- Ensures that users are who they claim to be (authentication).
- Controls access to resources based on user roles and permissions (authorization).

11. Static Files (CSS, Images, etc.):

- Static files are served by Django for styling and images.

Data Flow:

1. User Interaction:

- Users interact with the bug tracking system through the web browser by submitting forms, clicking buttons, etc.

2. HTTP Requests:

- The web browser sends HTTP requests to the Django web server.

3. URL Mapping:

- The URL dispatcher maps the incoming requests to the appropriate view functions.

4. Views Processing:

- Views process the requests, perform necessary actions (e.g., fetching or updating data), and render Django templates.

5. Database Interaction:

- Models interact with the database to store or retrieve data.

6. Business Logic:

- Business logic ensures that data meets specified rules and requirements.

7. HTTP Responses:

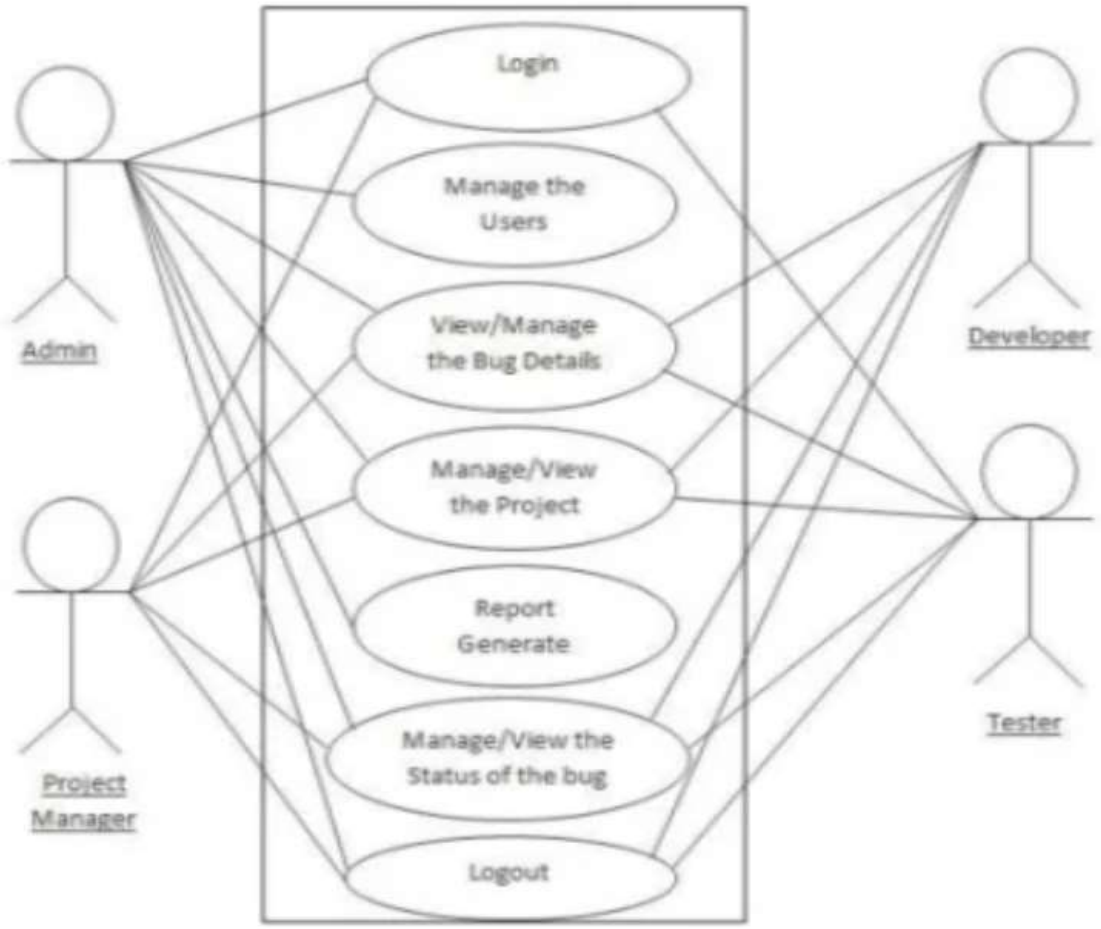
- Views send HTTP responses back to the web browser, which renders the HTML pages.

8. User Interface Updates:

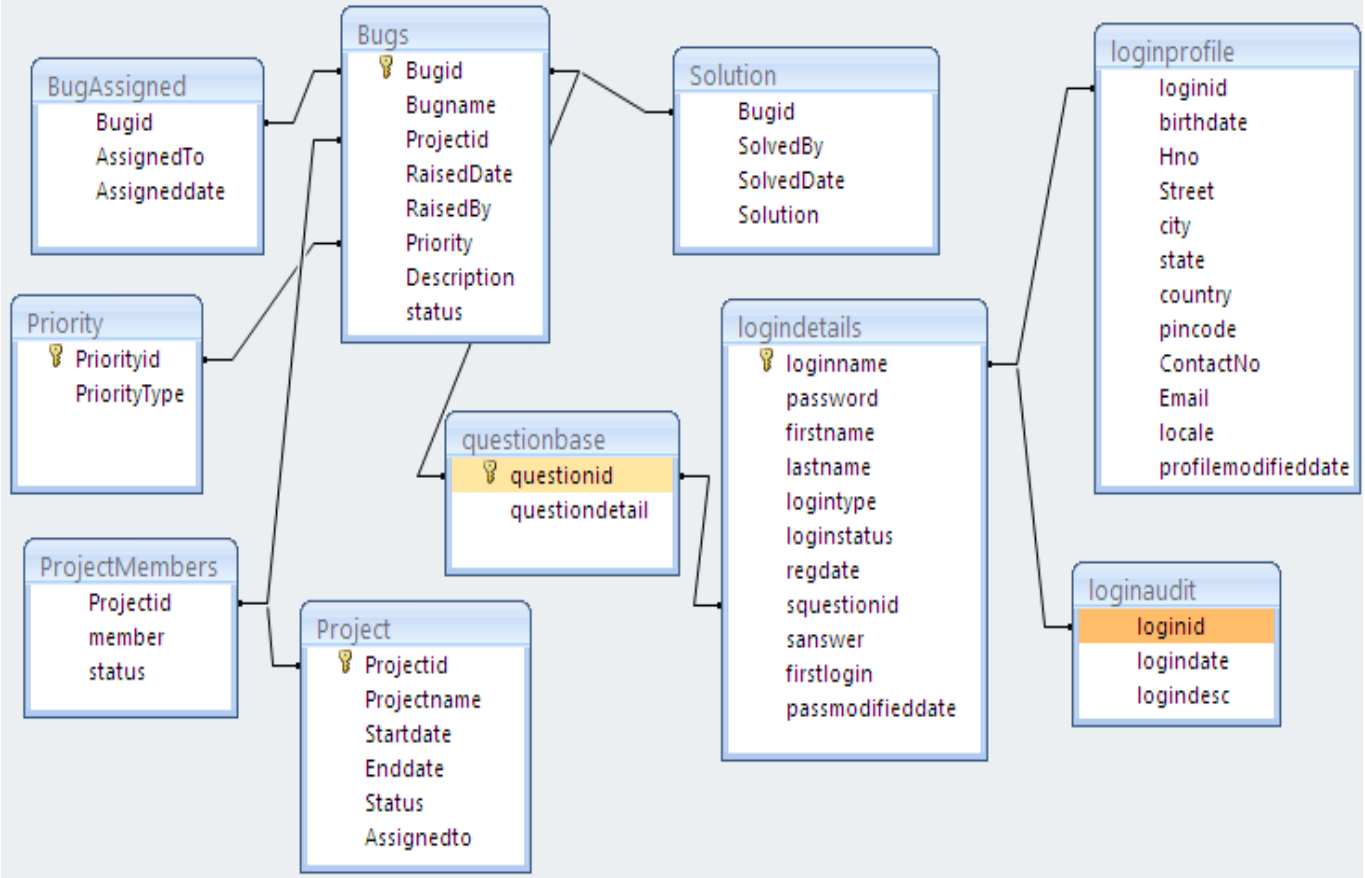
- JavaScript may be used to update the user interface dynamically without requiring a full page reload.

5.3 USE CASE ANALYSIS :

- Use case diagram consists of use cases and actors and shows the interaction between them. The key points are:
- The main purpose is to show the interaction between the use cases and the actor.
- To represent the system requirement from user's perspective.
- The use cases are the functions that are to be performed in the modele.



5.4. CLASS DIAGRAM :



Class Descriptions:

1. User:

- Attributes: id, username, password, email, role
- Methods: (potentially) authenticate(), create_bug()

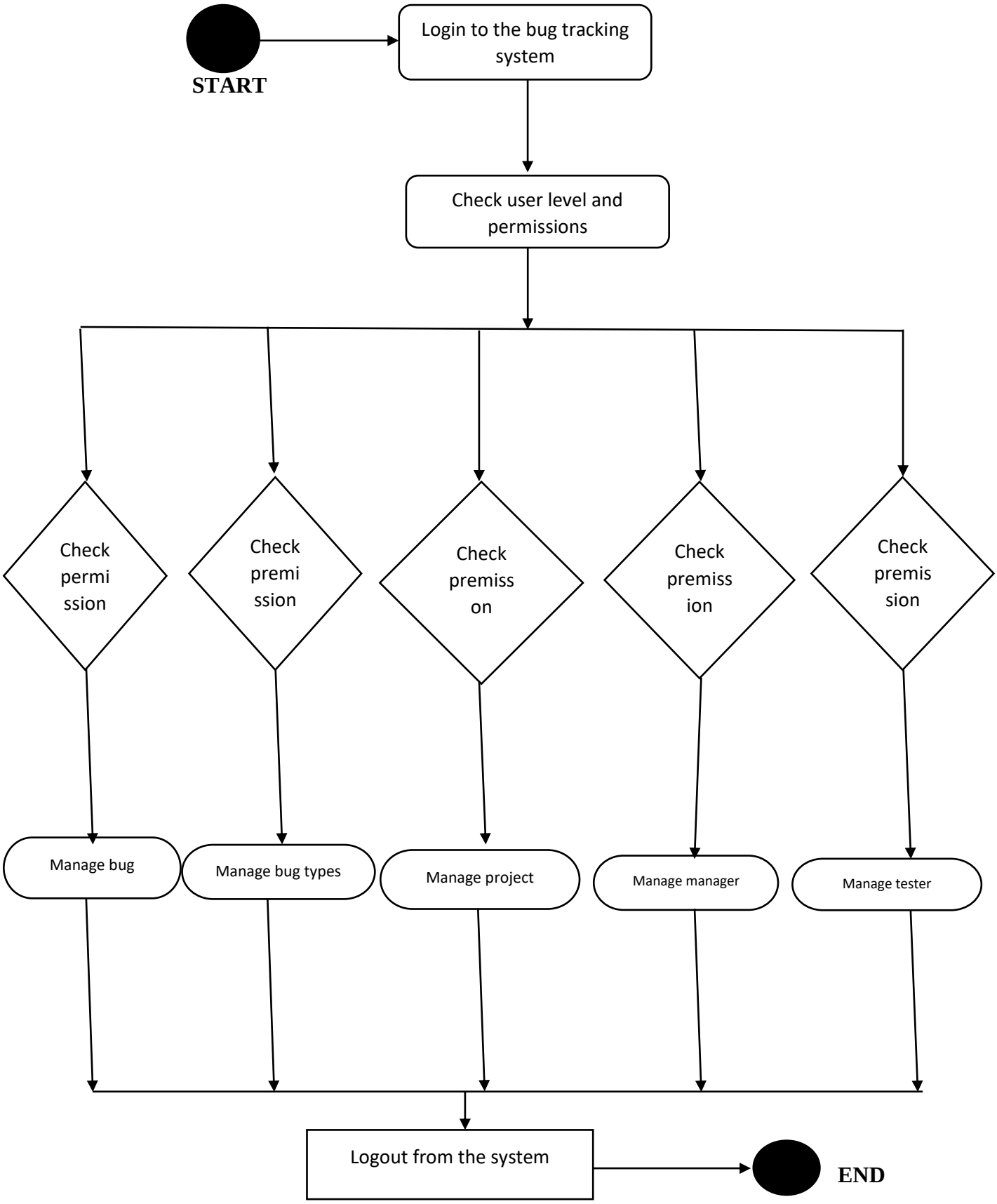
2. Bug:

- Attributes: id, title, description, status, priority, created_by, assigned_to, created_at, updated_at
- Methods: (potentially) add_comment(), change_status()

3. Comment:

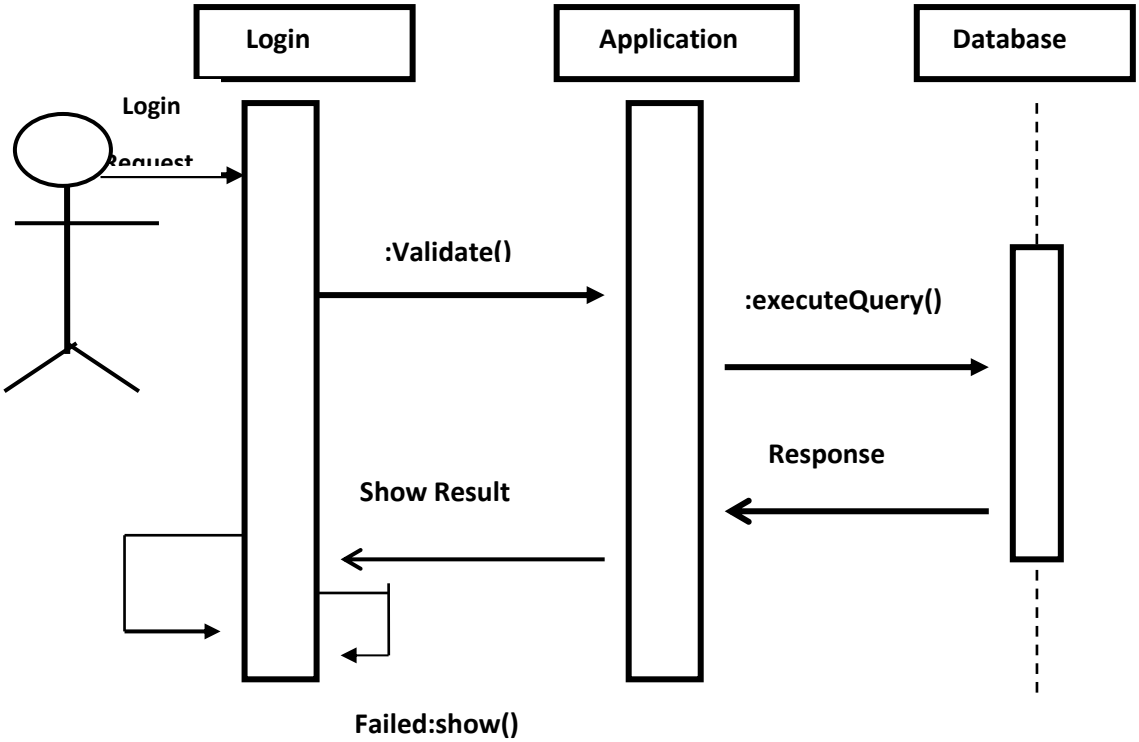
- Attributes: id, text, created_at
- Associations: user (ForeignKey), bug (ForeignKey)

