# **BUG TRACK**

Software Bug Reporting and Tracking System

# **Team members**

# Najeem VT

# Arunima Jayaraj

# Maria Benny

# Uma Devi

# Muhammad Ajnas

**HIGH LEVEL DESIGN**

**Version History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <01/07/22> | <1.0> | HLD 1.0 | Uma Devi |

**Contents**

1. **Introduction** 
   1. Why this High-Level Design Document?
   2. Scope
   3. Overview
2. **General Description** 
   1. Product Perspective
   2. Tools used
   3. General Constraints
   4. Assumptions
3. **Design Details** 
   1. Main Design Features
      1. Sequence Diagram
      2. Use Case Diagram-Customer
      3. Use Case Diagram- Administrator
      4. Use Case Diagram- Engineer
   2. Technology Architecture
      1. Application Architecture
      2. Presentation Layer
      3. Data Access Layer
      4. Tools Used
   3. Standards
   4. Database design
   5. Files
   6. User Interface
   7. Reports
   8. Error Handling
   9. Interfaces
   10. Help
   11. Performance
   12. Security
   13. Maintainability
   14. Portability
   15. Reusability
   16. Application compatibility
   17. Major Classes

## 1. Introduction

**1.1. Why this High-Level Design Document?**

The purpose of this High-Level Design (HLD) Document is to add the necessary detail to the current project description to represent a suitable model for coding. This document is also intended to help detect contradictions prior to coding, and can be used as a reference manual for how the modules interact at a high level.

**1.2. Scope**

The HLD documentation presents the structure of the system, such as the database architecture, application architecture (layers), application flow (Navigation), and technology architecture. The HLD uses non-technical to mildly-technical terms which should be understandable to the administrators of the system.

* C++ – A programming language to interface between Application and File Management system.

**1.3. Overview**

The HLD will:

* Present all of the design aspects and define them in detail
* Describe the user interface being implemented
* Describe the hardware and software interfaces
* Describe the performance requirements
* Include design features and the architecture of the project

• List and describe the non-functional attributes like:

o Security o Reliability o Maintainability o Portability o Reusability o Application compatibility o Resource utilization o Serviceability

## 2. General Description

**2.1. Product Perspective**

The Bug Reporting System is to facilitate the software user (Customer) to report the bug present in the software and to get it resolved at the earliest. This system is designed in such a way that the whole process is made hassle-free. This system is comprised of several different components. These components will be programmed. The language implemented will be dictated by its purpose. The administrative and user interfaces will be using relational database to retrieve, insert, delete and update the data. C++ will be used to submit commands for the automated part of the project such as updating the Bug history and data. This setup will allow for multiple users to login and interact with the program. It will also be set up using three user levels.

First type of user is the Customer. Customer is the user who will be reporting the bug in the corresponding software. The second type of user is the one who acts as an intermediary between the Customer and the Engineer. i.e., Administrator. The Administrator handles the bug reports and its current status, assigns Engineers to resolve the bugs, creates and handles the user accounts. This page is automatically displayed based on their username and password. The third type of user is the Engineer. They have the ability to change information in the database such as Engineer and Admin details and vehicle and record history can be viewed. This user level can only be attained by logging into the system.

This system will reduce the manual work to a great extend and the whole process can be managed just through an automated software. It reduces the paperwork and gives out all the details regarding the bugs in a single click.

**2.2. Tools used**

1. Draw.io, a Java based UML design program, is used to generate all of the diagrams used in analysis and design phases of the project.
2. The project will have a relational database backend using File system.
3. Automated interfacing with the database behind the scenes will be using C++.
4. Microsoft Visual Studio-2019 is the development platform.
   1. **General Constraints**

The Bug Reporting System must be user friendly and as automated as possible. Administrators should not be required to do anything besides the initial setup and monitoring, and users should not be required to know any of the workings. Without logging in, the user will not have any access. After logging in, that user has the ability to change settings and user histories.

* 1. **Assumptions**

This project is based on the idea of a Bug Reporting Software, and the goal is to make this idea a reality using Software Engineering practices. In doing so, many documents are created, and it is assumed that design flaws will be found early on. It is also assumed that all aspects of this project have the ability to work together in the way the designer is expecting.

Another assumption is that the current intended documentation will suffice to make this project count towards the Software Engineering Subtract. There is also an assumption that none of the work or hardware will be stolen or sabotaged. The final assumption is that a demonstration and presentation will be possible at the end of the training.

