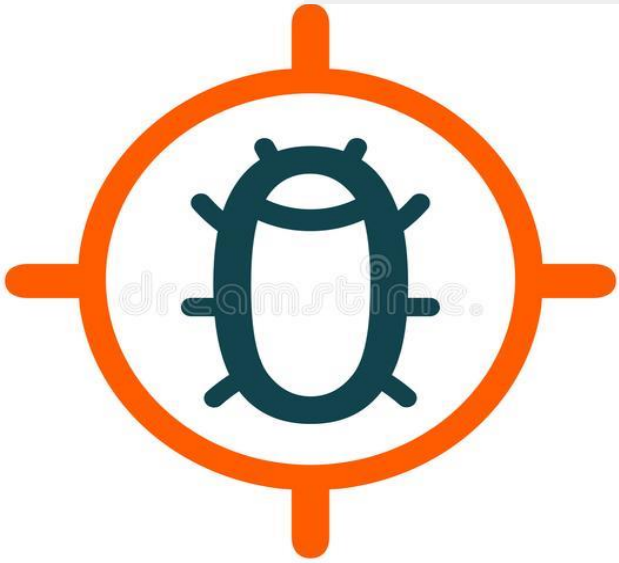
Faculty Of Engineering
Helwan university

Software Requirement System Report for Bug Tracker System



Version 0.1

Prepared by:

- Ahmed Khaled Mohana Tohamy.
- Mahmoud Ehab Mahmoud Abdeen.
- Mariam Mohamed Abdelmonem Ismail.

Under Supervision of:

DR: Manal Showman

Table of Contents:

Team members:

Name	Code	Work in SRS
Ahmed Khaled Mohana Tohamy.	20026	○ Introduction. ○ System architecture. ○ System non-functional requirements.
Mahmoud Ehab Mahmoud Abdeen.	20010	○ Preface. ○ User requirements definitions. ○ Interface requirements.
Mariam Mohamed Abdelmonem Ismail	20047	○ System users. ○ System functional requirements ○ System evolution ○ Future work.

List of Figures:

1. Preface:

1.1 Document Purpose:

- Through this document, a complete and detailed specification of the Bug Tracker System will be provided, making it possible to understand the ins and outs of the software and what each user will be able to achieve by using it.
- In the first section, an overview of the system will be elaborated upon, and then each part of the system will be discussed in more detail in the following sections.

1.2 Target Users:

- This document is authored by system engineers based on the requirement obtained from the senior software engineers, developers and testers of the company that issued the development of the system.
- The document is to be approved and acknowledged by the CEO, senior SW engineers, developer, testers, and the crew working on developing the project.

1.3 Revision History:

Version	Author	Description	Date
0.1	All members of the team	Initial	23-7-2022

2. Introduction:

2.1 Purpose:

- purpose of the software is to track bugs and allow the end user to manage bugs and store them if they ever appear in other projects.