5.5. ACTIVITY DIAGRAM :

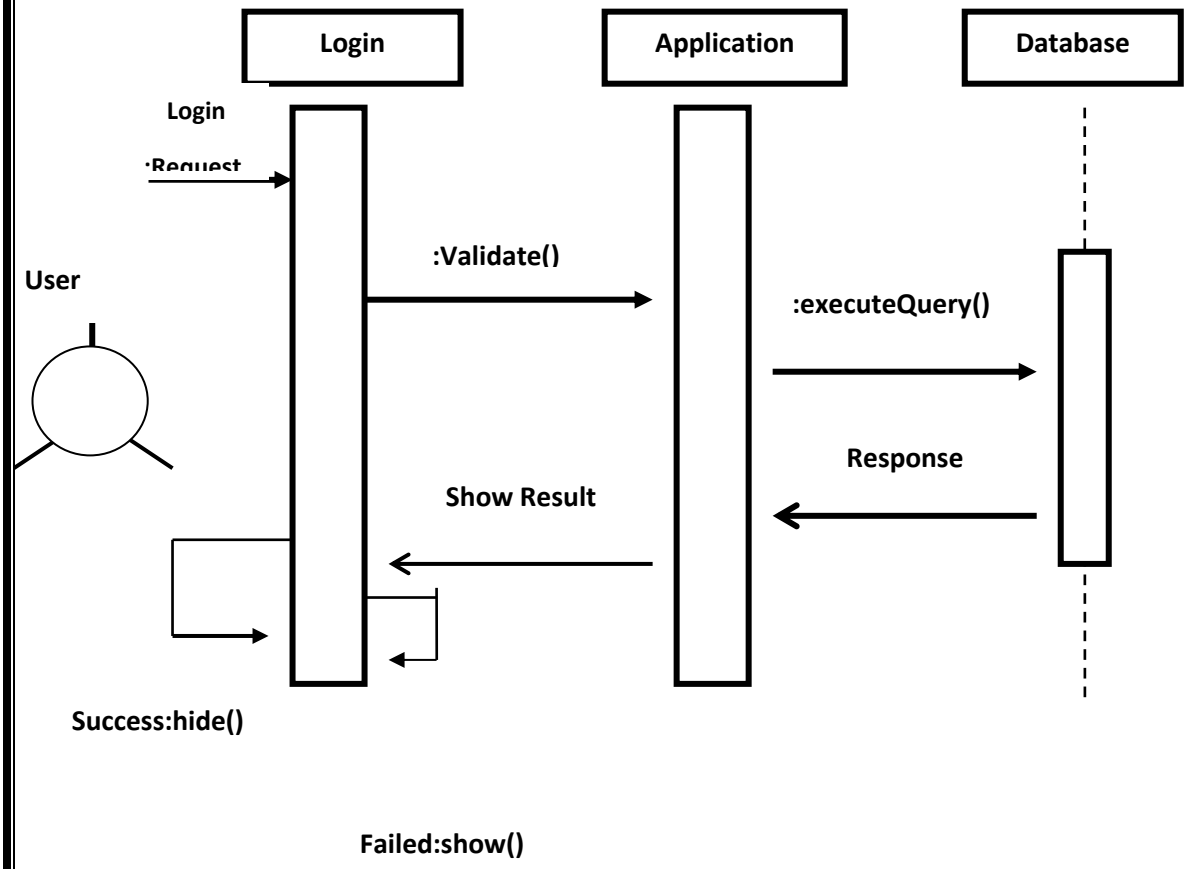


5.6. SEQUENCE DIAGRAM

Sequence Diagram For Administrator :-

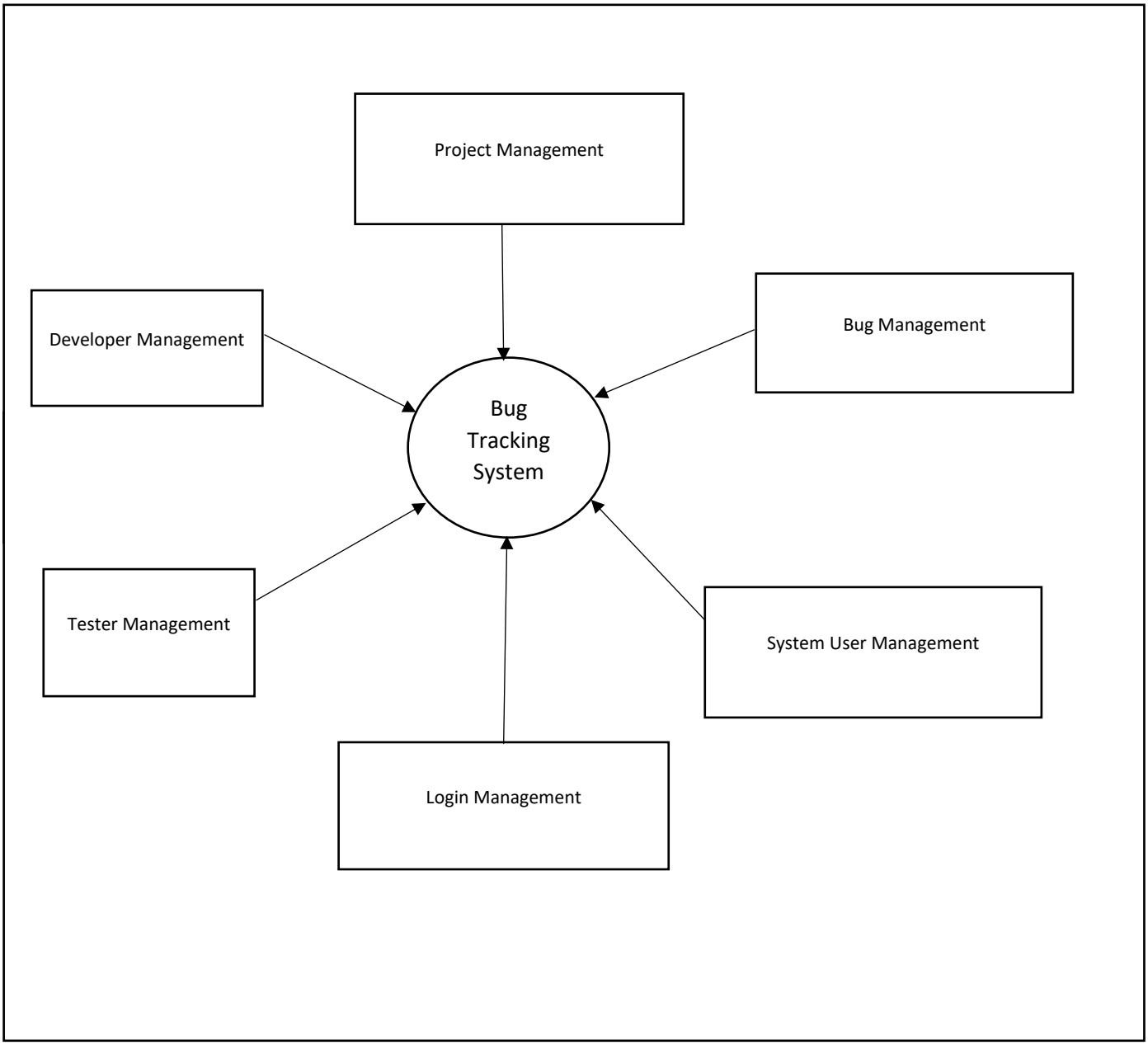


Sequence Diagram For User :-

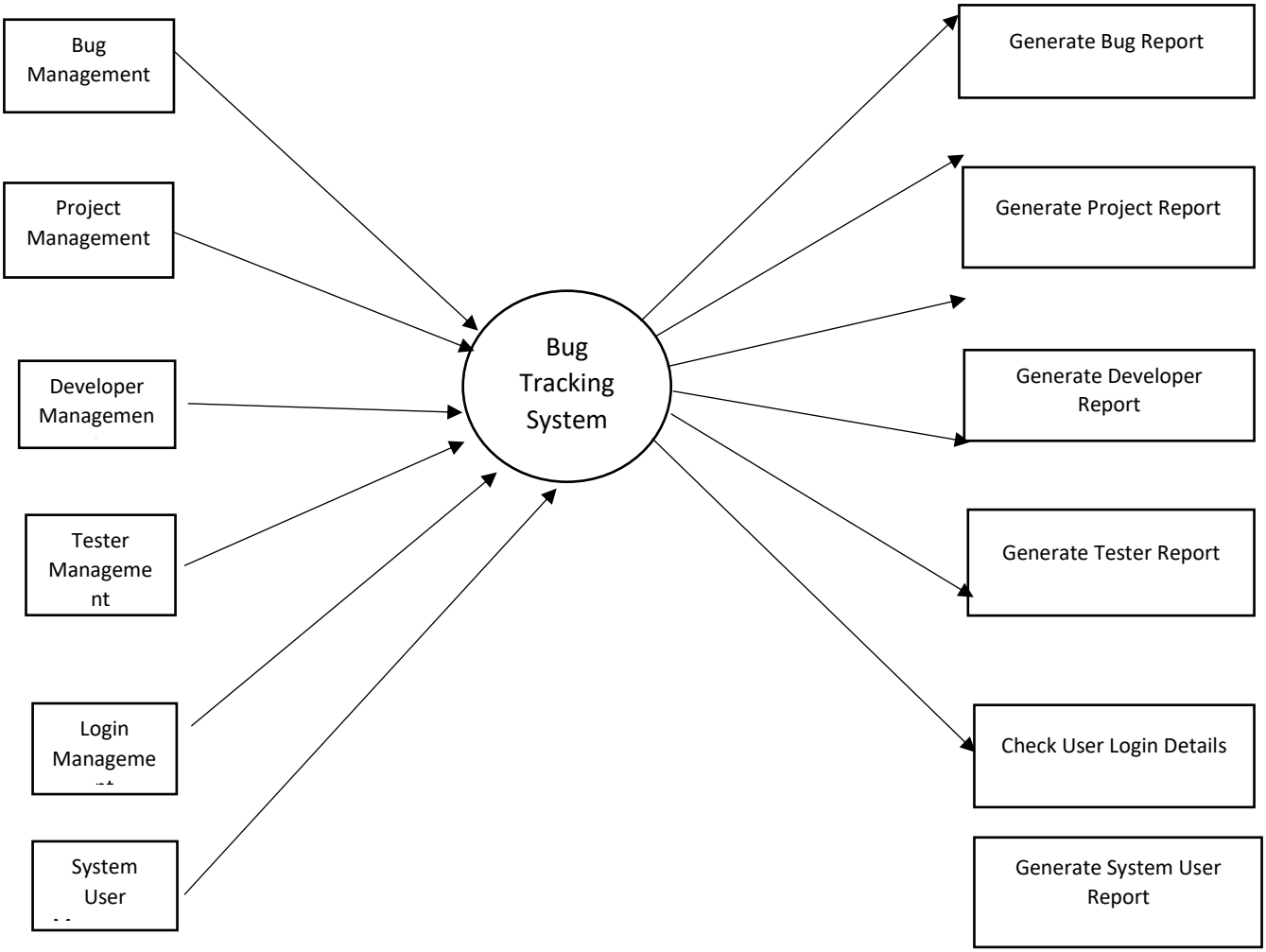


5.7 DATA FLOW DIAGRAM :

DFD Level 0

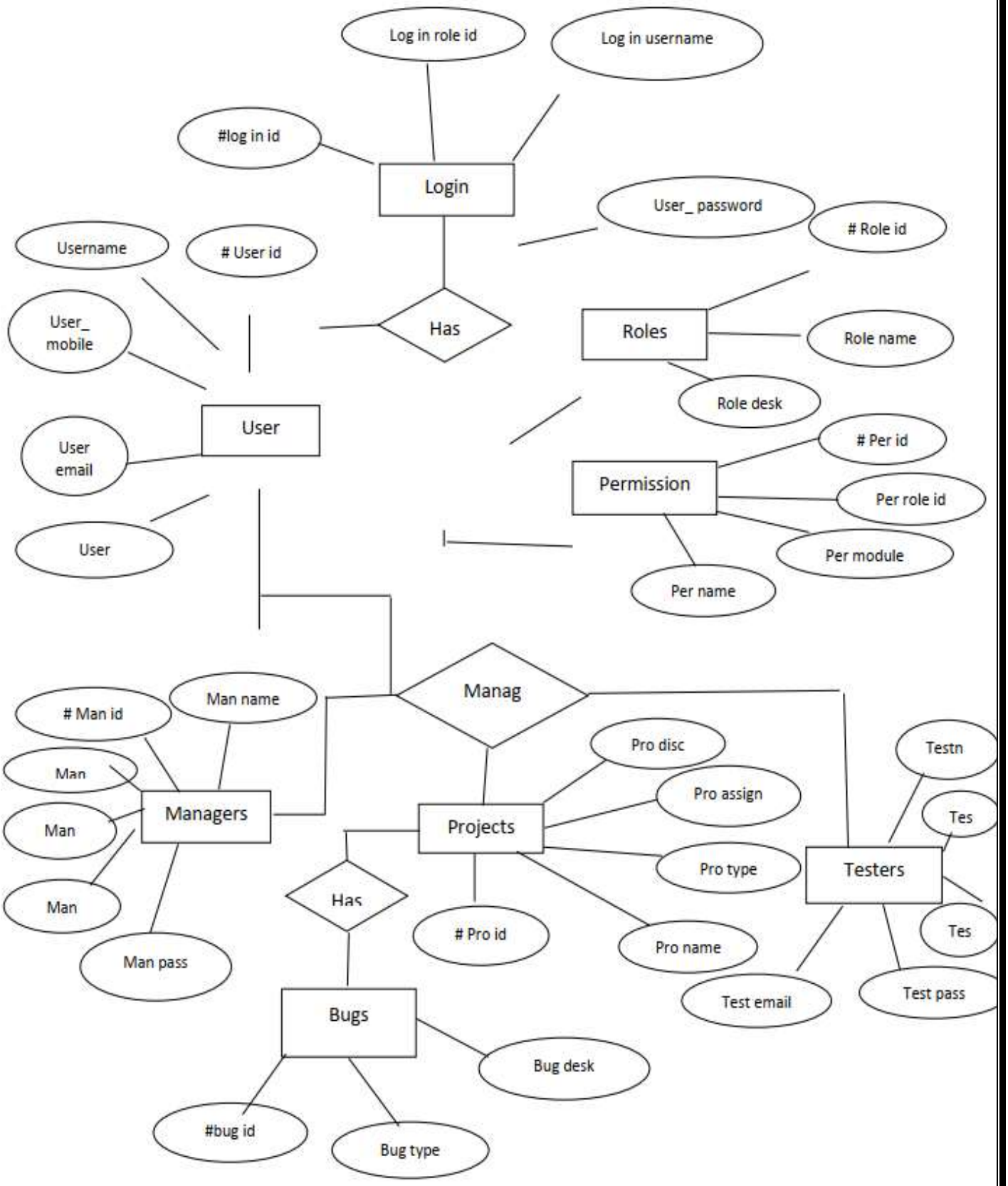


DFD Level 1


Bug Management
Project Management
Developer Managemen
Tester Manageme nt
Login Manageme
System User
Bug Tracking System
Generate Bug Report
Generate Project Report
Generate Developer Report
Generate Tester Report
Check User Login Details
Generate System User Report

6. DETAILED DESIGN

6.1. ER DIAGRAM



6.2. DATABASE DESIGN

1. Entities and Attributes

- Bug Entity

- Attributes:

- BugID (Primary Key, Integer)

- Title (String, 255 characters)

- Description (Text)

- Status (String, choices=['Open', 'In Progress', 'Closed'])

- Priority (String, choices=['Low', 'Medium', 'High'])

- CreatedBy (Foreign Key to User)

- AssignedTo (Foreign Key to User, Nullable)

- CreatedAt (DateTime)

- UpdatedAt (DateTime)

- User Entity

- Attributes:

- UserID (Primary Key, Integer)

- Username (String, 50 characters)

- Email (String, 255 characters)

- Password (String, encrypted)

- Role (String, choices=['Admin', 'Developer', 'Tester'])

2. Relationships

- Bug - User Relationship

- Cardinality: Many-to-One (Multiple bugs can be created by one user)

- Participation: Mandatory on Bug side (Each bug must have a creator)

- Bug - User Assignment Relationship

- Cardinality: Many-to-One (Multiple bugs can be assigned to one user)

- Participation: Optional on Bug side (A bug may not be assigned to anyone)

3. Normalization

Describe the normalization process to eliminate redundancy and improve data integrity. Discuss how the design adheres to at least 3rd normal form.

4. Data Integrity

Explain how data integrity is maintained using primary keys, foreign keys, and other constraints. For example, the Bug-User relationship ensures that each bug has a valid creator.

5. Indexing

Discuss the indexing strategy, including any indexes applied to enhance query performance. Consider indexing fields used frequently in search or filter operations.

- Use diagrams, such as ER diagrams, to visually represent the relationships between entities.

- Provide sample SQL queries to illustrate common database operations.

- Consider the scalability and performance implications of your design choices.

- Review and refine the report based on feedback from stakeholders or team members.

7. IMPLEMENTATION

7.1. SYSTEM IMPLEMENTION (SAMPLE CODE):

urls.py

```
from django.contrib import admin

from django.urls import path

from bugapp.views import *

from django.conf import settings

from django.conf.urls.static import static

urlpatterns = [

    path('admin/', admin.site.urls),

    path('', home, name='home'),

    path('login-user', login_user, name='login_user'),

    path('logout-user', logout_user, name='logout_user'),

    path('add-new-bug', add_new_bug, name='add_new_bug'),

    path('add-new-project', add_new_project, name='add_new_project'),

    path('add-new-user', add_new_user, name='add_new_user'),

    path('bug-report', bug_report, name='bug_report'),

    path('project-report', project_report, name='project_report'),

    path('user-report', user_report, name='user_report'),

    path('change-password', Change_Password, name='Change_Password'),

    path('profile', profile, name='profile'),

    path('edit-user/<int:pid>', edit_user, name='edit_user'),

    path('delete-user/<int:pid>', delete_user, name='delete_user'),
```

```
    path('edit-project/<int:pid>', edit_project, name='edit_project'),
    path('delete-project/<int:pid>', delete_project, name='delete_project'),
    path('edit-bug/<int:pid>', edit_bug, name='edit_bug'),
    path('delete-bug/<int:pid>', delete_bug, name='delete_bug'),
    path('bug-detail/<int:pid>', bug_detail, name='bug_detail'),
]+static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

View.py

```
from django.contrib.auth.decorators import login_required
from django.shortcuts import render, redirect
from django.contrib.auth import authenticate, logout, login
from django.contrib import messages
from .models import *
# Create your views here.

def home(request):
    return render(request, 'home.html')

def login_user(request):
    if request.method == "POST":
        uname = request.POST['name']
        pwd = request.POST['password']
        user = authenticate(username=uname, password=pwd)
        if user:
            login(request, user)
```