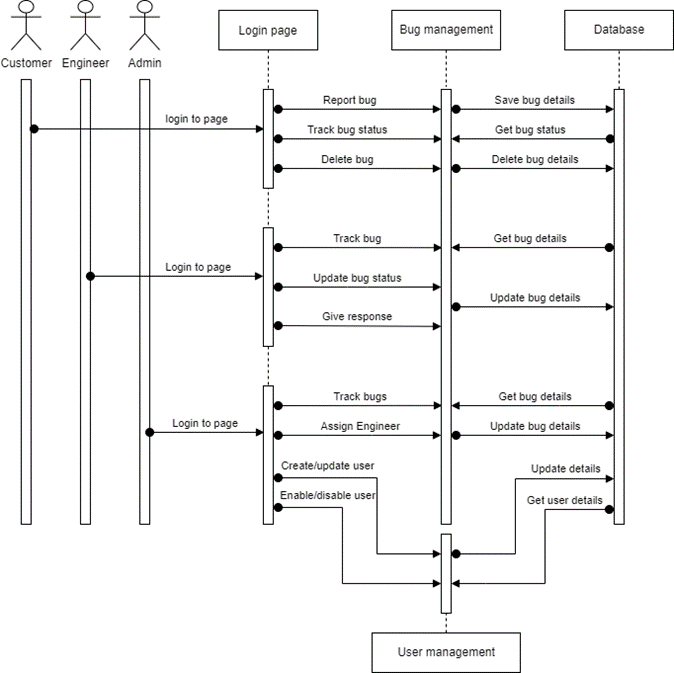
## 3. Design Details

**3.1. Main Design Features**

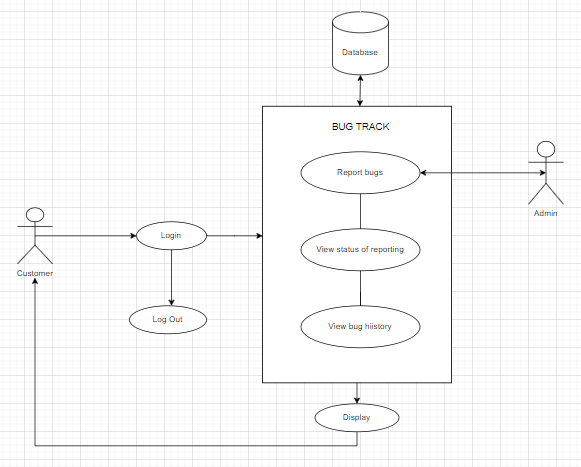
The main design features include five major parts: the architecture, the user interface design, external interface, the database, process relation, and automation. In order to make these designs easier to understand, the design has been illustrated in attached diagrams

(Sequence diagram, Use case and Database design).

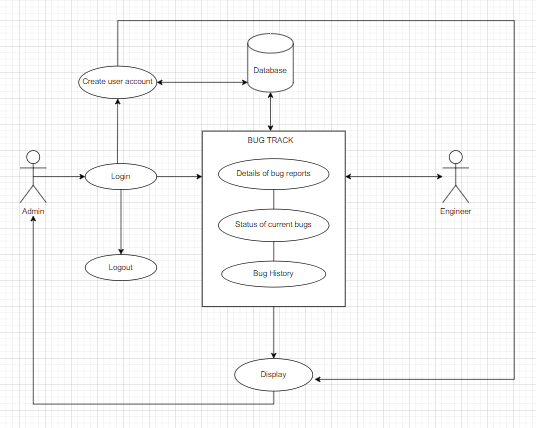
**3.1.1 Sequence Diagram**



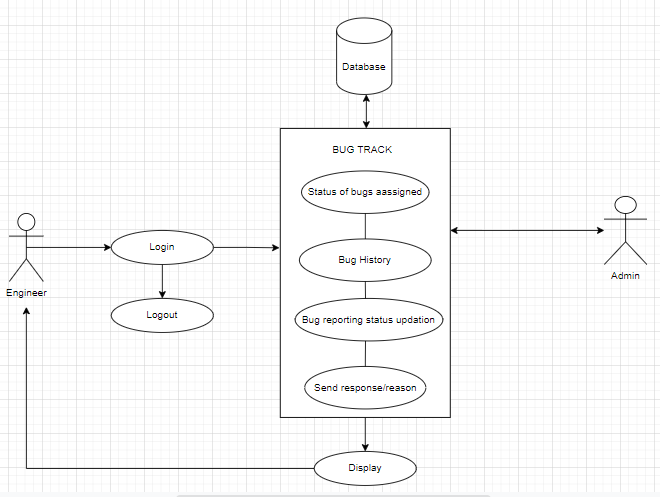
**3.1.2 Use Case Diagram- Customer**



**3.1.3 Use Case Diagram- Administrator**



**3.1.3 Use Case Diagram- Engineer**



The Bug reporting system can be accessed by three different users. One is the Customer who will be reporting the bug. The second is the Engineer who will be assigned to resolve the bug. And the third user is the Administrator who will be acting as an intermediary between the Customer and the Engineer. The Customer, Engineer and the Administrator can log in to the system using their valid credentials. The Customer has the options to report bugs, view the history of the bugs reported so far and also view the current status of the bugs. The Administrator is the one who will be acting as an intermediary between the Customers and Engineers. The Admin will be assigning the reported bug to an Engineer depending on their availability. The Admin can view the details of the bug reports, status of the current bugs and also the history of the bugs reported so far. Another privilege of the Administrator is that only the Admin can create, access and modify the User accounts in the system. The Engineer has view the status of bugs assigned and the bug history. The Engineer can update the status of the bugs assigned. Also the Engineer can send response or feedback to the Administrator if necessary. The database is created using File system. Mainly two types of files are required in our system. One is used to store all the details regarding the bugs and another one is for the user accounts.

