CMSC 447 ⧫ Software Engineering 1 ⧫ Professor Druffel



Code Inspection Report

# Crisis Response Ticket System for Shawn Davis by Team Cloud

**Team Members:** David Foster, Jack McGrann, Julie Nau, Patrick Wheeler

**Last updated:** November 15th, 2020

*Crisis Response Ticket System*

Code Inspection Report

**Table of Contents**

[**Crisis Response Ticket System for Shawn Davis by Team Cloud**](#_heading=h.x1mm5bfjk8x3) **1**

[**Introduction**](#_heading=h.gjdgxs) **3**

[Purpose of This Document](#_heading=h.30j0zll) 3

[References](#_heading=h.1fob9te) 3

[Coding and Commenting Conventions](#_heading=h.3znysh7) 4

[Defect Checklist](#_heading=h.2et92p0) 4

[**Code Inspection Process**](#_heading=h.tyjcwt) **6**

[Description](#_heading=h.3dy6vkm) 6

[Impressions of the Process](#_heading=h.1t3h5sf) 7

[Inspection Meetings](#_heading=h.4d34og8) 7

[**Modules Inspected**](#_heading=h.2s8eyo1) **9**

[**Defects**](#_heading=h.17dp8vu) **11**

[**Appendix A – Team Review Sign-off**](#_heading=h.26in1rg) **14**

[**Appendix B – Document Contributions**](#_heading=h.lnxbz9) **14**

**Document Versioning Control**

|  |  |  |  |
| --- | --- | --- | --- |
| Version Number | Date | Changes from Previous Version | Author |
| 1.0 | 11/4/20 | Original Document built from Template | Team Cloud |
| 1.1 | 11/8/20 | Cover Page, Sections 1.1, 1.2 added | Team Cloud |
| 1.2 | 11/13/20 | Sections 1.1, 1.2, 1.4, Intro for Section 2, and 2.1 Added | Team Cloud |
| 1.3 | 11/15/20 | Final version created | Team Cloud |

# Introduction

## Purpose of This Document

The purpose of this document is to develop a comprehensive set of coding and commenting conventions for the development team to follow. Having these requirements set as a standard for all the code in the system will allow the development team to more easily identify inconsistencies and errors in the system as well as address these issues in the most efficient and appropriate manner. These standards will also help to provide easy readability to allow developers to easily make future modifications or improvements on the system. In this document we also outline all code inspections that were performed on the code segments of the system that have been developed so far so as to identify as many errors early on in development as possible and to ensure we are still building the system as it was requested by the client. We present all defects found in each segment of code, and also give our thoughts on the code inspection and peer review processes.

## References

Druffel, K. (2020, October 27). 447-10-27-peer\_review\_process. Retrieved November 4, 2020, from <https://umbc-my.sharepoint.com/:w:/g/personal/kdruffel_umbc_edu/EfEEJH3ee_pMsYq7Q8TYcscBig-nRx6Z6WJdg4aKHrFQEg?e=QmOBDA>

Druffel, K. (2020, October 27). 447-10-27-Implementation-Phase. Retrieved November 4, 2020, from <https://umbc-my.sharepoint.com/:p:/g/personal/kdruffel_umbc_edu/ESF8sTEz_LNOnoUoq_jGpQoBWEZ4qag3vVX6t9rU9GZBew?e=YyUmBL>

Druffel, K. (2020, October 29). 447-10-29-VV-Peer-Review. Retrieved November 4, 2020, from <https://umbc-my.sharepoint.com/:p:/g/personal/kdruffel_umbc_edu/EQyq5wmOoFpDs_YInbms0VEBus6NuOuyysZMpWHfC54uoQ?e=330IQD>

Druffel, K. (2020, September). 2-ProjectDescription-CalltoMission. Retrieved November 4, 2020, from <https://umbc-my.sharepoint.com/:w:/g/personal/kdruffel_umbc_edu/EYVFEiSwyVJHv5VKzjr2OJUBkPo4klqrugJTDvI-cusDCA?e=SbTP27>

Foster, D. McGrann, J. Nau, J. Wheeler, P. (2020, October 4). System Requirement Specification. Retrieved November 4, 2020 from <https://docs.google.com/document/d/1AmYUFUAu0lfXBrhUMcs8OTf3v3z33YLIvxKWJdgAxdk/edit>