2.2 Scope:

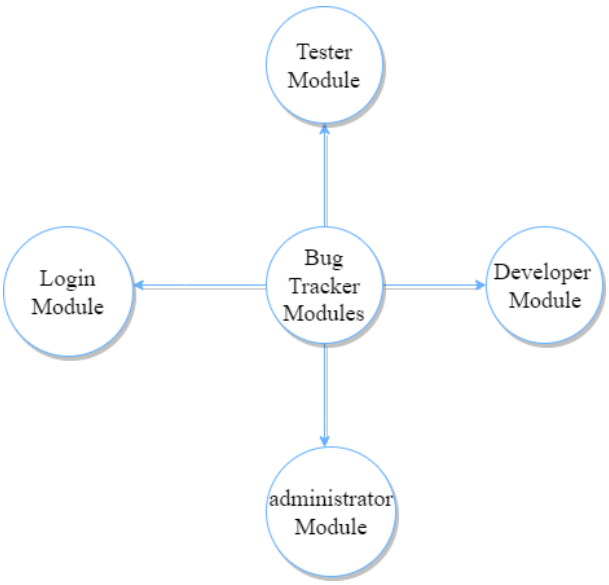


Figure 1: System Modules hierarchy

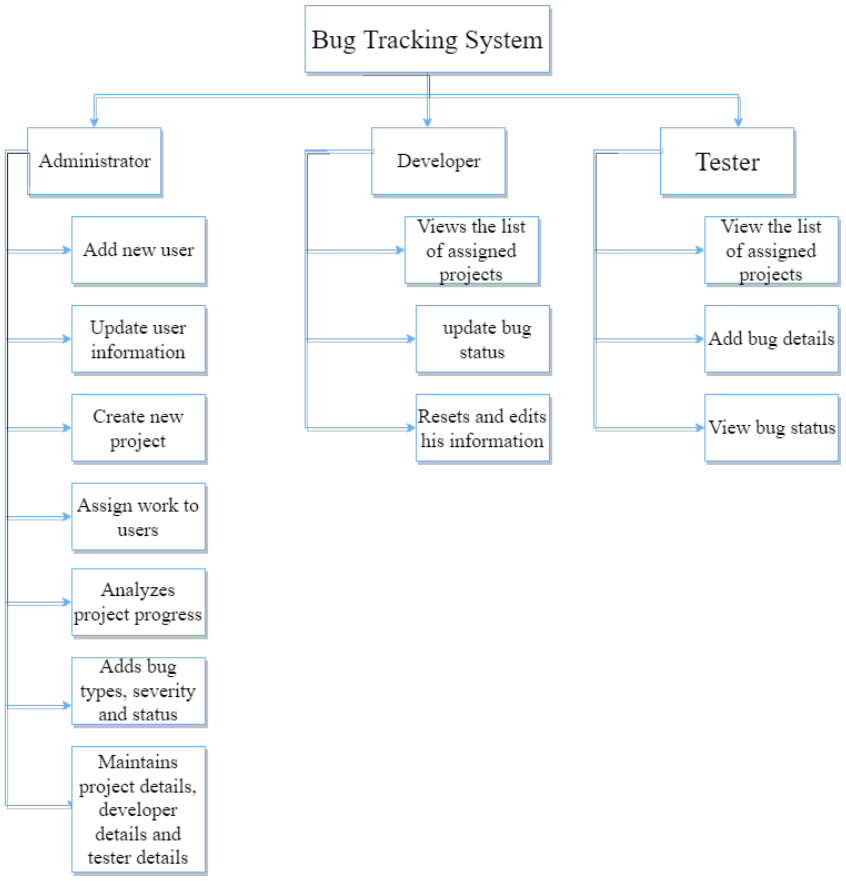


Figure 2: System Functions hierarchy

2.3 Overview:

- The document is organized as follows: an overview description of the bug tracker system and

its high-level functions are presented (section 2.1 and 2.2)

  - section 4 state types of users who can use bug tracker.
  - Section 5 in the document provides user requirement definitions.
  - Section 6 in the document provides system architecture.
  - Section 7 in the document provides a detailed description of the system functions and requirements.
  - Section 8 in the document provides interface requirement.
  - Section 9 in the document provides the system non functions and requirements.
  - Section 10 presents some helping information and diagrams that will facilitate the understanding of the contents.
  - Section 11 in the document provides system evolution.
  - Section 12 in the document provides future work.
  - Section 13 in the document provides time plan.
  - Section 14 in the document provides appendices.
  - Finally, Section 15 in the document provides references.

3. Glossary:

3.1 Acronyms, definitions, and abbreviations:

- **FOEHU**: Faculty of Engineering, Helwan University.
- **BTS:** Bug Tracker System.
- **User:** Developer or tester who uses the application.

4. System Users:

4.1 System stakeholders:

- System Engineer.
  - Responsible for gathering requirements.
  - Responsible for development.
  - Responsible for support.
- Administrator:
  - Adds new users.
  - Updates current user’s information.
  - Creates projects.
  - Assigns projects to users.
  - Analyses project progress assigned to particular users.
  - Adds bug types, severity, and status.
  - Maintains project details, developer details and tester details.
  - View user’s data.

- Developer:
    - Views the list of assigned projects.
    - Responsible for updating bug status.
    - Resets and edits his information like passwords.
- Tester:
    - View the list of assigned projects.
    - Adds bug details.
    - View bug status.

