Team-based Programming Project

SWE1 - Spring 2021

**“OurTicket”** – Issue Tracking / Ticket Management System

**Brief Description:**

OurTicket is a support ticket management system / issue tracker which the group consisting of **Bailey Carothers, Christopher Miller, and Cole Roper** will design and implement.

**Scope:**

Insofar as scope is concerned, the goal is for our project to be a “bare-bones” demonstration of a support ticket management system that would be best suited for open-source community-driven endeavors. The project is designed around the concept of community voting, and is to be built such that a community as a whole decides what contributors should prioritize. The project will have a central hosted backend with a SQL/MariaDB database, have a functional web-based interface, and include many features one would expect of such a system (such as priority, tagging, classification of tickets).

**Motivation:**

Support ticket systems are implemented in a wide variety of manners for an even wider variety of use-cases. Programming initiatives, development firms, open-source community-driven projects, customer service companies, and many others make extensive use of issue trackers and support ticket management systems. Our goal is to build a proof of concept for “OurTicket,” a functional issue tracker/ticket management system which allows the community to prioritize issues for the contributors.

In “classical” support ticket systems, issues are assigned a priority manually by the support team and based on the classification of the ticket. This proves very useful for extremely centralized initiatives, as well as for customer service agencies. This falls short, however, for less centralized, community-driven initiatives with a weaker sense of leadership. Who decides what needs to get done? Rather than vest faith in some individual or committee and compromise the open-source community-driven nature of a project, our goal is to make OurTicket an issue tracker for the community projects, where the users of the project alongside its contributors make democratic, decentralized decisions about what needs to be done. This allows for non-contributors to feel more involved in the open-source tools they use, and can increase the effectiveness of contributors’ time since they can works towards community-decided initiatives.

**Five Primary Features:**

1. Online Database-centric backend.
   1. The issue tracker is not local, and will be hosted on a Raspberry Pi 4. This will allow us to access the tracker and its issues remotely through a web interface.
   2. Issues will be stored in a (likely MariaDB) database on the Pi 4.
   3. Users will, through a web interface, perform various actions on the database. These will be handled through webservice-esque microservices.
2. Open / Close / Read ticket
   1. The key component of any ticket management system is, of course, the tickets. Users will be able to remotely read, create, and close tickets.
3. Categorizable ticket class
   1. Tickets can be assigned different “classes”. Feature request, bug report, and other classifications will allow individual tickets to be categorized.
   2. Ticket category will be decided at ticket creation.
4. Ticket Priority
   1. Tickets will be able to be publicly voted on to decide their priority. The system will automatically categorize each ticket by priority based on the number of votes it receives relative to the other tickets in the system.
   2. Such a system is scalable, meaning a system with 3 tickets and 20 total votes should have no issues with priority classification, and neither should a project with 500 tickets and 50,000 votes.
5. User Privilege Levels
   1. Users designated as “verified contributor” are the only ones able to close tickets.
   2. Anyone can create and vote on a ticket.

**Implementation of (1):**

Create new database, OurTicket.

CREATE DATABASE OurTicket;

Create new user. For the sake of security, the password is ommitted.

CREATE USER 'ussr'@'localhost' IDENTIFIED BY '<pass>';

Grant privileges to user for the OurTicket database.

GRANT ALL PRIVILEGES ON OurTicket.\* TO 'ussr'@'localhost';

Log out of root and into new user.

Begin using the database.

USE OurTicket;

Create table for log in information with columns username, password hashes, and privilege level.

CREATE TABLE login (

username VARCHAR(16) CHARACTER SET utf8,

passhash VARCHAR(16) CHARACTER SET utf8,

priv TINYINT UNSIGNED

);

Insert user for testing purposes. Note, the smaller the value of privilege, the more privileges granted.

INSERT INTO login VALUES ("cole", "hash", 0);

Create table for the submitted tickets with columns priority, name, votes, category, and description.

CREATE TABLE ticket (

priority TINYINT UNSIGNED,

votes INT UNSIGNED,

name VARCHAR(16) CHARACTER SET utf8,

category VARCHAR(64) CHARACTER SET utf8,

description VARCHAR(512) CHARACTER SET utf8

);

To demonstrate the creation of these tables, we may run the following SQL "SHOW" statements:

SHOW COLUMNS FROM login;

+----------+-------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+----------+-------------+------+-----+---------+-------+

| username | varchar(16) | YES | | NULL | |

| passhash | varchar(16) | YES | | NULL | |

| priv | tinyint(4) | YES | | NULL | |

+----------+-------------+------+-----+---------+-------+

SHOW COLUMNS FROM ticket;

+-------------+---------------------+------+-----+---------+-------+

| Field | Type | Null | Key | Default | Extra |

+-------------+---------------------+------+-----+---------+-------+

| priority | tinyint(3) unsigned | YES | | NULL | |

| votes | int(10) unsigned | YES | | NULL | |

| name | varchar(16) | YES | | NULL | |

| category | varchar(64) | YES | | NULL | |

| description | varchar(512) | YES | | NULL | |

+-------------+---------------------+------+-----+---------+-------+