```
            messages.success(request, "Login Successfully")
            return redirect('home')
    return render(request, 'login.html')


def logout_user(request):
    logout(request)
    messages.success(request, "Logout Successfully")
    return redirect('home')


def add_new_bug(request):
    if request.method == "POST":
        name = request.POST['name']
        project = request.POST['project']
        tester_code = request.POST['tester_code']
        bug_date = request.POST['bug_date']
        bug_level = request.POST['bug_level']
        bug_priority = request.POST['bug_priority']
        bug_type = request.POST['bug_type']
        status_name = request.POST['status_name']
        description = request.POST['description']
        proj_obj = Project.objects.get(id=project)
        creator = UserProfile.objects.get(user=request.user)
        bug = Bug.objects.create(title=name, description=description, project=proj_obj, creator=creator,
status=status_name, bug_type=bug_type, created=bug_date, priority=bug_priority, bug_level=bug_level,
tester_code=tester_code)
        messages.success(request, "Added Successfully")
```

```
        return redirect('bug_report')
    project = Project.objects.all()
    return render(request, 'add_new_bug.html', {'project': project, 'bug_type': BUGTYPE, 'bug_status':
BUGSTATUS})


def add_new_project(request):
    if request.method == "POST":
        project_name = request.POST['project_name']
        project_duration = request.POST['project_duration']
        submission_date = request.POST['submission_date']
        client_name = request.POST['client_name']
        client_address = request.POST['client_address']
        phone_number = request.POST['phone_number']
        email = request.POST['email']
        department_name = request.POST['department_name']
        project_lead = request.POST['project_lead']
        description = request.POST['description']
        proj_lead = UserProfile.objects.get(id=project_lead)
        bug = Project.objects.create(name=project_name, description=description,
submission_date=submission_date, duration=project_duration, client_name=client_name,
client_address=client_address, phone_number=phone_number, email_id=email,
department_name=department_name, project_lead=proj_lead)
        messages.success(request, "Added Successfully")
        return redirect('project_report')
    project = Project.objects.all()
    user = UserProfile.objects.all()
    return render(request, 'add_new_project.html', {'project': project, 'user': user})
```

```
def add_new_user(request):
    if request.method == "POST":
        userrole = request.POST['userrole']
        user_name = request.POST['user_name']
        # confirm = request.POST['confirm']
        password = request.POST['password']
        first_name = request.POST['first_name']
        last_name = request.POST['last_name']
        email = request.POST['email']
        phone_number = request.POST['phone_number']
        gender = request.POST['gender']
        file = request.FILES['file']
        address = request.POST['address']
        dob = request.POST['dob']
        user = User.objects.create_user(username=user_name, first_name=first_name,
last_name=last_name, password=password, email=email)
        bug = UserProfile.objects.create(user=user, userrole=userrole, contact=phone_number,
gender=gender, file=file, address=address, dob=dob)
        messages.success(request, "Added Successfully")
        return redirect('user_report')
    return render(request, 'add_new_user.html', {'userrole': USERROLE})


def bug_report(request):
    bug = Bug.objects.all()
    return render(request, 'bug_report.html', {'bug': bug})
```

```
def project_report(request):
    project = Project.objects.all()
    return render(request, 'project_report.html', {'project': project})



def user_report(request):
    user = UserProfile.objects.all()
    return render(request, 'user_report.html', {'user': user})


@login_required(login_url="login")
def Change_Password(request):
    if request.method=="POST":
        n = request.POST['pwd1']
        c = request.POST['pwd2']
        o = request.POST['pwd3']
        if c == n:
            u = User.objects.get(username__exact=request.user.username)
            u.set_password(n)
            u.save()
            messages.success(request, "Changed Successfully")
            logout(request)
            return redirect('home')
        else:
            messages.success(request, "Password not matching")
    return render(request, 'change_password.html')
```

```
def profile(request):
    data = UserProfile.objects.get(user=request.user)
    if request.method == "POST":
        userrole = request.POST['userrole']
        user_name = request.POST['user_name']
        # confirm = request.POST['confirm']
        # password = request.POST['password']
        first_name = request.POST['first_name']
        last_name = request.POST['last_name']
        email = request.POST['email']
        phone_number = request.POST['phone_number']
        gender = request.POST['gender']
        address = request.POST['address']
        dob = request.POST['dob']
        user = User.objects.filter(username=user_name).update(first_name=first_name,
last_name=last_name, email=email)
        bug = UserProfile.objects.filter(user=user).update(userrole=userrole, contact=phone_number,
gender=gender, address=address, dob=dob)
        try:
            file = request.FILES['file']
            UserProfile.objects.filter(user=user).update(file=file)
        except:
            pass
        messages.success(request, "Updated Successfully")
        return redirect('profile')
```

```
    return render(request, 'profile.html', {'data': data, 'userrole': USERROLE})


def edit_user(request,pid):
    data = UserProfile.objects.get(id=pid)
    if request.method == "POST":
        userrole = request.POST['userrole']
        user_name = request.POST['user_name']
        # confirm = request.POST['confirm']
        # password = request.POST['password']
        first_name = request.POST['first_name']
        last_name = request.POST['last_name']
        email = request.POST['email']
        phone_number = request.POST['phone_number']
        gender = request.POST['gender']
        address = request.POST['address']
        dob = request.POST['dob']
        user = User.objects.filter(username=user_name).update(first_name=first_name,
last_name=last_name, email=email)
        bug = UserProfile.objects.filter(user=user).update(userrole=userrole, contact=phone_number,
gender=gender, address=address, dob=dob)
        try:
            file = request.FILES['file']
            UserProfile.objects.filter(user=user).update(file=file)
        except:
            pass
        messages.success(request, "Updated Successfully")
```

```
        return redirect('user_report')
    return render(request, 'edit_user.html', {'data': data, 'userrole': USERROLE})


def delete_user(request, pid):
    data = UserProfile.objects.get(id=pid)
    data.delete()
    messages.success(request, "Deleted Successfully")
    return redirect('user_report')


def edit_project(request,pid):
    data = Project.objects.get(id=pid)
    if request.method == "POST":
        project_name = request.POST['project_name']
        project_duration = request.POST['project_duration']
        submission_date = request.POST['submission_date']
        client_name = request.POST['client_name']
        client_address = request.POST['client_address']
        phone_number = request.POST['phone_number']
        email = request.POST['email']
        department_name = request.POST['department_name']
        project_lead = request.POST['project_lead']
        description = request.POST['description']


        proj_lead = UserProfile.objects.get(id=project_lead)
        bug = Project.objects.filter(id=pid).update(name=project_name, description=description,
submission_date=submission_date,
```

```
                                duration=project_duration, client_name=client_name,
client_address=client_address,
                                phone_number=phone_number, email_id=email,
department_name=department_name,
                                project_lead=proj_lead)
        messages.success(request, "Updated Successfully")
        return redirect('project_report')
    user = UserProfile.objects.all()
    return render(request, 'edit_project.html', {'data': data, 'user': user})


def delete_project(request, pid):
    data = Project.objects.get(id=pid)
    data.delete()
    messages.success(request, "Deleted Successfully")
    return redirect('project_report')


def edit_bug(request,pid):
    data = Bug.objects.get(id=pid)
    if request.method == "POST":
        name = request.POST['name']
        project = request.POST['project']
        tester_code = request.POST['tester_code']
        bug_date = request.POST['bug_date']
        bug_level = request.POST['bug_level']
        bug_priority = request.POST['bug_priority']
        bug_type = request.POST['bug_type']
```

```
        status_name = request.POST['status_name']
        description = request.POST['description']
        proj_obj = Project.objects.get(id=project)
        creator = UserProfile.objects.get(user=request.user)
        bug = Bug.objects.filter(id=pid).update(title=name, description=description, project=proj_obj,
creator=creator,
                                status=status_name, bug_type=bug_type, created=bug_date, priority=bug_priority,
                                bug_level=bug_level, tester_code=tester_code)

        messages.success(request, "Updated Successfully")
        return redirect('bug_report')
    project = Project.objects.all()
    return render(request, 'edit_bug.html', {'data': data, 'project': project, 'bug_type': BUGTYPE,
'bug_status': BUGSTATUS})

def delete_bug(request, pid):
    data = Bug.objects.get(id=pid)
    data.delete()
    messages.success(request, "Deleted Successfully")
    return redirect('bug_report')

def bug_detail(request, pid):
    data = Bug.objects.get(id=pid)
    if request.method == "POST":
        title = request.POST['title']
        description = request.POST['description']
```

```
        comment = Comment.objects.create(user=UserProfile.objects.get(user=request.user), title=title,
message=description, bug=data)
        messages.success(request, "Comment Successfully")
        return redirect('bug_report')
    comment2 = Comment.objects.filter(bug=data)
    return render(request, 'bug_detail.html', {'data': data, 'comment': comment2})
```

<Home : template>

```
{% extends 'index.html' %}
{% load static %}
{% block body %}


<!-- ======= Hero Section ======= -->
<section id="hero" class="d-flex align-items-center">
    <div class="container" data-aos="zoom-out" data-aos-delay="100">
      <h1 style="background:black;width:70%">Welcome to <span>Bug Tracking
System</span></h1>
      <h2 style="background:aqua;width:70%">We are team of talented designers making websites
with Bootstrap</h2>
      <div class="d-flex">
        <a href="#about" class="btn-get-started scrollto">Get Started</a>
        <a href="https://www.youtube.com/watch?v=jDDaplaOz7Q" class="glightbox btn-watch-
video"><i class="bi bi-play-circle"></i><span>Watch Video</span></a>
      </div>
    </div>
</section><!-- End Hero -->


<main id="main">
```

```
<!-- ======= Featured Services Section ======= -->
<section id="featured-services" class="featured-services">
  <div class="container" data-aos="fade-up">

    <div class="row">
      <div class="col-md-6 col-lg-3 d-flex align-items-stretch mb-5 mb-lg-0">
        <div class="icon-box" data-aos="fade-up" data-aos-delay="100">
          <div class="icon"><i class="bx bxl-dribbble"></i></div>
          <h4 class="title"><a href="">Bug Report</a></h4>
          <p class="description">Provide Bug Report to development team.</p>
        </div>
      </div>

      <div class="col-md-6 col-lg-3 d-flex align-items-stretch mb-5 mb-lg-0">
        <div class="icon-box" data-aos="fade-up" data-aos-delay="200">
          <div class="icon"><i class="bx bx-file"></i></div>
          <h4 class="title"><a href="">Project Report</a></h4>
          <p class="description">Provide Project Report to development team.</p>
        </div>
      </div>

      <div class="col-md-6 col-lg-3 d-flex align-items-stretch mb-5 mb-lg-0">
        <div class="icon-box" data-aos="fade-up" data-aos-delay="300">
          <div class="icon"><i class="bx bx-tachometer"></i></div>
          <h4 class="title"><a href="">User Report</a></h4>
```

```
            <p class="description">Provide Project Report to Organisational team.</p>
          </div>
        </div>

        <div class="col-md-6 col-lg-3 d-flex align-items-stretch mb-5 mb-lg-0">
          <div class="icon-box" data-aos="fade-up" data-aos-delay="400">
            <div class="icon"><i class="bx bx-world"></i></div>
            <h4 class="title"><a href="">Department Manage</a></h4>
            <p class="description">Manage Department Easily through bug tracking system.</p>
          </div>
        </div>

      </div>

    </div>

  </section><!-- End Featured Services Section -->

  <!-- ======= About Section ======= -->
  <section id="about" class="about section-bg">
    <div class="container" data-aos="fade-up">