Foster, D. McGrann, J. Nau, J. Wheeler, P. (2020, October 25). System Design Document. Retrieved November 4, 2020 from <https://docs.google.com/document/d/1Q4s01lPTVFuPf3dLJaQZkk2kOyCFJNJCad9Ifspoz4A/edit>

Foster, D. McGrann, J. Nau, J. Wheeler, P. (2020, October 25). User Interface Design Document. Retrieved November 4, 2020 from <https://docs.google.com/document/d/1soKtDNpGoPIUKSBBHk-8vfZ29GVELDdiWIaKQDhTPog/edit>

Mae, F. (1996, May 29). The Hunter Group, Inc. PeopleSoft Implementation Development Standards. Retrieved November 4, 2020, from <https://umbc-my.sharepoint.com/personal/kdruffel_umbc_edu/_layouts/15/onedrive.aspx?id=%2Fpersonal%2Fkdruffel%5Fumbc%5Fedu%2FDocuments%2FUMBC%2DCMSC447%2F2020b%2DFall%2DKDruffel%2F20%2DBBDocuments%2F3%2DDocument%2DTemplates%2FFNMA%5FDEVSTNDS%2D1996%2Epdf&parent=%2Fpersonal%2Fkdruffel%5Fumbc%5Fedu%2FDocuments%2FUMBC%2DCMSC447%2F2020b%2DFall%2DKDruffel%2F20%2DBBDocuments%2F3%2DDocument%2DTemplates&originalPath=aHR0cHM6Ly91bWJjLW15LnNoYXJlcG9pbnQuY29tLzpiOi9nL3BlcnNvbmFsL2tkcnVmZmVsX3VtYmNfZWR1L0VhZ2E2UmlkcllSSXBvanZoMkNlalNrQlNrbUlOZk9KQmZobk1McHBMZmpIS0E_cnRpbWU9c0hjTUx2Q0kyRWc>

## Coding and Commenting Conventions

For coding and commenting conventions, our team has adopted the popular methodologies employed by UMBC professors. This is a simple strategy that is neither too strict nor too lenient, which aims to make sure all code is easily readable and understandable, with proper attribution given to the author and a clear definition of each file and method in the code.

Here are several UMBC professorial sources for the coding conventions used in this project:

Tang, Jason “Coding Standards” <https://www.csee.umbc.edu/~jtang/cs411.s20/homework/codingstd.html>

UMBC 201 Spring 2019 “Python Coding Standards” <https://www.csee.umbc.edu/courses/undergraduate/201/spring19/docs/CMSC%20201%20-%20Python%20Coding%20Standards.pdf>

UMBC 341 Fall 2020 “CMSC 341 - C++ Coding Standards - Fall 2020” <https://swe.umbc.edu/~donyaee/341/f20/projects/coding-standards.html>

All of these sources advocate for the same basic principles. Clear and readable code, commented files and methods, meaningful variable names, and ideally future-proofing it with modular code that can be easily updated and improved later on.

With that in mind our code is written so that each different view is created by a separate HTML file, called by a specific python method. This allows us to easily work on separate parts of the application simultaneously without running into versioning control or data hazard concerns.

## Defect Checklist

In this section we will be showing the list of possible defects that each member of the development team was responsible for searching for in each code inspection report for every segment of code that was reviewed. These defects are organized into categories and each type of defect in these categories is given a severity level defined as follows:

|  |  |
| --- | --- |
| **Severity** | **Description** |
| Critical | Must be fixed before go-live, no reasonable workaround, significant impact, or potential impact, to the system |
| Major | Must be fixed soon after go-live, workaround available or system will function without impact to the user. |
| Minor | No impact to the user, to be fixed when other parts of the code are being fixed. |