**3.2. Technology Architecture**

**3.2.1. Application Architecture**

The Functionality will vary based user privileges of the user logged in. Administrators will have access to administrative privileges based on the authorization provided to them.

**3.2.2. Presentation Layer**

Information will include the email and password, all information pertaining to that email and password. And also all the details regarding the bugs. Administrative screens will have access to information pertaining to all users, and the ability to change histories and system settings.

**3.2.3. Data Access Layer**

The database will be accessible to all users, administrators, and automated services. A login will determine what parts of the database can be accessed and changed.

**3.2.4. Tools Used**

See section 2.2 for tools used in the design of this project.

**3.3. Standards**

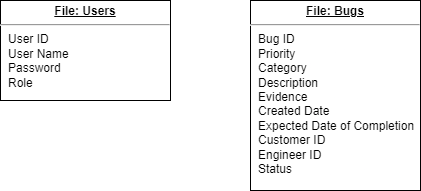
Database – relational

Inputs – entered through text field and stored in file system.

Security – username and password are required for access to the system.

Quality – by keeping the interface simple and direct, quality should be kept at a maximum.

**3.4. Database design**



**3.5. Files**

This system will not use a large number of files. There are two files that will be used in the system. One is for storing all the details regarding the bugs and another one is for storing the credentials of the user. The user file will be used to store all the username, passwords and all other attributes specified for those users. This file will be accessed at the time of login. It can be modified by the administrator at any time. Another file will store all the data related to the bugs. These details can be viewed all three users.

**3.6. User Interface**

The user interface is a very simple plain layout with little graphics. It will display information very clearly for the user and will primarily output information to the user through application window. Screen shots have been provided to demonstrate the user and administrative interface.

**3.7. Reports**

The reports will display the bug details up to the last time the system updated it and their history. Reports can also be viewed by an administrator for any vehicle.

**3.8. Error Handling**

Should errors be encountered, an explanation will be displayed as to what went wrong. An error will be defined as anything that falls outside the normal and intended usage.

**3.9. Interfaces**

There are four main interfaces for this project. First, the login interface. Second is the interface according to the user i.e., Customer, Admin or Engineer is sent from the login page to the database. The third interface is the User detail interface of administrative interface. The final interface will be between the application and the database.

**3.10. Help**

Help will come in the form of all the documentation created prior to coding, which explain the intended uses. Should time allow, detailed instructions will be written on how to create and implement the system with the intentions of publishing as an Open-Source solution.

**3.11. Performance**

Performance is going to be very important for this project. For everything to run smoothly for this project, the gateways will have to be able to update data on the File management system and refresh the data tables before it is supposed to do so again. This is likely to be the most processor intensive aspect of the project. The pages will also need to supply requested pages to the software users at a reasonable speed. The file management system will need to keep up with all the data requests and transactions.

**3.12. Security**

Because security is not the prime focus of this project, only the minimal aspects of security will be implemented. A username and password will be required to log into an administrative, engineer or customer interface and to access the database.

**3.13. Maintainability**

Very little maintenance should be required for this setup. An initial configuration will be the only system required interaction after system is put together. The only other user maintenance would be any changes to settings after setup, and any specified special cases where user settings or history need to be changed. Physical maintenance on the system’s parts may be required, and would result in temporary loss of data or Internet. Upgrades of hardware and software should have little effect on this project, but may result in downtime.

**3.14. Portability**

This system should have the ability that, once it is together, the entire system should be able to be physically moved to any location. Code and program portability should be possible. For everything to work properly, all components should be compiled from source.

**3.15. Reusability**

The code written and the components used should have the ability to be reused with no problems. Should time allow, and detailed instructions are written on how to create this project, everything will be completely reusable to anyone.

**3.16. Application compatibility**

The different components for this project will be using C++ as an interface between them. Each component will have its own task to perform, and it is the job of the C++ code to ensure proper transfer of information.

**3.17. Major Classes**

The relationships between major classes are:

* An administrator can create user accounts.
* An administrator can view or modify a user’s details.
* An administrator can view Bug reported by customers.
* An administrator can view the status of the bug reports.
* An administrator can assign particular bugs to Engineers based on their availability.