      <div class="section-title">
        <h2>About</h2>
        <h3>Find Out More <span>About Us</span></h3>
        <p>A bug tracking system is software that keeps track of bugs that the user encountered in any software development or in any project.</p>
```

```
</div>    <div class="row">
  <div class="col-lg-6" data-aos="fade-right" data-aos-delay="100">
    <img src="{% static 'assets/img/about.jpg' %}" class="img-fluid" alt="">
  </div>
  <div class="col-lg-6 pt-4 pt-lg-0 content d-flex flex-column justify-content-center" data-aos="fade-up" data-aos-delay="100">
    <h3>What is Bug tracking system?</h3>
    <p class="fst-italic">
      A bug tracking system is software that keeps track of bugs that the user encountered in any software development or in any project.
    </p>
    <ul>
      <li>
        <i class="bx bx-store-alt"></i>
        <div>
          <h5>Creating a new text file and writing the details entered by the user into the text file.</h5>
          <p>Option to change the status of the bug.</p>
        </div>
      </li>
      <li>
        <i class="bx bx-images"></i>
        <div>
          <h5>Report of specific bug file.</h5>
          <p>Switch Statements are used to Switch into the functionalities as preferred by the user.</p>
        </div>
```

```
            </li>
          </ul>
          <p>
            Driver Function: The idea is to keep a variable id that stores the id of the Bugs that are registered till now. There are mainly three options out of which user can select the functionality:
<!-- ======= Counts Section ======= -->
    <section id="counts" class="counts">
      <div class="container" data-aos="fade-up">

        <div class="row">

          <div class="col-lg-3 col-md-6">
            <div class="count-box">
              <i class="bi bi-emoji-smile"></i>
              <span data-purecounter-start="0" data-purecounter-end="232" data-purecounter-duration="1" class="purecounter"></span>
              <p>Happy Clients</p>
            </div>
          </div>

          <div class="col-lg-3 col-md-6 mt-5 mt-md-0">
            <div class="count-box">
              <i class="bi bi-journal-richtext"></i>
              <span data-purecounter-start="0" data-purecounter-end="521" data-purecounter-duration="1" class="purecounter"></span>
              <p>Projects</p>
            </div>
```

```
          </div>


          <div class="col-lg-3 col-md-6 mt-5 mt-lg-0">
            <div class="count-box">
              <i class="bi bi-headset"></i>
              <span data-purecounter-start="0" data-purecounter-end="1463" data-purecounter-
duration="1" class="purecounter"></span>
              <p>Hours Of Support</p>
            </div>
          </div>


          <div class="col-lg-3 col-md-6 mt-5 mt-lg-0">
            <div class="count-box">
              <i class="bi bi-people"></i>
              <span data-purecounter-start="0" data-purecounter-end="15" data-purecounter-duration="1"
class="purecounter"></span>
              <p>Hard Workers</p>
            </div>
          </div>


        </div>


      </div>

    </section><!-- End Counts Section -->
```

```
<!-- ======= Contact Section ======= -->
<section id="contact" class="contact">
  <div class="container" data-aos="fade-up">


    <div class="section-title">
      <h2>Contact</h2>
      <h3><span>Contact Us</span></h3>
      <p>A bug tracking system is software that keeps track of bugs that the user encountered in any software development or in any project.</p>
    </div>


    <div class="row" data-aos="fade-up" data-aos-delay="100">
      <div class="col-lg-6">
        <div class="info-box mb-4">
          <i class="bx bx-map"></i>
          <h3>Our Address</h3>
          <p>ABC Academy Sector C, Delhi</p>
        </div>
      </div>


      <div class="col-lg-3 col-md-6">
        <div class="info-box  mb-4">
          <i class="bx bx-envelope"></i>
          <h3>Email Us</h3>
          <p>contact@example.com</p>
        </div>
```

```
        </div>


        <div class="col-lg-3 col-md-6">
          <div class="info-box  mb-4">
            <i class="bx bx-phone-call"></i>
            <h3>Call Us</h3>
            <p>+1 5589 55488 55</p>
          </div>
        </div>
  </div>
  <div class="row" data-aos="fade-up" data-aos-delay="100">
    <div class="col-lg-6 ">
                    <iframe
src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d651854.4073493392!2d76.8725
0558738809!3d28.734283373790067!2m3!1f0!2f0!3f0!3m2!1i1024!2i768!4f13.1!3m3!1m2!1s0x390
cfd5b347eb62d%3A0x37205b715389640!2sDelhi!5e0!3m2!1sen!2sin!4v1647883332962!5m2!1sen!
2sin" style="border:0; width: 100%; height: 384px;" allowfullscreen="" loading="lazy"></iframe>
        </div>
    <div class="col-lg-6">
          <form action="forms/contact.php" method="post" role="form" class="php-email-form">
            <div class="row">
              <div class="col form-group">
                <input type="text" name="name" class="form-control" id="name" placeholder="Your
Name" required>
              </div>
              <div class="col form-group">
                <input type="email" class="form-control" name="email" id="email" placeholder="Your
Email" required>
```

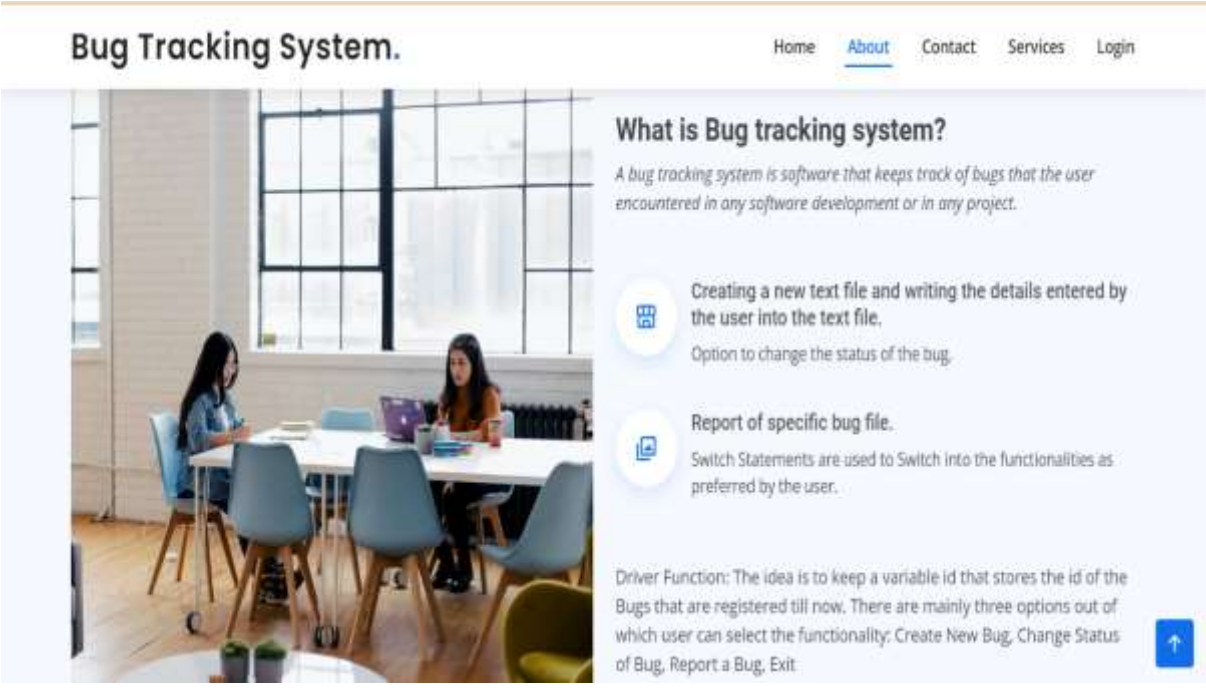
```
            </div>
          </div>
          <div class="form-group">
            <input type="text" class="form-control" name="subject" id="subject"
placeholder="Subject" required>
          </div>
          <div class="form-group">
            <textarea class="form-control" name="message" rows="5" placeholder="Message"
required></textarea>
          </div>
          <div class="my-3">
            <div class="loading">Loading</div>
            <div class="error-message"></div>
            <div class="sent-message">Your message has been sent. Thank you!</div>
          </div>
          <div class="text-center"><button type="submit">Send Message</button></div>
        </form>
      </div>
    </div>
  </div>
  </section><!-- End Contact Section -->

