System/Subsystem Design Description for the Issue Tracker Web Application

Team 8:

Kyle Castle

Aaron Korb

James Lee

Samuel Moran

John Reed

Version History

Date	**Version**	**Description**	**Contributors**
11/17/2019	1.0	Document fully filled out	Sam Moran, John Reed

Table of Contents

1. Scope

1.1 Identification

This SSDD document shall provide a detailed description of the Issue Tracker website; The website shall provide the customer with a means for clients to communicate, track, and resolve issues in cooperation with service agents.

The Issue Tracker system was designed in coordination with Dr. Kazi Ahsanullah under Dr. Musad Haque.

1.2 System Overview

The Issue Tracker system is designed to provide an easy-to-use issue resolution ticketing system for a customer to use in coordination with a service agent (in particular, the system was designed for *Clean Energy Partners* in Melbourne, Australia). The website shall provide the means for creating and submitting a ticket for a specific issue, a dashboard for service agents to see all the tickets in play, and the comment history for each and every ticket. This project began in collaboration with a former-employee at *Clean Energy Partners*, Kazi Ahsanullah, Ph.D. *Clean Energy Partners* has been using an Excel spreadsheet to document issues that customers were having and passing the document between employees for amendments. This system is highly inefficient and poses risks for the overall workflow of projects. The Issue Tracker system was conceived to resolve this issue.

1.3 Document Overview

The purpose of this document is to describe the Issue Tracker system architectural design in detail. This document specifies how the Issue Tracker system has been implemented to meet customer requirements, how the system is organized, and how customers can interact with it.

Section 1, Scope: Identifies the Issue Tracker system and provides a brief description of the system.

Section 2, Referenced Documents: Identifies all outside references made within this document.

Section 3, System-wide Design Decisions: Identifies decisions that were made regarding the system's behavioral design and the system's components.

Section 4, System Architectural Design: Identifies the Issue Tracker system's architectural design and functional flow.

Section 5, Requirements Traceability: Presents traceability from each system component identified in this document, as well as the traceability from each requirement to the component to which it is allocated.

2. Referenced Documents

CMSC447-FA2019-G08-SRS-1-0

This is the Systems Requirements Specification document that was used as a way of first enumerating the requirements and needs for the Issue Tracker system.

3. System-wide Design Decisions

3-d: Safety, security, and privacy requirements

3-e: Hardware/Hardware-software systems

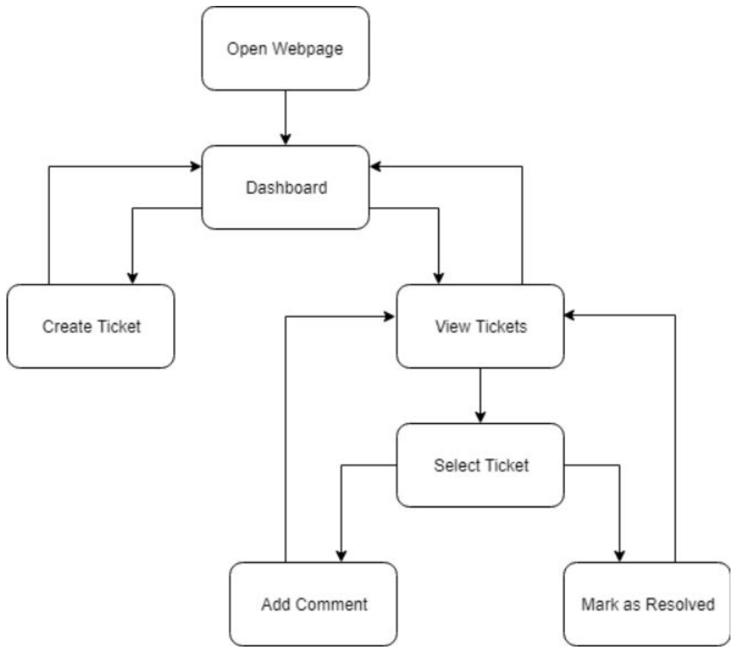
3-f: Other system-wide design decisions

3.1 Interface Design Decisions

The system as a whole consists of three HTML pages (Dashboard, Create Ticket, and Ticket Comments), a functional PHP interface, and an SQL database.

