

**Operation Sundew**



November 14, 2019

Console Commander

https://www.consolecommander.net/

Contents

[1. Introduction 1](#_Toc24800427)

[1.1 Purpose 2](#_Toc24800428)

[1.2 Intended Audience 3](#_Toc24800429)

[1.3 Scope 4](#_Toc24800430)

[Benefits 4](#_Toc24800431)

[Objectives 4](#_Toc24800432)

[1.4 Definitions and Acronyms 5](#_Toc24800433)

[2. Overall Description 6](#_Toc24800434)

[2.1 User Needs 7](#_Toc24800435)

[2.2 Assumptions and Dependencies 8](#_Toc24800436)

[3. System Features and Requirements 9](#_Toc24800437)

[3.1 Functional Requirements 10](#_Toc24800438)

[3.2 External Interface Requirements 11](#_Toc24800439)

[3.3 System Features 12](#_Toc24800440)

[3.4 Nonfunctional Requirements 13](#_Toc24800441)

# 1. Introduction

We have a problem in which we are constantly using Microsoft Word to show what bugs we have and how we are attempting to solve them. We need something like Jira so that we can log bugs. We don’t’ have the budget for that.

My plan is to create Sundew a bug tracking software which can contain projects, the people assigned to the project and bugs for each project. So if I work with a team, we can use it to report bugs and solve them together as a group.

This will stop us from having to report bugs verbally and interrupting each other’s work or sending emails back and forth that contain duplicate descriptions about the bug.

As an optional requirement we can allow this system to email other parties upon logging of the bug including the sender as well.

Not just anyone can be able to use this so we will need a login system.

## Purpose

To make life easier for developers to stop from emailing duplicate bug reports to each other. To have a system that show bugs that are resolved and not resolved and have an automated process of emailing developers the bug status until app can be pushed into production or made useful for software development teams.

Purpose is also to make this application reusable by having backend database as reusable as possible for other CRUD business scenarios.

This software will operate on a server giving teams 24hr access.

## Intended Audience

Just me and any developers who collaborate in this project.

## Scope

We are developing bug tracking web software.

### Benefits

* Will use open source software that can run on any hosting environment.
* Team leader will be the only one who can create read, update and delete projects in the system. This will stop the team from creating duplicate projects.
* Increased productivity and less interruption.
* Centralized system that teams can tap into especially while remote.
* Allow team to track reported bugs, and comment on each one and be notified via email and application notifications of future progress including the status and severity of the bug.
* Allows team members to prioritize bugs in severity.

Objectives

|  |  |
| --- | --- |
| **Objectives** | **Complete?** |
| To allow team leader to create projects only. |  |
| To have title about the problem that tells a story in several different information fields, including the date/time/, ID, description, steps to replicate, who’s assigned to it. |  |
| To allow team to assign members to bugs including themselves. |  |
| Can login and out of system. |  |
| Can display information about bug. |  |
| Can write information about bug to database. |  |
| Can have backend information as generic as possible to be reused for other applications. |  |

[Risk table here](Risk%20Matrix.docx)

## Definitions and Acronyms

**Sundew:** The project which is named after a carnivorous plant.

**ASP.NET Core**: Active Server Pages. Open source web language that helps build dynamic web sites, applications and services

**Technical Debt**: The cost of additional rework, caused by choosing an easy (limited) solution now instead of using a better approach that would take longer.

**Developer**: A person who create a program.

**Automation Tester**: Someone who automates tests.

**Quality Assurance**: Someone who ensures the application is performing to client specifications.

**Product Owner**: Someone who drives product from business perspective. There are a customer intermediately.

**Scrum Master**: The Project Manager.

**Bootstrap**: CSS framework directed at responsive, mobile-first front-end web development.

**Entity Framework Core**: Open source ORM that allows .NET developers to eliminate the need for most of the data-access code.

**Auth 0**: a universal authentication & authorization platform for web, mobile and legacy applications.

**SQL Server:** A database management system.

**ORM:** Relational Object Mapper.

# 2. Overall Description

This will be a whole new product. We are going to attempt to back the backend as generic as possible in it’s naming structure so it will become more reusable. For example instead of calling a table tblBug we will call it tblProblem as every customer usually wants software to solve a business problem and not a bug in their system. We will also attempt to do the same with the front end, whenever possible.

## 2.1 User Needs

**Developers, quality assurance members/automation testers:** Will receive problem about the project they have all been assigned to and be notified by email and or text. They input additional info and update the problem creator in the chat log. The person who opened the bug is the one who can close it.

## 2.2 Assumptions and Dependencies

* I have never done this before.
* I assume anyone can create an issue that is a part of the project. I don’t feel I have to make separate fields to indicate what type of employee the user is.
* I don’t feel the project needs to have a set of permissions for different types of employees. The only real permission should be where the user who created the problem should be the same one who can close it and approve the status of whether the problem was fixed.
* I don’t feel we need to specify different types of users such as testers and developers. The people using this system are more concerned about the what, when, where, why and most importantly how. How do we solve the problem?

# 3. System Features and Requirements

## 

## 3.1 Functional Requirements

The software will have login for each user. There should be an option to indicate if the user is a leader of that job. If user is a leader then only they will be able to generate projects. This will stop users from creating duplicate projects. Each project will allow user to create, read update and delete many bugs.

## 3.2 External Interface Requirements

* **User:**

Frontend: ASP.NET Core MVC with bootstrap.

* **Hardware:**
* **Software:**
* **Communications:**

## 3.3 System Features

* User Interfaces

Front-end software ASP.NET Core.

* Hardware Interfaces
* Software Interfaces
* Communications Interfaces

## 3.4 Nonfunctional Requirements