|  |  |  |
| --- | --- | --- |
| **Defect Checklist:** | | |
| **Defect Category** | **Defect** | **Severity** |
| Coding Conventions | Fails to adhere to the coding conventions as defined in section 1.3 | Major |
| Commenting Conventions | Fails to adhere to the commenting conventions as defined in section 1.3 | Minor |
| Logic Errors | Conditionals are not correctly set | Critical |
| Wrong variable name is used | Critical |
| Variable is set to incorrect data type | Critical |
| Data | Undeclared variables, or variables used before declaration | Critical |
| User input is not converted to the proper data type to be input into the database | Critical |
| Proper relations are not maintained between tables in database | Critical |
| SQL queries are not performed correctly or efficiently | Minor |
| Interface | Specifications on each user input for validity have not been made clear to the user | Major |
| Interface does not handle SQL injections correctly | Critical |
| The list of selectable options are not correctly displayed to the user (when applicable) | Major |
| Control | Sections of the code will never be reached or utilized by a user or administrator | Minor |
| User is locked into or trapped in certain views without a way back to their initial interface | Critical |
| All use cases are not properly handled or addressed | Major |
| Exception Management | The system does not reprompt user for missing data or inputs | Major |

# Code Inspection Process

This section outlines the Code Inspection Process that Team Cloud undertook. It describes in detail the exact steps that were taken by the development team to develop each Code Inspection Report. After this description we provide our honest impressions of the process overall, as well as indicating which two coding modules out of the ones inspected we would rank as the best and worst in terms of the likelihood that they have remaining flaws. At the end of this section we provide a log of when and where every code inspection peer review took place as well as which segment of code was being inspected.

## Description

For each code inspection peer review, virtually hosted on Google Meet or Discord, the author of the code to be inspected would share their screen and give a walkthrough presentation of each part of their code. During this time, every member that was attending the meeting played the role of inspector and recorder. Each of these members are responsible for documenting any defects as well as any other comments, questions or concerns about the code in a shared document without interrupting the author presenting. One of these team members will also serve the role of moderator, and is responsible for making sure the code presentation stays on track and at a good enough pace for themselves and the other inspectors and recorders to make the necessary notes for the inspection process.

Once the author is finished presenting, the moderator will clean up and compile the notes from all the other team members into a comprehensive report to be sent to the author, following the format laid out in section 4 of this document while still including all of the critiques. At this time, the author answered or asked any clarifying questions about the criticisms raised before making any of the specified changes at their own discretion.

Our main goal for the code inspection peer reviews was to be able to complete them in the most respectful way possible and to ensure that the author of the code being inspected did not have to feel overwhelmed or in any way attacked when presenting their contributions. It is for these reasons that we chose to slightly modify the standard peer review process first by making all of the criticisms only presented in a written format after the authors presentation is done, and second by making all of the criticisms anonymous when compiling them into one report for the author. In this way we were able to complete the peer review process in the most respectful manner possible while minimizing any friction between team members. It was also important to keep the final deliverable in mind, which is another reason we adopted the written and tabular format for reporting defects.

## Impressions of the Process

The code inspection report process has proven to be an effective tool for peer review of code. It gives a framework and a context to carefully judge each team member’s coded contribution against the chosen ideal framework. We’ve been able to identify flaws in our code when compared to the coding and commenting conventions outlined in section 1.3.

During this process we had the opportunity to learn how each other codes, and to accurately and fairly judge each other’s work. The process has added extra work to our already busy schedules, but it was overall a positive experience through improving each other’s coding process and working to improve our efforts through constructive criticism.

The most robust module of our system is the SQL Database. It provides a comprehensive schema, proper key constraints, and is securely hosted on a remote AWS server.

Our weakest module is the Bootstrap Stylization. At the moment, this system is simply underdeveloped and the tech-stack involved is new to everyone. As such, it may prove the biggest vulnerability to our design.

## Inspection Meetings

In this section, we will give a log of when, where, and how each code inspection took place. We will present, in tabular format, the date, method of communication used, time started and ended, participants, the roles of the participants, and which sections of code were analyzed for each code inspection that took place.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Date | Method of Communication | Time Span | Participants + Roles | Code Segment |
| 11/11/20 | Google Meet | 3pm-4pm | **Author:**  Jack M  **Inspector/Recorder:** David Foster  Patrick Wheeler  Julie Nau  **Moderator:**  David Foster | SQL Database Code, also includes inserts for all prepopulated sample data. |
| 11/13/20 | Discord | 3-4pm | **Author:**  David F  **Inspector/Recorder:** Jack M  Patrick W  Julie N  **Moderator:**  Jack M | Python Flask code and accompanying HTML/CSS pages for basic website functionality. |
| 11/14/20 | Discord | 3-4pm | **Author:**  David F  **Inspector/Recorder:** Jack M  Patrick W  Julie N  **Moderator:**  Patrick W | Python Flask code to allow for connection and verification of user input within the website to MySQL AWS database. |
| 11/15/20 | Discord | 3-4pm | **Author:**  Patrick W  **Inspector/Recorder:** David F  Jack M  Julie N  **Moderator:**  David F | Integration of Flask and Bootstrap to standardize and simplify the layouts and styles of related pages. |
| 11/15/20 | Discord | 4-5pm | **Author:**  Julia Nau  **Inspector/Recorder:**  David F  Jack M  Patrick W  **Moderator:**  David F | Python Flask and CSS code to manage user interface elements. |