4.2 Users objectives:

- System Engineer.
    - Gains experience in software engineering and developments.
- Administrator:
    - Follows up on the work of the users.
- Developer:
    - Simplification in finding the assigned work.
    - Prevent assigned work mixing.
    - Work in an organised environment.
- Tester:
    - Simplification in finding the assigned work.
    - Prevent assigned work mixing.
    - Work in an organised environment.

5. User Requirements definitions:

5.1 System Functions:

1. Add new user.
2. Update existing user information.
3. Create projects.
4. Assign project to user.
5. Analyze project progress.

6. Add bug details (types, severity, and status).
7. Maintain details (projects, developers, and testers).
8. View assigned projects.
9. Update bug status.
10. Reset password.
11. Report bug.
12. View bug status.

5.2 Constraints:

1. Company Policies:
    - Users must be limited by permissions to insure data confidentiality.
2. Cultural Constraints:
    - Code must follow company standards.
3. Technological Limitations:
    - Integration with Git is required to allow database synchronization.

6. System Architecture:

- We use database to connect between all users of the system and store information.

Figure 3: System Architecture

7. System Functional requirements:

7.1 Add new user:

- The administrator can add a new user to the system (Administrator – Developer – Tester) by adding the following information:
  - Main information (name, birth date, birth location, gender, graduation college and university, status).
  - Contact information (mobile number, phone number, address).
- The administrator is also supported to scan the following documents for each user:
  - Personal image.
  - Birth certificate.
  - Personal ID.

7.2 Update user information:

- Only the administrator can change the user information.

7.3 Create new project:

- Only the administrator can change the user information.

7.4 Assign work to users:

- Only the administrator can assign to the other users in the project (Developers and Testers) a specific task in different parts of the project.

7.5 Analysis project progress:

- Only the administrator can follow up the progress on each project and how the other members in the project works.

7.6 Add bug types, severity and status:

- Only the administrator can add bug types, severity and status.

7.7 Maintain project details, developer details and tester details:

- Only the administrator can maintain the activity of the project, each developer in the project and each tester in the project.

7.8 View the list of assigned projects:

- Only the developers and testers can view the list of the assigned projects and works which have assigned by the administrator.

7.9 Update bug status:

- Only the developers can update the bug status from unsolved to solved after adding the solution of this bug.
- Developers don't have the access to the whole bugs in the project, but they have the access to the assigned bugs only.

7.10 Reset and edit information on the profiles:

- Developers have the access to update or reset their information, but the admin will receive a notification when this action happens.

7.11 Add bug details:

- Only testers have the access to create a new bug a add the details of this bug.

7.12 View bug status:

- Testers can see the bug status, but they don't have the access to change it.

8. Interface requirements:

8.1 User interfaces:

- Here we will include the GUI of each module in our project.

8.2 Software Interfaces:

- Database will be created using apache derby and will be maintained and updated using Git servers.

9. Non-functional requirements:

9.1 Availability:

- The system should be available during working hours.

9.2 Availability:

- No one can access the system from outside the faculty.

9.3 Safety:

- Servers will be kept in the vice dean office.

9.4 Maintainability:

- The system consists of modules every module consists of classes in order, so the system is easy to maintainable.

9.5 Portability:

- The system is a desktop application can run on several operating systems.

9.6 Usability:

- The system is usable for all users (developers & testers).

9.7 Efficiency

- The system is efficient in a lot of software companies because it solve the bugs of the software.

10. System Models and Diagrams:

- Here we will include the system's context diagram, use case diagram, class diagram andsequence diagrams for any needed functional requirement.

11. System Evolution:

- The system should be able to work on different operating systems.
- It should work properly on devices with low specifications. Only the server on which it's installed will be powerful.

12. Future Work:

- Administrator:
    - Print users data in a table.
    - Print entire project's bug information.
    - Print information about each bug in the project individually.

- Developer:
    - Add errors and request assistance.
    - Print information about each bug in the project individually.

13. Time Plan:

13.1 Work Breakdown Structure:

13.2 Gantt chart:

14. Appendices:

15.References:

- https://projectsgeek.com/2016/02/bug-tracking-system-java-project.html