Software Requirements Specification

for

Bug Tracker System

Prepared by Jack Wade

10/04/2021

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Jack Wade	10.04.2021	Initial version drafted	1.0.0

1. Introduction

1.1 Purpose

This SRS will cover a piece of software to track bugs and allow the end user to manage bugs within system design better, breaking down its sub systems and features. This is a personal project, and I am the only stakeholder for the design.

1.2 Intended Audience and Reading Suggestions

This SRS is setup for a developer, I will start with a general overview of what will be covered within this project and then proceed to break down the work into manageable sprints to complete the tasks at hand.

1.3 Project Scope

This piece of software is mainly aimed towards developers and dev managers, the core functionality will be managing bugs and allowing the users to manage, edit & complete tasks/bugs within the system. There will be authentication using OAuth2 for the project along with a DB connection to store the data. I want to cover all CRUD functions within this project and have a long-term vision of expanding the functionality to allow the user to report on what Bugs are within the system.

1.4 References

Bootstrap Templates - https://bootstrapmade.com/ninestars-free-bootstrap-3-theme-for-creative/

2. Overall Description

2.1 Product Features

- *Login Page – Oauth2*
- *CRUD Users to login*
- *Ability to CRUD bugs within system*
- *Table View of Current Open Bugs (Maybe Add Basic Filters)*
- *Ability to search for specific bug*
- *Connection & Interaction to DB (SQL)*

2.2 User Classes and Characteristics

- *Administrators – (CRUD Bugs & CRUD Users to login)*
- *Users – (CRUD Bugs)*

2.3 Operating Environment

ASP.Net (MVC) .Net Framework 4.7.2
SQL
Dapper – DB Connection
Bootstrap

2.4 Design and Implementation Constraints

Hosting options once complete.

3. System Features

3.1 Login Page

3.1.1 Description and Priority

The Login Page will be the point of entry for the application, I will handle the authentication using Oauth2. This will need to be clear and concise to the user what they must do.

3.1.2 Functional Requirements

LPA-1: Build MVC structure for page and layout buttons and text boxes appropriately.

LPA-2: Implement local authentication to DB or oAuth (twitter or Gmail TBC)

LPA-3: Create MVC landing page to be redirected to once authenticated.

3.2 Administrate Users Page

3.2.1 Description and Priority

This page will allow users with rights to access it the ability to grant access to users to view the data within the application. Grant CRUD rights to users when handling bugs within the application.

3.2.2 Functional Requirements

AUS-1: Build MVC structure for page and layout buttons and text boxes and labels appropriately.

AUS-2: Setup rights table in DB to handle all of the CRUD operations.

AUS-3: Create Dapper functionality to update permissions based on form data passed in.

AUS-4: Setup restrictive access to admin page based on user rights within DB.

3.3 CRUD Bug Items – Display current bugs to user

3.3.1 Description and Priority

This page will be the main page within the application, this is where the Bugs will be created, viewed, deleted, and updated. Its important that the data within this page is displayed in a clear format for the user to read.

3.3.2 Functional Requirements

CBU-1: Build MVC structure for page and layout data table view to display all of the bug items to the user. Have a button to handle creating a new bug item.

CBU -2: Create functionality to populate main view with all bugs from the DB using Dapper, give functionality to select items from the data view. (Split List bug view on left and item selected breakdown on right)

CBU -3: Create basic filter panel on the right side. (Filters TBC)

CBU -4: Create search option for specific bug.

3.4 Create form view when bug selected

3.3.1 Description and Priority

This will be apart of the Main Bug items page. TBC if I will build a modal to display selected item along side the list or take you to a whole new screen. Form shows all information for the bug along with allowing the user to add a comment to the bug and change any fields around the bug if required.

3.3.2 Functional Requirements

CFO-1: Build (Modal or New Page TBC) to show the form data retrieved from DB.

CFO -2: Create logic around a save button to pass fields back to DB and save any updates.

CFO -3: Build ability to return to list or change item selected and repopulate form.

4. External Interface Requirements

4.1 Software Interfaces

SQL DB, the connection Strings held withing app.config. Communication will be carried out via dapper.
oAuth will be used to handle authentication for the application.

4.2 Communications Interfaces

Amazon simple email service will be used to handle email notifications from the application to the user.