The main form of input and output is through the creation, deletion, and updating of "Tickets". Tickets are objects that consist of a Title, a contact name, a contact email, and a description. Each ticket stored in the database has an associated thread of comments, where updates to the issue associated with an individual ticket are recorded. The role of the SQL database is to store tickets, the role of the PHP interface and the HTML pages is to provide access to these tickets in the SQL database

3.2 Interface Flow Diagram



4. Issue Tracker System Architectural Design

This section shall describe the Issue Tracker's system architectural design. For illustrative purposes, the Issue Tracker's components will be provided project-specific identifiers to act as shorthand for the illustrative UML diagrams that will show the various components of the Issue Tracker, as well as the static relationships among the components. As the Issue Tracker revolves around relationships between the back-end SQL databases, intermediary PHP functions that handle information retrieval/insertion, and front-end pages for user-friendliness, it is necessary to show how these work together.

4.1 System Components

The following are project-specific identifiers.

4.1.1 Front-End PHP/HTML Files

Below are the four files that are integral to user-experience. These files provide a graphical interface for the user to interact with.

Project Identifier	**File Function**
HTML-NT	“Create a New Ticket” page
HTML-TD	“Ticket Dashboard” page
HTML-CH	“Comment History” page
HTML-M	Main “Welcome” Page

4.1.2 PHP Intermediary Functions

The files below contain PHP code that interfaces between the front-end user-friendly code and the SQL functions. These files provide a means for taking user-provided data for a new ticket or adding a comment and pushing that data into a SQL database. Likewise, these functions pull data from SQL databases.

Project Identifier	**File Function**
FXN-NT	Insert new ticket
FXN-TD	Ticket info retrieval for dashboard
FXN-CH	Comment dashboard retrieval
FXN-VALID	Input validation
FXN-CHADD	Add new comment

4.1.3 SQL Database

The SQL database stores all the data related to the issue tracker. Said data is contained in two tables: one for the storage of ticket information (SQL-T), and one for the storage of the comment information (SQL-C) associated with a specific ticket by a ticket's unique ID number.

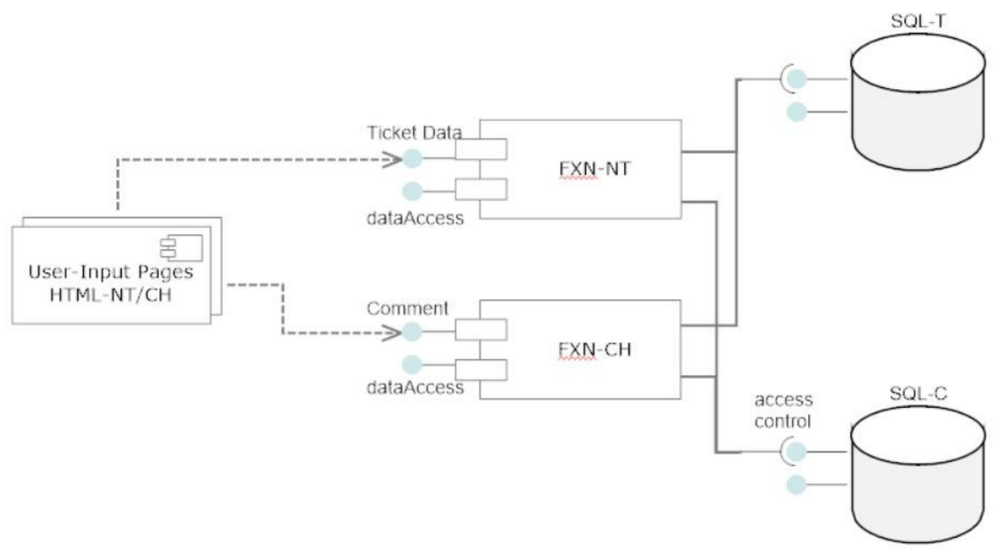
Identifier	**Data Fields**
SQL-T	id, title, description, status, poc_name, poc_email, modified_date
SQL-C	id, t_id, name, comment, date

4.1.4 Web-Hosting Service

There are many options available for hosting a project as small as the Issue Tracker. For example, one may obtain a server and host it within a LAN environment, or just simply download the files onto a computer and set up an XAMPP environment. However, for the purposes of this demonstration, a web-hosting site was used called "000webhost" (project identifier: HOSTSRVC).