 </main><!-- End #main -->
{% endblock %}
```

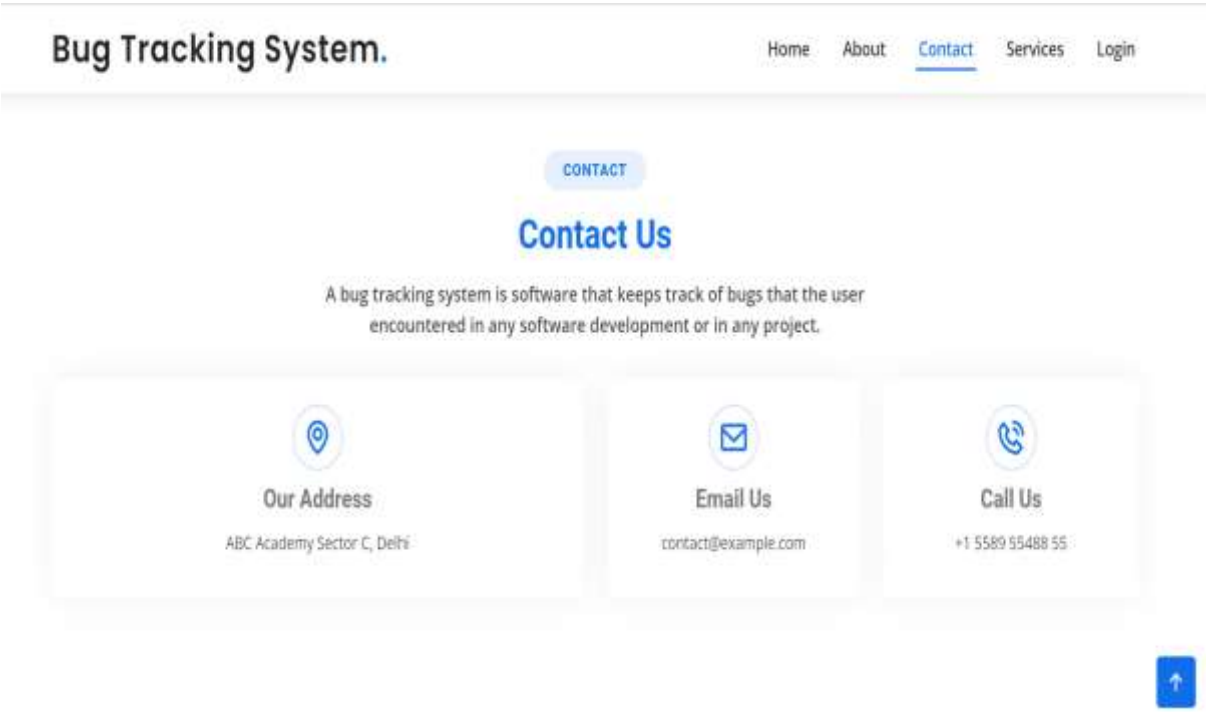
7.2 SCREEN DESIGNS(SCREENSHOTS) :

1. **Home page:**

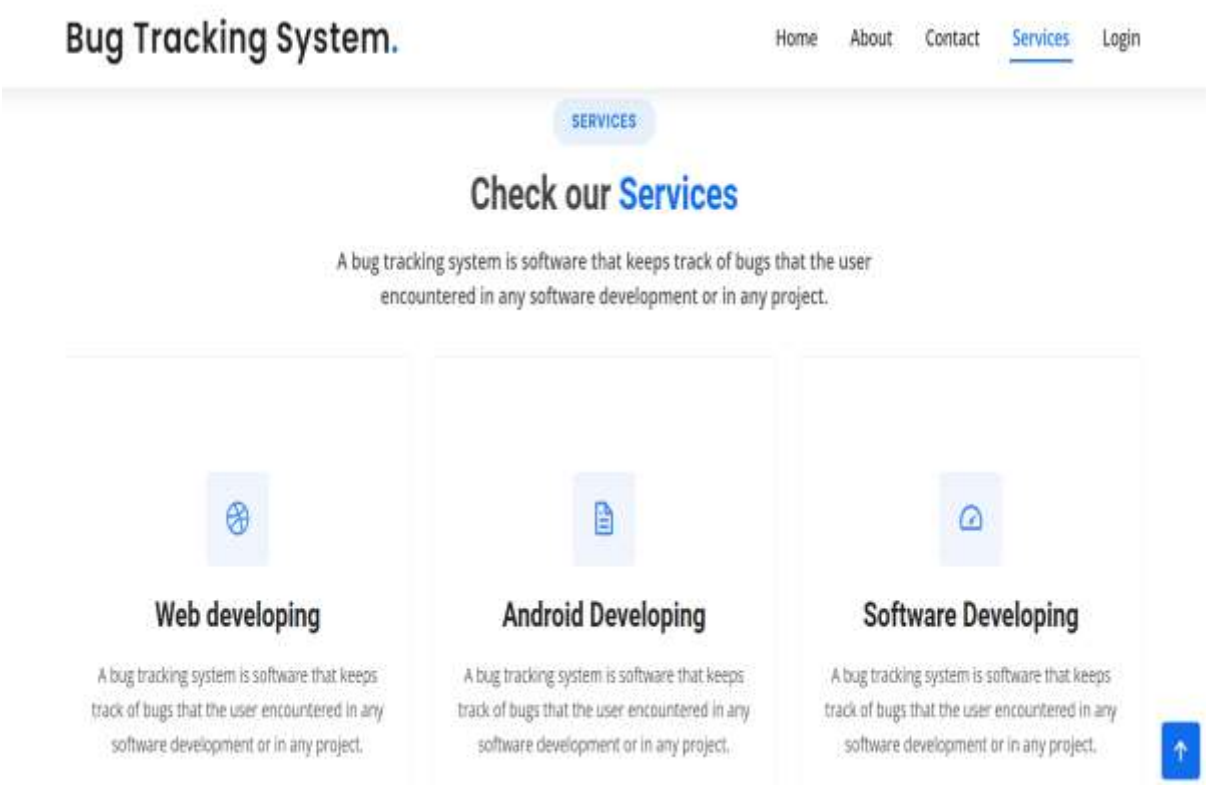
2. **About Page:**



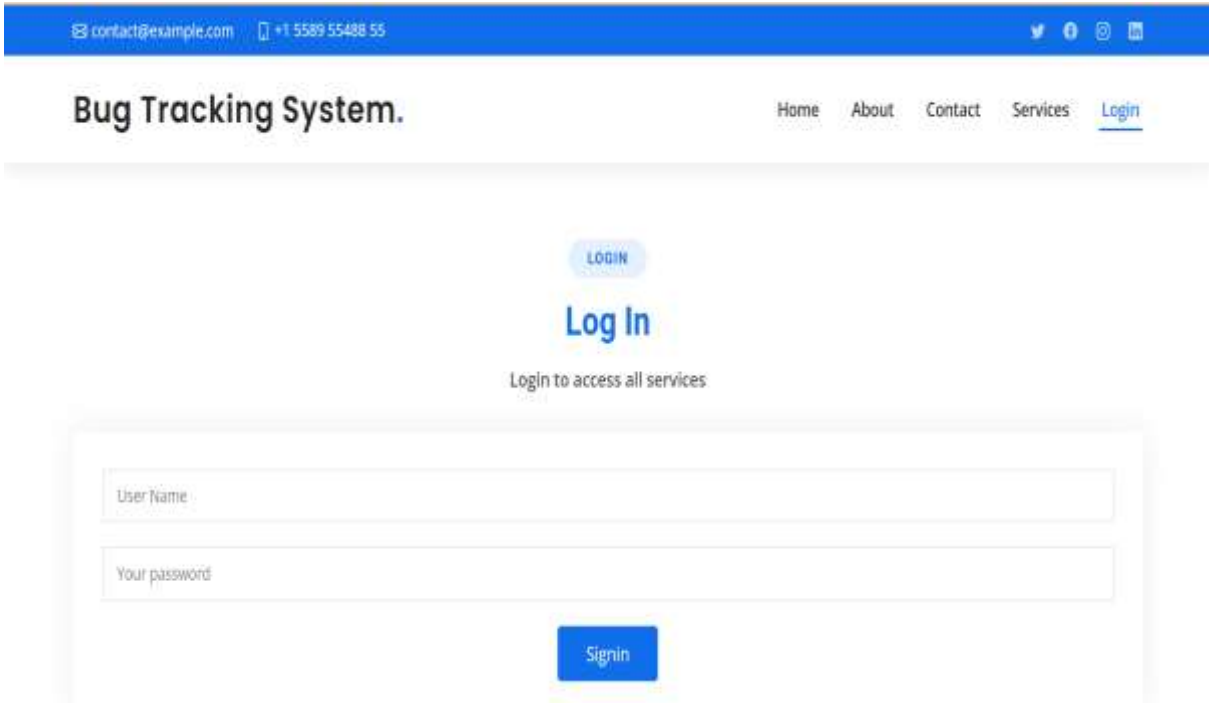
3. **Contact Page :**



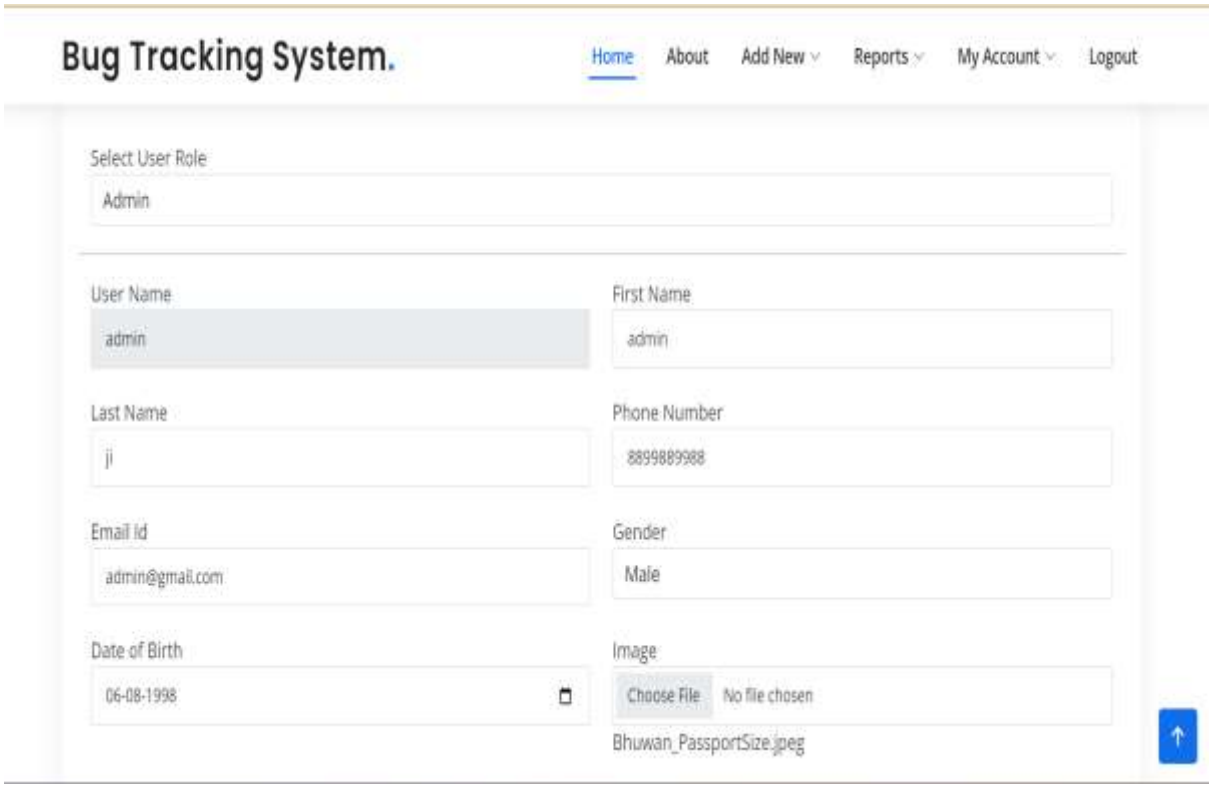
4. **Services page :**



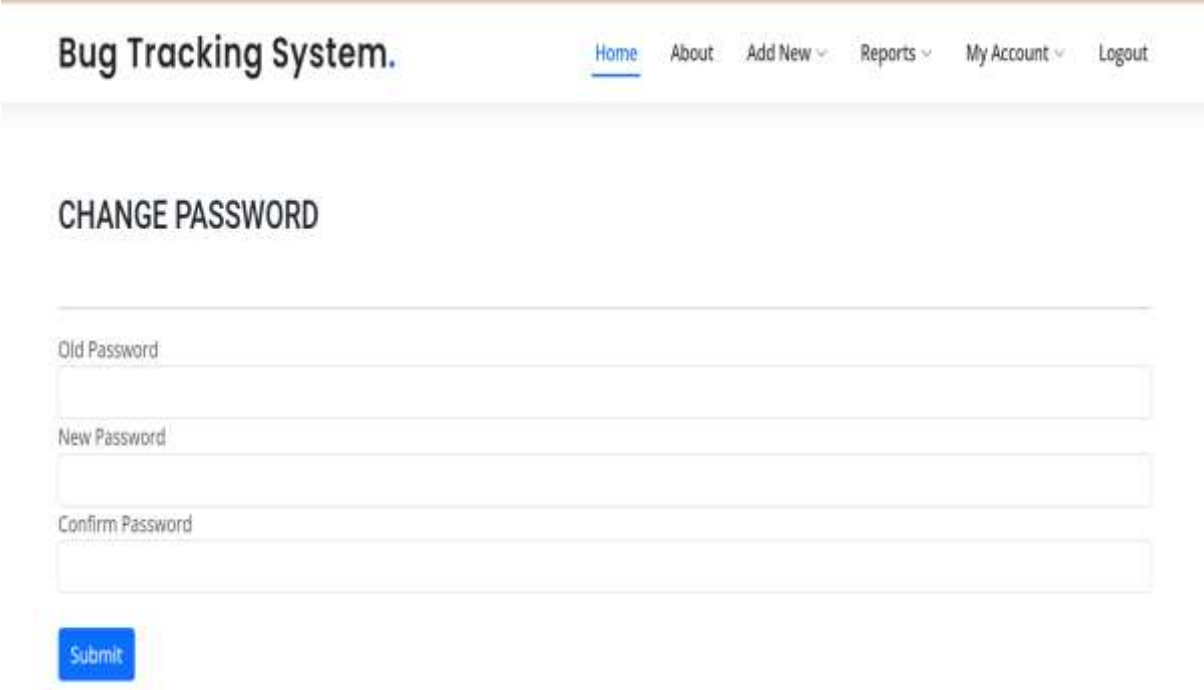
5. **Login Page :**



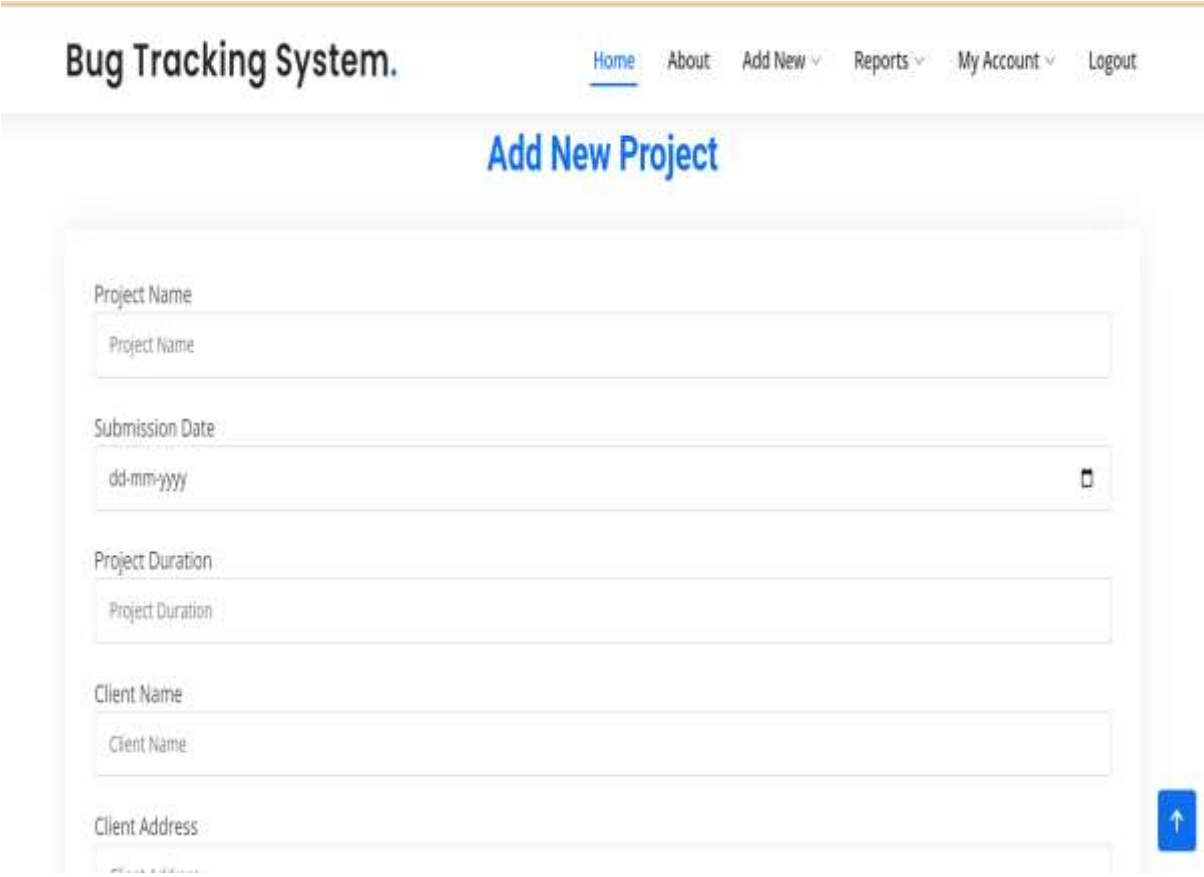
6. **Admin login profile**



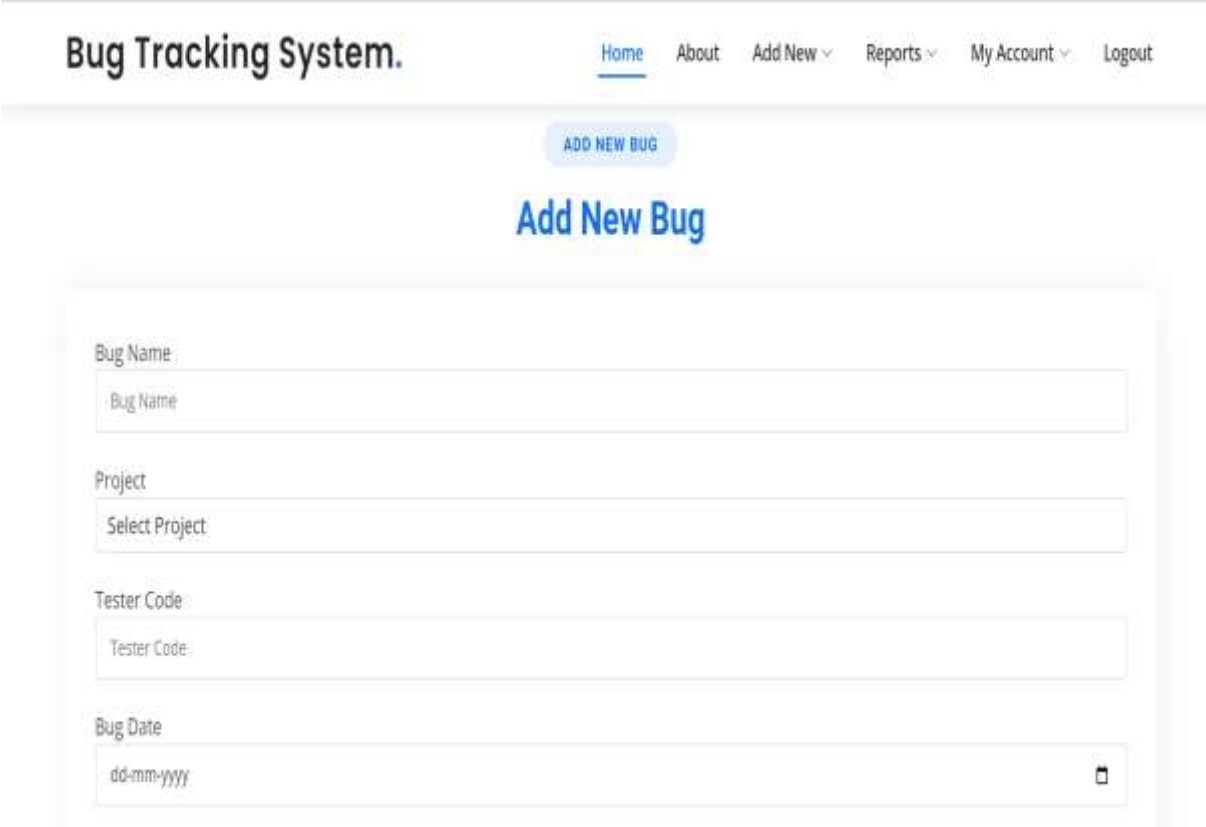
7. **Change password page**



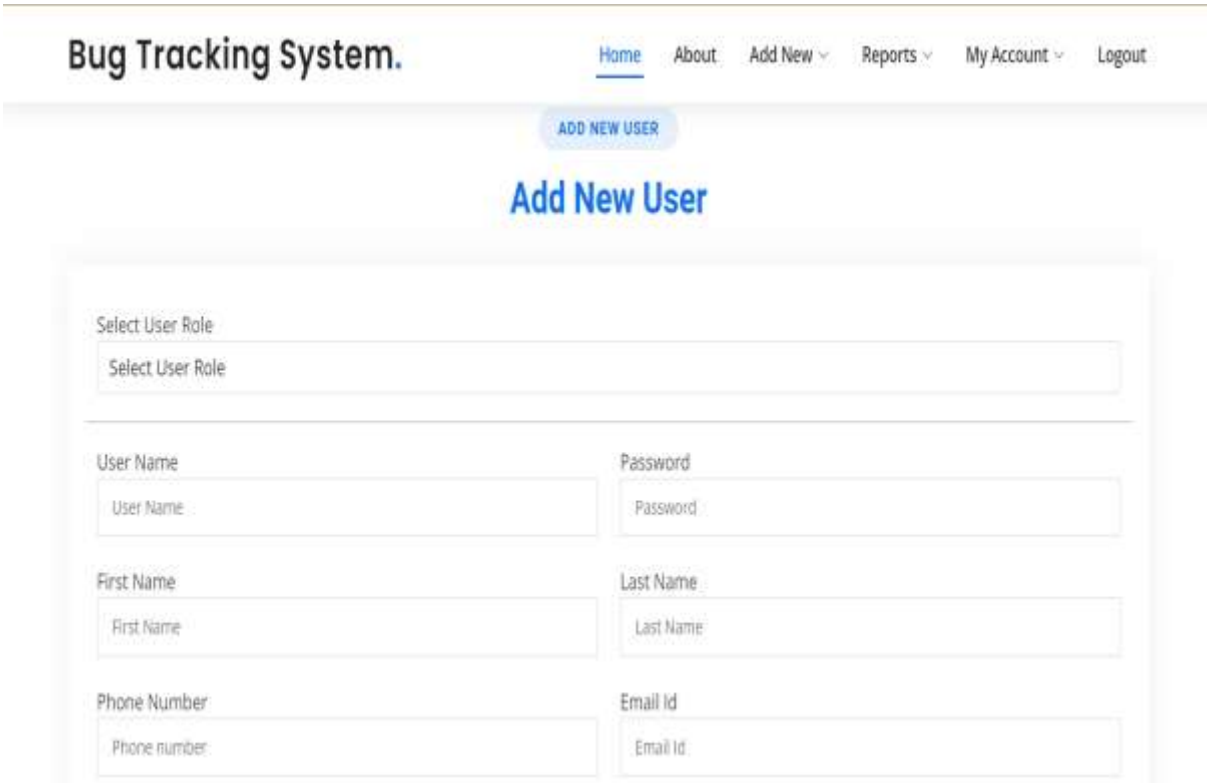
8. **Adding new project**



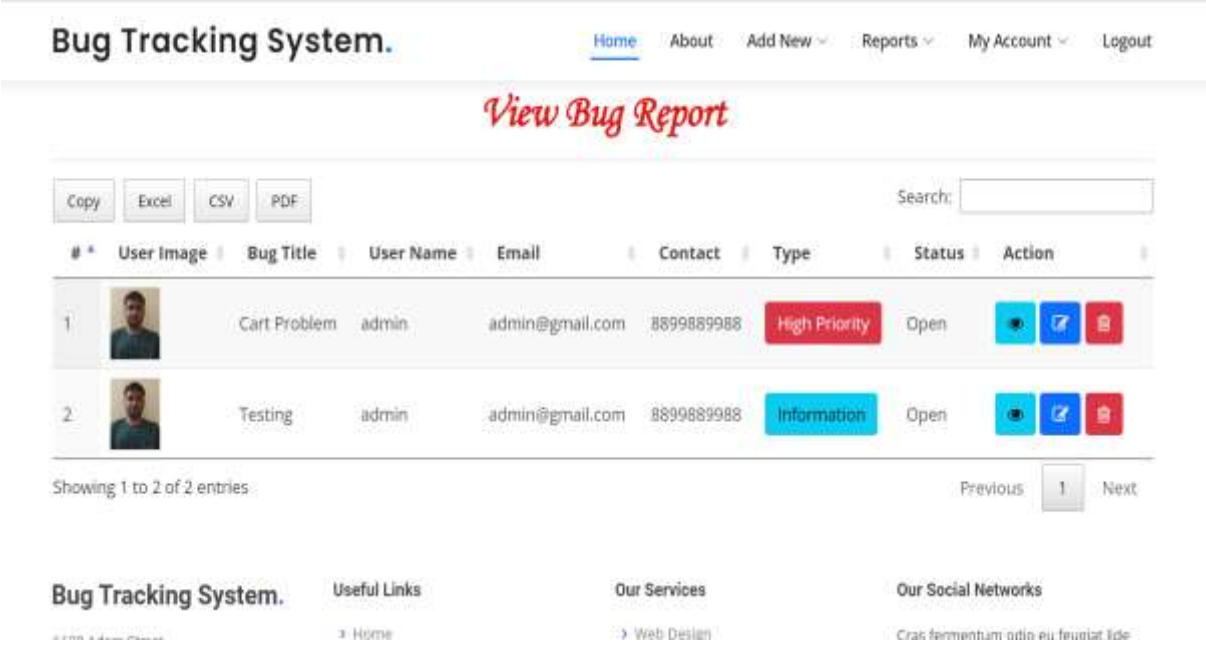
9. Adding new bug



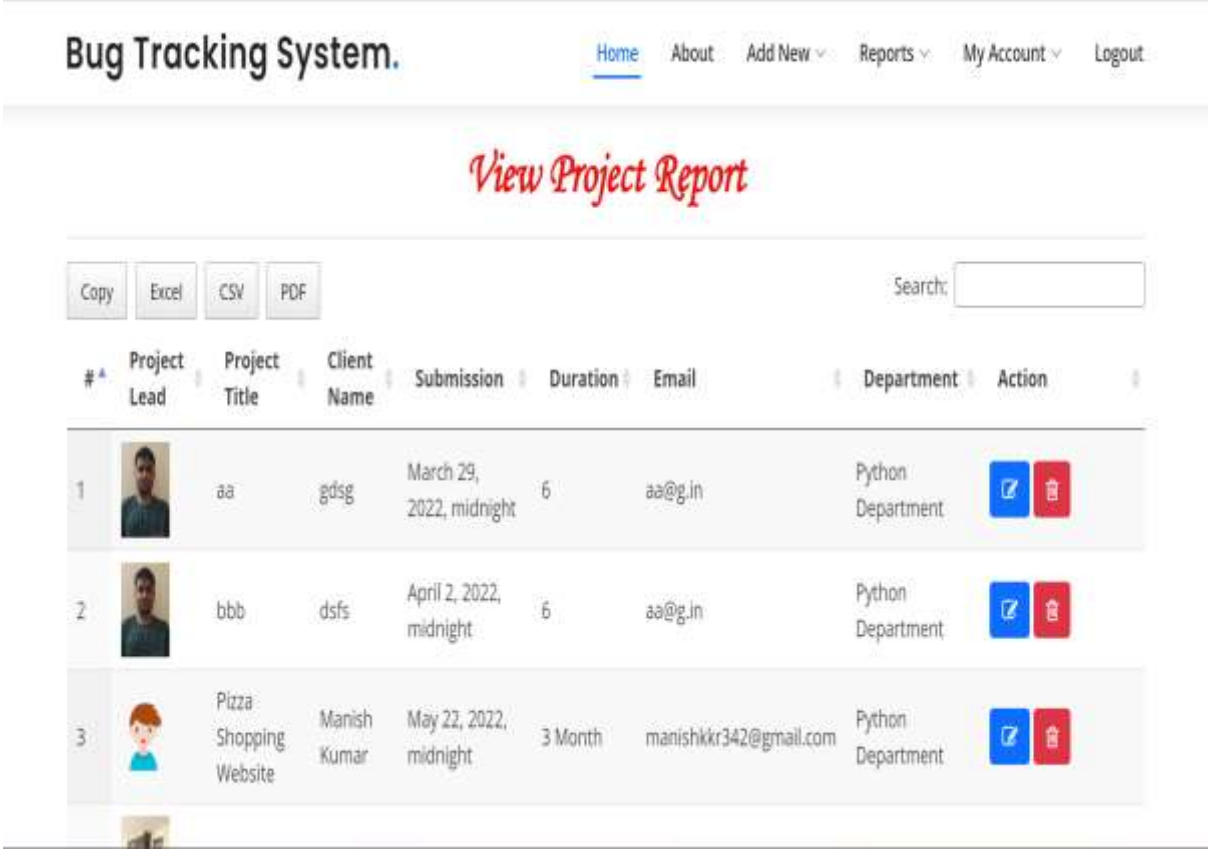
10. Adding new user



11. View new report



12. View project report



8. TESTING & RESULTS

8.1. TESTING AND TEST CASES:

UNIT TESTING

Unit testing, is a testing technique using which individual modules are tested to determine if there are issues by the developer himself.. it is concerned with functional correctness of the standalone modules. The main aim is to isolate each unit of the system to identify, analyze and fix the defects.

Unit Testing Techniques:

Black Box Testing - Using which the user interface, input and output are tested.

White Box Testing –Used to test each one of those functions behavior is tested.

DATA FLOW TESTING

Data flow testing is a family of testing strategies based on selecting paths through the program's control flow in order to explore sequence of events related to the status of Variables or data object. Dataflow Testing focuses on the points at which variables receive and the points at which these values are used.

INTEGRATION TESTING

Integration Testing done upon completion of unit testing, the units or modules are to be integrated which gives raise too integration testing. The purpose of integration testing is to verify the functional, performance, and reliability between the modules that are integrated.

BIG BANG INTEGRATION TESTING

Big Bang Integration Testing is an integration testing Strategy wherein all units are linked at once, resulting in a complete system. When this type of testing strategy is adopted, it is difficult to isolate any errors found, because attention is not paid to verifying the interfaces across individual units.

USER INTERFACE TESTING

User interface testing, a testing technique used to identify the presence of defects is a product/software under test by Graphical User interface [GUI].

Description	Preconditions	Test Steps	Expected Outcome	Pass/Fail
User creates a new bug report.	User is logged in.	1. Navigate to "Create Bug" page. 2. Fill in bug details. 3. Submit the bug.	Bug is created successfully.	Pass
Admin assigns a bug to a developer.	Admin is logged in. A bug exists in the system.	1. Navigate to "Assign Bug" page. 2. Select a bug and assign it to a developer.	Bug is assigned to the developer.	Pass