# Modules Inspected

Modules provide the basis for any successful software; by breaking an application into functionally distinct components, the overall complexity, turn-around time, and internal constraints of developing a system are reduced. Team Cloud decided to implement the crisis response system as a two-tier web app and, as such, maintains two primary modules. Using a Python Webshell, our front-end provides both an interface into, and logic for maintaining, the system's MySQL back-end.

TODO:

|  |  |  |
| --- | --- | --- |
| **Module Name** | **Functionality** | **Estimated Completion Date** |
| Bootstrap Stylization | Provides a method for maintaining customizable, reusable templates on common web-page elements (e.g. nav-bar, content containers, etc.). | 11/22/2020 |
| UI Methods | Provides various methods to handle user interface in relation to information retrieved from the database. | 11/22/2020 |

|  |  |  |
| --- | --- | --- |
| **Module Name** | **Functionality** | **SDD Equivalent + Changes** |
| SQL Database Code | Implements the database described in section 3 of the SDD in MySQL. It also generates inserts for all pre populated tables and any sample data already in the system for demo purposes. Also creates and grants permissions to users connecting to the server. | MySQL Database hosted on AWS’s Relational Database Service. Changes: The ERD from section 3 of the SDD was used as a functional guide for implementing the database in MySQL. In order to be able to do this, the lists of Agents and Vehicles in the Mission table were removed, and a foreign key pointing to which mission they were assigned to was added to the Vehicle and Agent tables. A phone number was also added to the Tickets, Mission, and Service Location tables in order to be able to contact the victim, mission managers, and operation chiefs respectively. The attributes labeled Map\_X\_Coordinate and Map\_Y\_Coordinate were changed to Longitude and Latitude so as to avoid future confusion. A Users table was also added in order to allow users to log in to their user view based on what kind of user they are. |
| Python Webshell | Contains the libraries and methods necessary to connect the SQL database to the web application. Queries the database and displays information through HTML pages. | UI and Business layer both in one. This code manages the business rules for the project (python code) while also managing the user input and display to the user (HTML/CSS). The UI specifics have not yet all been implemented based on the SDD but it is a work in progress moving towards that ideal. |
| Bootstrap Stylization | Provides a method for maintaining customizable, reusable templates on common web-page elements (e.g. nav-bar, content containers, etc.). | Inherent relation to the aforementioned Python Webshell module. Bootstrap provides a wrapper for templated web components; it combines HTML elements and CSS style attributes. This is a higher-level system than the SDD and UIDD originally intended (pure HTML, CSS, JS), but provides better modularity. |
| UI Methods | Provides various methods to handle user interface in relation to information retrieved from the database. | Corresponds to the UI layer along with the Python Webshell. |