4.1.5 Component Diagrams

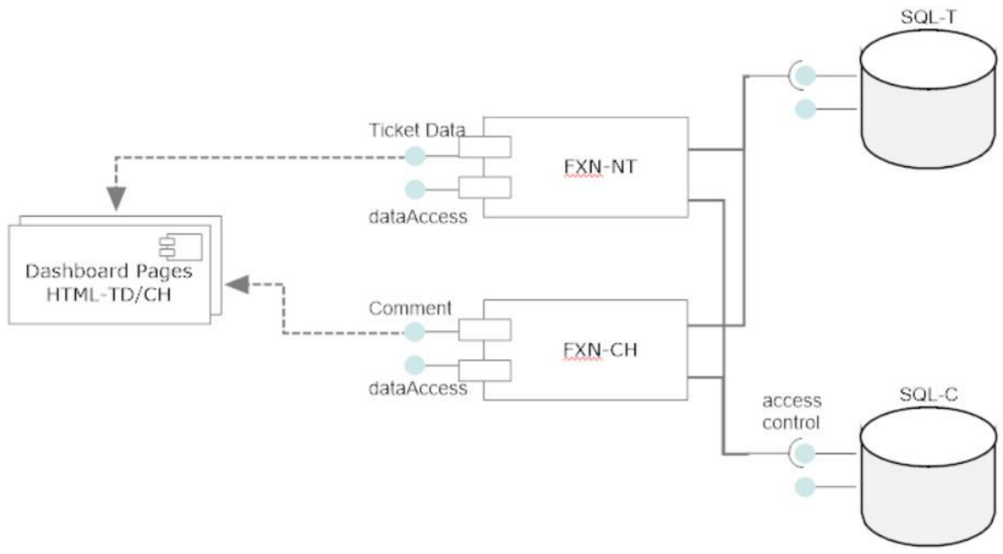
UML Component Diagram: Issue Tracker



The UML Component Diagram above depicts how the Issue Tracker website takes in user input for a ticket (or a comment), feeds the data into intermediary PHP/SQL functions to sanitize and prepare the input, and then place the data into the appropriate SQL ticket and comment table.

The UML Component Diagram below depicts the website components, namely how the dashboard display pages make use of intermediary PHP/SQL functions to query the SQL database tables (Tickets and Comments) to display the tickets and comments.

