Mantis - Multi-tenant Issue Tracking & Reporting System

This document outlines the abstract requirement for the proposed issue tracking and reporting system.

1. Introduction

Issue tracking is a key component of the SDLC, spanning from the mid-project to post-delivery phases. Proper issue tracking has several benefits,

a. Making sure that issues are not missed by the developers
b. Assisting in sprint planning to include issue fixes
c. Properly segregating issues from change requests
d. Preventing scope creep
e. Tracking developer time consumption on the issues which in turn provide performance metrics (bug/code ratio)
f. Customer impression management

At present, there are several popular SaaS-based issues tracking systems in the market. Some of them are,

a. JIRA
b. Bugzilla
c. Github issues
d. Asana
e. Trello

Even with several competitive products in the market, none of them provide a solution set that's entirely catered towards a developer-first approach ie. the tracking system should be an assistant to the developer versus being an additional work. Some of the features that are desired include,

a. Should be able to onboard several customers easily for multiple projects
b. Allow customers to quickly log bugs without added complexity
c. Allow developers to manage the bugs using kanban boards
d. Allow developers to estimate time/effort (time tracking)
e. Provide bug pool for developers as a knowledge repository
f. Easy reporting tools

1.1 Scope

The scope of this document is to provide an introduction to the system requirement. The developers should create a specification based on the description provided.

1.2 Terminology

“Bug” and “Issue” are used interchangeably.

2. Functional Requirements

2.1 Authentication & Authorization

The system should have multi-tenant capabilities, it should

- Allow defining a super admin (this can be created once at the system setup)
- Allow admin to create users
- Allow admin to create user roles,
    - Administrator
    - Developer
    - QA
    - Manager
    - Customer
- Allow admin to assign roles to users
- Allow admin or system to define permissions for user roles,
    - Example 01: Customer can only file and view bugs and can't see the developer views
    - Example 02: QA can only view filed bugs and change status
- Allow admin to create projects
- Allow admin to assign users to projects. When assigned the user's permissions are limited to the scope of the project (ex: Developer can only see the issues and sprints of the assigned project)
- Users can login to system
- Users can request a password request

2.2 Bug Capture Log

Bug capture log is the interface where Customers engage with. Required features are,

- Customer can login to the system with provided credentials
- Customer can view the projects where he/she is a part of in the dashboard
- Customer can select a project and file an issue
- Issue filing process:
    - Customer selects "Create new Issue"
    - Issue window allows input of a description
    - Issue window allows uploading files
    - Issue window allows recording screen
      This is the most vital feature in issue creation. Customer should be able to turn on-screen recording and record the bug. Existing libraries can be used to achieve this feature. Once the recording is done, it should be attached as a video to the issue.
- Customer can file the issue and it's added to the capture log
- Customer can comment on created issues if necessary
- [Refer to JIRA/BugZilla issue views to get an idea]

2.3 Bug Management

Bug management is where bug life cycle processes. Required feature are,

- Only Developers, Managers, and QA (referred to as User in this context) can access bug management dashboard
- User can login to the system
- User can select a project and view bug log of the project
- User can
    - Add comments to the bug
    - Change severity and tag of the bug
    - Assign himself/herself to the bug
    - Assign someone else to the bug
    - Attach files to the bug
    - User can add time spent for a bug (time tracking)
- Apart from the above, specific user roles have additional capabilities
- Sprint is a selection of bugs that will be completed within the given Sprint period (usually 2 weeks)
- Developer/Manager can create a Sprint
- Developer/Manager can add bugs from bug log to Sprint
- Sprint additionally consists of a Kanban board
- By default, there will be 5 kanban lanes
    - Open
    - In-progress
    - QA
    - Done
- When a bug is added to the sprint, it will be in the Open lane
- Manager/Developer/QA can move the bug from one lane to the other

2.4 Bug Solution Pool

Bug solution pool is a knowledge base that stores solved bugs for future references. This allows a smooth knowledge share process in the internal organization. Required features are,

- Manager/Developer can mark a resolved bug to be added to the solution pool
- Solution pool should list all added bugs and should allow full-text search
- Manager/Developer can add comments, updates to bugs in the solutions pool

2.5 Bug Reports

Reporting is critical in issue tracking to monitor project progress and after evaluation. Reports must be able to be filtered by

- Date from
- Date to
- Developer/QA
- Project

Reports required initially,

Title	Description	Type
Developer timesheet	A table that has issues worked on categorized by the developer along with the number of hours taken Show totals by developer and entire selected time Should be able to download as PDF/CSV Follow a generic timesheet template	Table
Project timesheet	A holistic view, for managers so see progress on multiple projects A table that has the same information as above but grouped by the Project	Table
Sprint summary		
Monthly bug summary	A table with issues grouped by the month of reporting	Table
Months x Total Bugs	X axis - Month Y axis - Total number of bugs reported	Bar Chart
Months x Bug Development	X axis - Month Y axis - Stack with - Total bugs reported - Total resolved - Total in progress	Stacked Bar Chart
Project x Bug Development	X axis - Project Y axis - Stack with - Total bugs reported - Total resolved	Stacked Bar Chart

	- Total in progress	
Developer x Bug Development	X axis - Month Y axis - Stack with - Total bugs assigned - Total resolved - Total in progress	Stacked Bar Chart
Developer performance	Total assigned / Total Completed / Total In progress	Pie Chart
Developer x Average Effort	X axis - Month Y axis - Average hours spent on an issue	Bar Chart
Total bugs	Total bugs reported	Number
Total Active Projects	Total number of projects with active bugs	Number
Bug completion	Bugs completed/Bug reported*100	Number

3. Technologies

Suggested tech stack.

Component	Technology
Frontend	Reactjs
Backend	Python/Django
Database	Postgress SQL

4. Next Steps

To start this off, the next steps are as follows,

1. Background study
   Study on the existing issue tracking systems and the concepts of Bug, Bug tracking, Sprint, Kanban, Gantt
   And check the products JIRA, BugZilla, Asana

2. Create platform specification
   I'll give a project specification template or you could use the IEEE template. You have to convert this required to a specification

3. Create a design document
   From the specification, create the system design document - can use the IEEE template. It should include
   a. Wireframes for key views
   b. System architecture
   c. Use cases

4. Create a project timeline and work breakdown

5. Development
   Actual development we can start on after creating the specification. Because initially, you will have some learning curve