# Defects

This section presents each defect found in every code inspection in a tabular format. Each defect will have the name of the module it was found in, which category of error it falls under (based on the Defect Categories given in section 1.4), a description of the defect, the severity of the defect, and finally it’s resolution.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Module** | **Category** | **Description** | **Severity** | **Resolution** |
| SQL Database Code | Commenting Conventions | Comments are too generic, should describe why tables are built the way they are as well. | Minor | To be fixed after full functionality is achieved |
| Logical Errors | Wrong Variable Name Used; Accidentally switched Map\_X\_Coordinates and Map\_Y\_Coordinates | Critical | Fixed; Variable names changed to Longitude and Latitude to avoid further confusion |
| Data | Proper relations are not maintained between tables in the database; The foreign key relation for Assigned\_Mission for both Agent and Vehicle tables was set to cascade on delete; i.e. they would be removed whenever a mission was removed. Agents and Vehicles should not be deleted when a mission is completed or deleted, and should be able to be reassigned | Critical | Fixed; No longer set to cascade on deletions |
| Coding Conventions | SQL Coding Convention violated: foreign key relation definitions for Agent and Vehicle tables shared the same name for both their Service\_Location\_ID and Assigned\_Mission foreign key relations. | Critical | Fixed; FK relation names changed to be unique |
| Logic Errors | Wrong data type used and Variable is set to incorrect data type; Used the Description from the Event Type table for the Event\_Types in the Tickets table for all sample data instead of the Type\_ID that the foreign key relation is built on. | Critical | Fixed; Set to corresponding Type\_ID’s. |
| Python Webshell | Commenting Conventions | Code was not properly commented, needs author attribution | Minor | Fixed and re-uploaded to the shared folder. |
| Correctness | Final HTML/CSS output does not match UIDD sample views, missing functionality | Critical | Webparts to be added ASAP. |
| Other | Password entering field is not hidden | Critical | Not yet fixed, need to research. |
| Bootstrap Stylization | Data | Templated variables have undefined scope, rendering some unusable. | Major | Unresolved; researching solutions. |
| Other | Issues resolving associations between html files within a given user view. | Major | Not fixed. Attempting to get familiar with Flask and its scope resolution properties, as well as Bootstrap class syntax. |
| UI Methods | Commenting Conventions | Fails to adhere to the commenting conventions as defined in section 1.3 | Minor | To be fixed after functionality is achieved |
| Data | Proper relations are not maintained between tables in database; Vehicles and Agents not updated when Missions are updated | Critical | To be fixed before final release |
| Interface | Information retrieved from the database is not yet properly and usefully displayed to the user. | Critical | To be fixed before final release |

# Appendix A – Team Review Sign-off

By signing below, all team members have reviewed the document and agree on its content and format. The comment section is to be used for minor unresolved issues.

|  |  |  |  |
| --- | --- | --- | --- |
| **Typed Name** | **Electronic Signature** | **Date Signed** | **Comments** |
| David F | David Foster | 11/15 | Good work team |
| Jack M | John McGrann | 11/15 |  |
| Julie N | Julia Nau | 11/15 |  |
| Patrick W | Patrick Wheeler | 11/15 | We are awesome :) |

# Appendix B – Document Contributions

As Implementation leader, Jack M is in charge of this document.

|  |  |  |
| --- | --- | --- |
| **Team Member** | **Contribution** | **Estimated Percent Work** |
| David F | Logo, Sections 1.3, 2,2, respect parts of sections 2.3, 3 and 4. | 25% |
| Jack M | Cover Page, Sections 1.1, 1.2, 1.4, Intro for 2, 2.1, and respective parts of sections 2.3, 3 and 4. | 40% |
| Julie N | Respective parts of sections 2.3, 3, and 4, and Appendices | 15% |
| Patrick W | Intro for section 3, helped with section 2.2, respective parts of sections 2.3, 3, and 4. | 20% |

Customer Sign-off:

As Implementation Leader, I (Jack M) take responsibility for forgetting to send a link to the document to the customer at least 24 hours ahead of the due date. This is the message sent to our customer that we sent to seek final approval:

From: John Mcgrann <jmcgra1@umbc.edu>

Date: Sun, Nov 15, 2020 at 3:51 PM

Subject: Code Inspection Report

To: Shawn Davis <Shawn.Davis@collabraspace.com>, David Foster <dfos1@umbc.edu>, Julie Nau <jnau1@umbc.edu>, Patrick Wheeler <pw2@umbc.edu>

Good afternoon Shawn,We are almost done finalizing all the details on the Code Inspection Report. Here is a list of everything that still needs to be completed before tonight at midnight:Things Still need to be completed on CIR:

-Patrick and Julie needs to fill out Sections 2.3, 3 and 4 for their code segments

-Intro for Section 3 (Patrick volunteered)

-The second part of Section 2.2 (Patrick volunteered to complete this if we do not hear from Julie)

-Formatting and updating page numbers, etc. and Final Submission (As implementation leader, Jack is in charge of this)

Everything that is not completed will follow a similar format to everything that has been done so far or needs to have everyone's contributions completed before we can finish off those sections. If you could please give it a look by the end of today and send us an email approval for final submission we would greatly appreciate it. Regards,Team Cloud