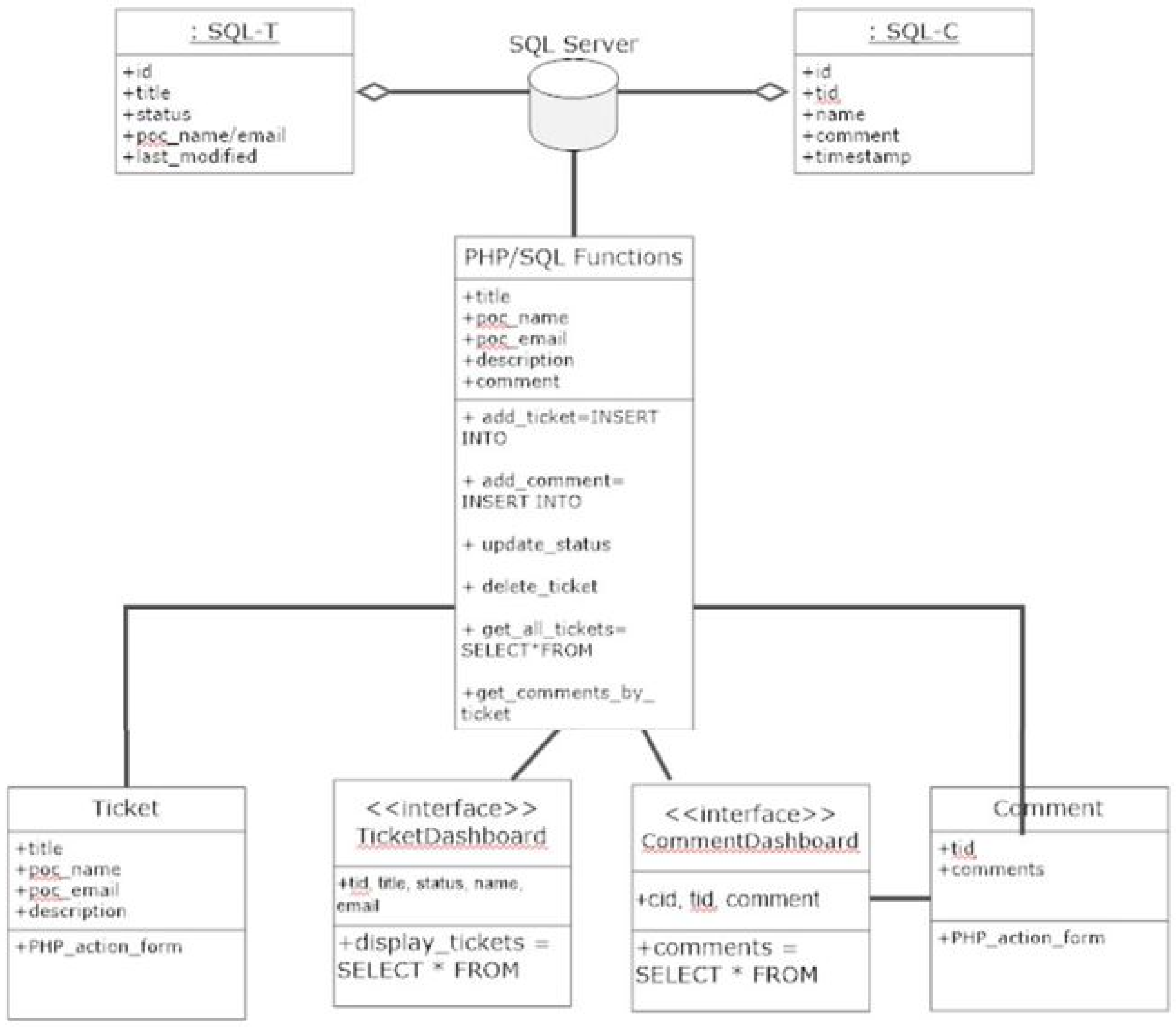
UML Component Diagram: Issue Tracker



4.1.6 Class Diagram

The Class Diagram below further depicts the relationships among the various components of the Issue Tracker system, namely how the Ticket and Comment classes feed data into the intermediary PHP/SQL functions that organize the data for the SQL database, which contains two tables: Tickets and Comments.