Developer resolves a bug.	Developer is logged in. An assigned bug exists.	1. Navigate to "Resolve Bug" page. 2. Select an assigned bug and mark it as resolved.	Bug status is updated to "Resolved."	Pass
User views all bugs in the system.	User is logged in. Bugs exist in the system.	1. Navigate to "View Bugs" page. 2. Review the list of bugs.	All bugs in the system are displayed.	Pass
Admin views bugs assigned to a specific developer.	Admin is logged in. Bugs are assigned to the developer.	1. Navigate to "Assigned Bugs" page. 2. Select a developer and view assigned bugs.	List of bugs assigned to the developer is displayed.	Pass
User adds a comment to an existing bug.	User is logged in. An existing bug with comments exists.	1. Navigate to the bug details page. 2. Add a comment and save.	Comment is added to the bug details.	Pass

9. FUTURE ENHANCEMENT

The development of this project surely helps to solve and address all the problems faced by software testers and developers. It can be implemented in almost any software development firms even Freelance developers can make use of this system on being upgraded in the future. The bugs can be fixed with ease using this system.In a nutshell, it can be summarized that the future scope of the project circles around maintaining information regarding:

- Generate reports on the overall performance of the system.
- Making site responsive to work with mobile.
- Developing mobile app.
- Email/SMS notification to User/Admin.
- Discussion Forum.
- We can give more advance website for Bug Tracking Systemincluding more facilities.
- We will host the platform on online servers to make it accessible worldwide.
- Integrate multiple load balancers to distribute the loads of the system.
- Create the master and slave database structure to reduce theoverload of the database queries.
- Implement the backup mechanism for taking backup of codebase anddatabase on regular basis on different servers.

The above mentioned points are the enhancements which can be done toincrease the applicability and usage of this project. We have left all theoptions open so that if there is any other future requirement in the system bythe user for the enhancement of the system then it is possible to implement them.

10. CONCLUSION

10.1 User Manual

A bug tracking system aids in the effective detection and management of defects in software products. This project BTS can be used to track defects in project modules and help with error debugging during testing and development processes. This project eliminates all potential sources of delay in problem reporting levels inside project modules in the software industry.It is substantially more when the application is installed on a company server.

Our project is only a humble venture to satisfy the needs of Software Companies forusing Bug Tracking. Several user friendly coding have also adopted. By using this, acompany can manage resources in a better way and offer solutions much faster. BugTracking can be used in each and every stage of the development process, thushelping developers to be content and more productive. This needs to be donerigorously and if you are not using it, then probably your development efforts can goin vain.At the end it is concluded that we have made effort on following points.