Class Diagram: Issue Tracker

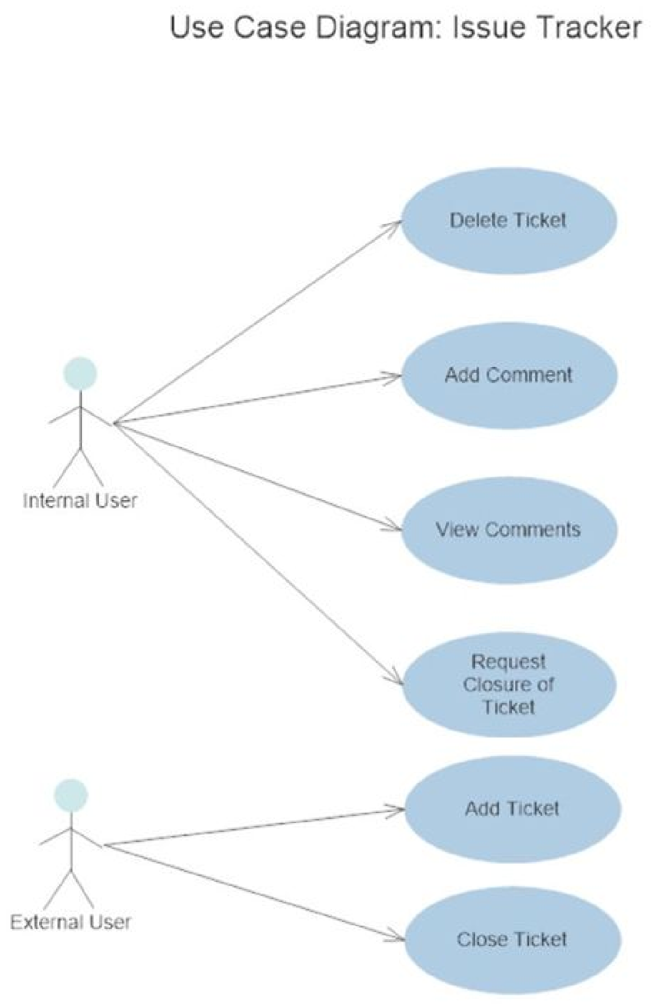


4.2 Issue Tracker Concept of Execution

This section describes the flow of data and execution control within the software. It details the dynamic interactions among the components.

4.2.1 Interface Diagram

This subsection presents the flow of execution from the user perspective.