- A description of the background and context of the project and itsrelation to work already done in the area.
- Made statement of the aims and objectives of the project.
- The description of Purpose, Scope, and applicability.
- We describe the problem on which we are working in the project.
- We describe the requirement Specifications of the system and theactions that can be done on these things.
- We understand the problem domain and produce a model of thesystem, which describes operations that can be performed on thesystem.
- We included features and operations in detail, including screen layouts.
- We designed user interface and security issues related to system.
- Finally the system is implemented and tested according to test cases

10.2. BIBLIOGRAPHY

REFERENCES

[1] Torky Sultan, AymanE.Khedr, MostafaSayed, "A Proposed Imperfection Following Demonstrate for Classifying the Embedded Imperfection Reports to Improve Program Quality Control", ACTA Advise MED. 2013 Jun; 21(2): 103-108

[2] Yajie Wang, Ming Jiang, Yueming Wei. "A Software Quality System for Mobile Application Testing" 2012, the Fourth Universal Conference on Propels in Framework Testing and Approval Lifecycle.

[3] Priyanka Chandani, Chetna Gupta. "A Study on Successful Imperfection Avoidance - 3T Approach" 2014, 1, 32-41 Distributed Online February 2014 in MECS

[4] Frank Salger, Stefan Sauer, Gregor Engels"An Coordinates Quality Confirmation System for Indicating Trade Data Systems"2009, Procedures of CAiSE Gathering

[5] Stephen S. Yau, Yeou-Wei Wang, Jules G. Huang, and Jinshuan E. Lee, "An Coordinates Master Framework System for Computer program Quality Assurance*",0730- 3157/90/0000/0161$01.OO 1990 IEEE

[6] V.B. Singh, Krishna Kuma, Chaturvedi "Bug Following and Unwavering quality Evaluation System (BTRAS)"International Diary of Computer program Building and Its Applications Vol. 5 No. 4, October 2011

[7] Dane Bertram, Amy Voida, Saul Greenberg, and Robert Walker. "Communication, Collaboration, and Bugs: The Social"Nature of Issue Following in Little, Collocated Groups" CSCW 2010, February 6– 10, 2010, Savannah, Georgia, USA. Copyright 2010 ACM 978-1- 60558-795-0/10/02.

[8] SujataSolanke* and Prof. Prakash N. Kalavadekar**, "Deformity Following Framework", Worldwide Diary of Electrical, Hardware ISSN No. (Online): 2277-2626 and Computer Building 3(1): 212- 217(2014).

[9] Gabriela Avram, Anne Sheehan and Daniel K. Sullivan"Defect Following Frameworks in Worldwide Computer program Improvement – a work hone ponder" Interaction Plan Middle, Division of Computer Science & Data Frameworks, College of Limerick, Ireland.

[10] Gauri M. Puranik. "Plan of Bug Following Framework" Universal Diary of Imaginative Investigate in Science, Building, and Innovation (An ISO 3297: 2007 Certified Organization) Vol. 3, Issue 7, July 2014.