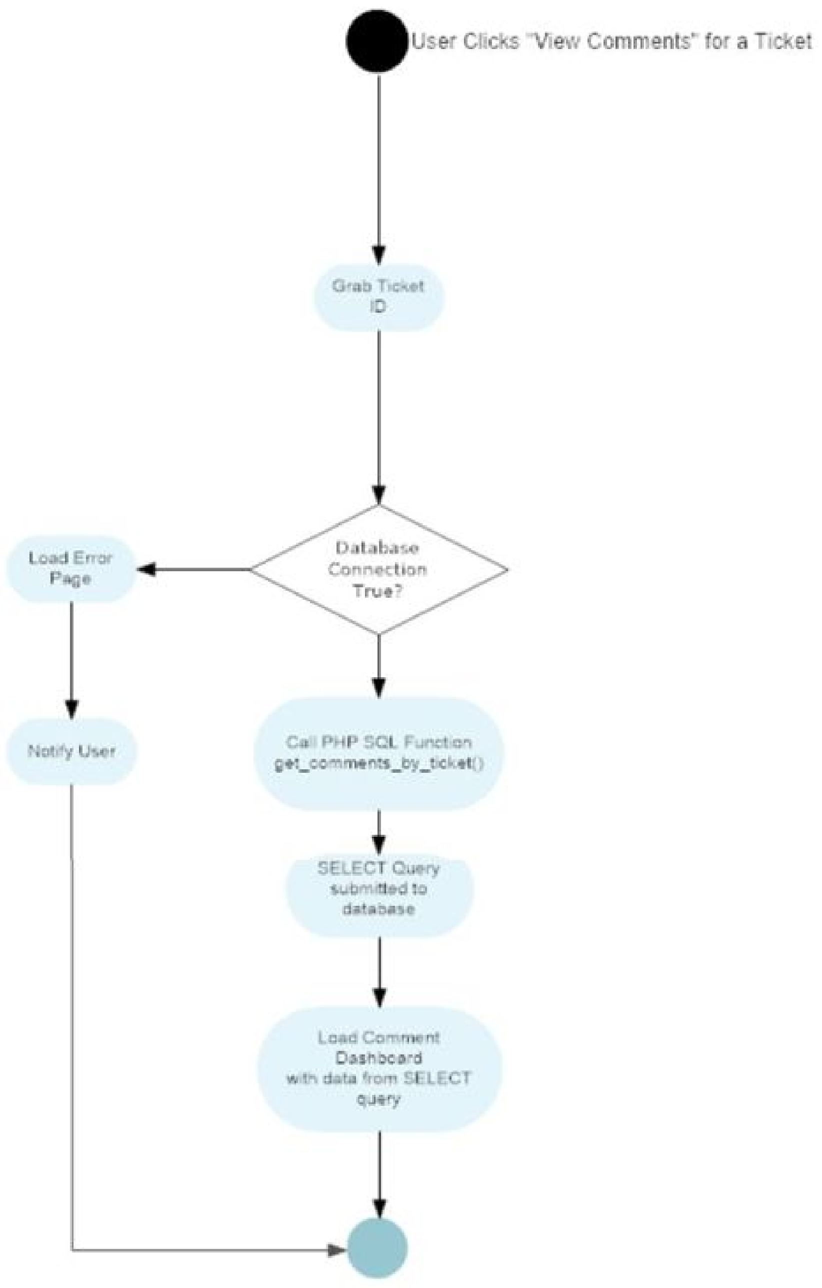


4.3 Issue Tracker Interface Design

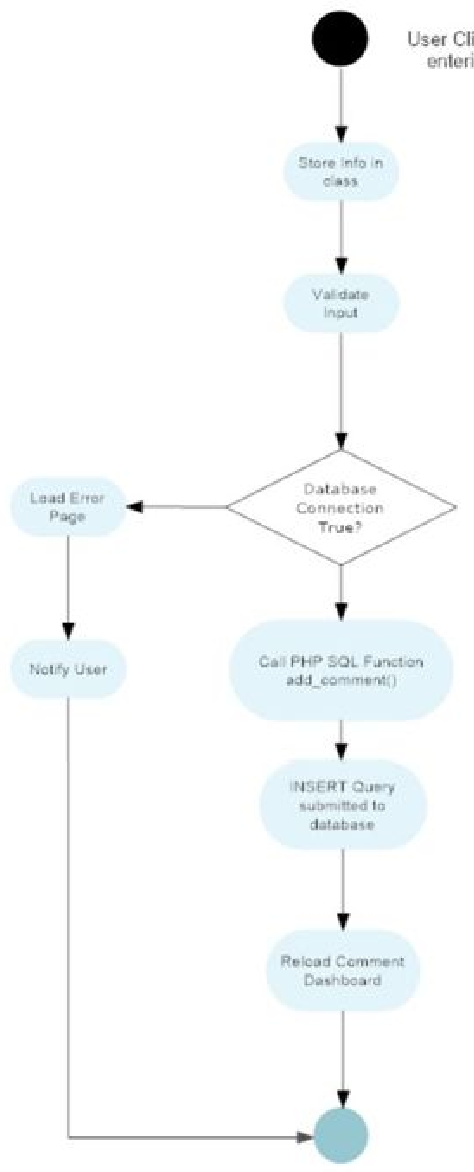
4.3.1 Interface Diagrams

The following diagrams detail how user input (for viewing and adding comments) interfaces with the components to deliver/update the comments for a ticket.

Issue Tracker View Tickets Activity Diagram


User Clicks "View Comments" for a Ticket
Grab Ticket ID
Database Connection True?
Load Error Page
Notify User
Call PHP SQL Function get_comments_by_ticket()
SELECT Query submitted to database
Load Comment Dashboard with data from SELECT query

Issue Tracker Add New Comment Activity Diagram


User Clicks "Add Comment" after entering relevant information
Store Info in class
Validate Input
Database Connection True?
Load Error Page
Notify User
Call PHP SQL Function add_comment()
INSERT Query submitted to database
Reload Comment Dashboard

5. Requirements Traceability

This section contains a matrix detailing what requirements of the Issue Tracker website have been found to be in need of modification or unfeasible given the time limit to accomplish the project.

Requirement	**Description**
SR-#3.2.1.4-3.2.1.7	The Issue Tracker website does not yet have a means for alerting the ticket's issuer if a ticket is to be closed. It is expected that an alerting mechanism will be implemented before the project's deadline. The Issue Tracker's ticket dashboard does not yet have a means to change a ticket's status from "Open" to "Closed - Pending Review" and "Closed". It was decided that an "In-Progress" status was not needed and would only confuse customers and service agents.
SR-#3.2.2.1-3.2.2.3	A front page has not yet been implemented. It has been deemed that this page would take the least amount of time to construct and is not the most important, functional part of the overall project, so it will be implemented last.
SR-#3.2.2.11	It was deemed unnecessary for a customer to provide the contact information of a specific Service Agent. In the real world, when a customer submits a ticket to a customer service platform, they do not need to provide the name of a service agent.
SR-#3.8	Testability was the most difficult to implement uniformly among the project's developers. As each developer provides new functionality to the website, the individual would test their changes locally before pushing the final changes to a central GitHub repository for everyone to integrate into their own local folders.
SR-#3.9	With Dr. Kazi Ahsanullah's job change during the project, much time has elapsed between the first meetings with Dr. Ahsanullah and today. This has meant that consistent demonstrations have not been possible. To remedy this, the developers will continue the project and try to regain contact with Dr. Kazi to provide a demonstration.