5. Other Nonfunctional Requirements

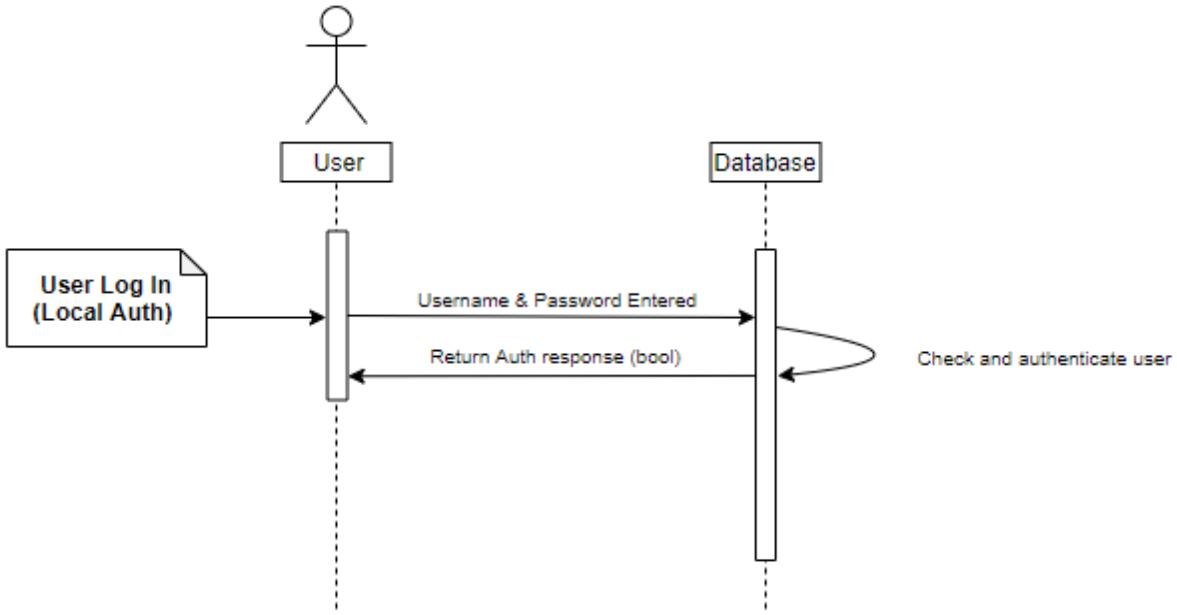
5.1 Security Requirements

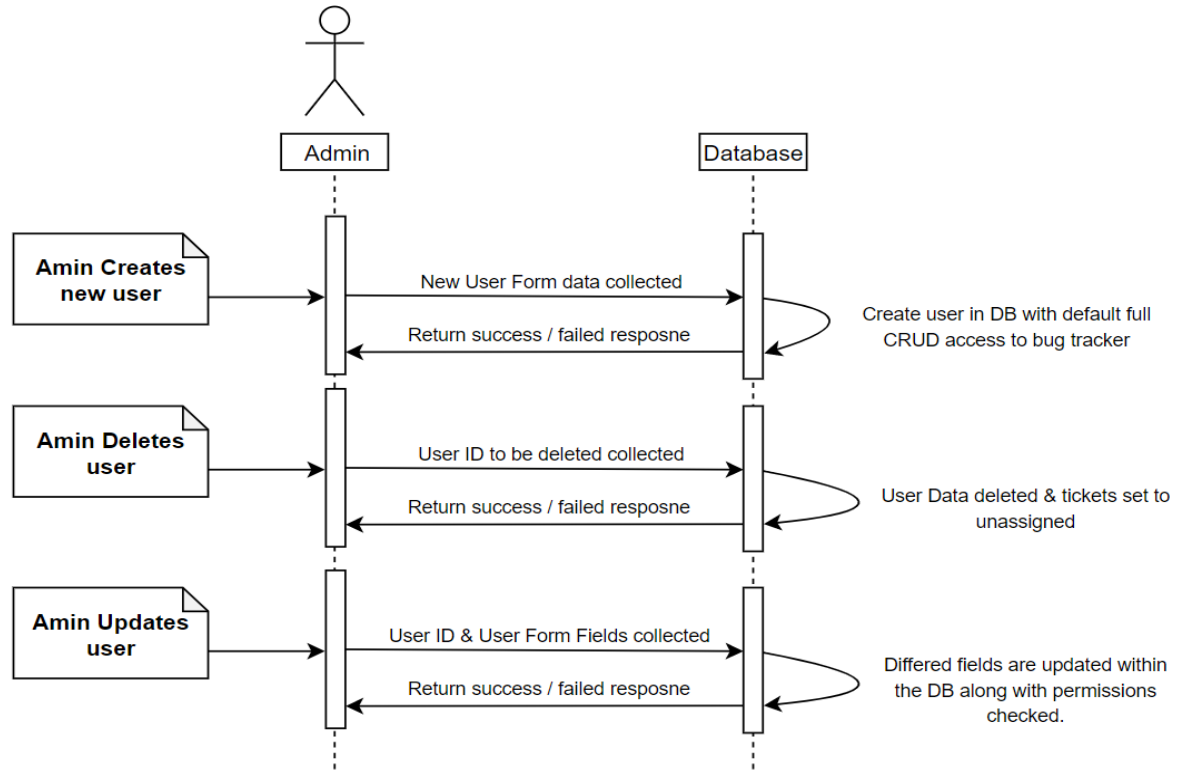
Authentication will be handled via two "Groups" of permissions – Admins & Users. This will allow the ability to grant access to the application to be safeguarded.

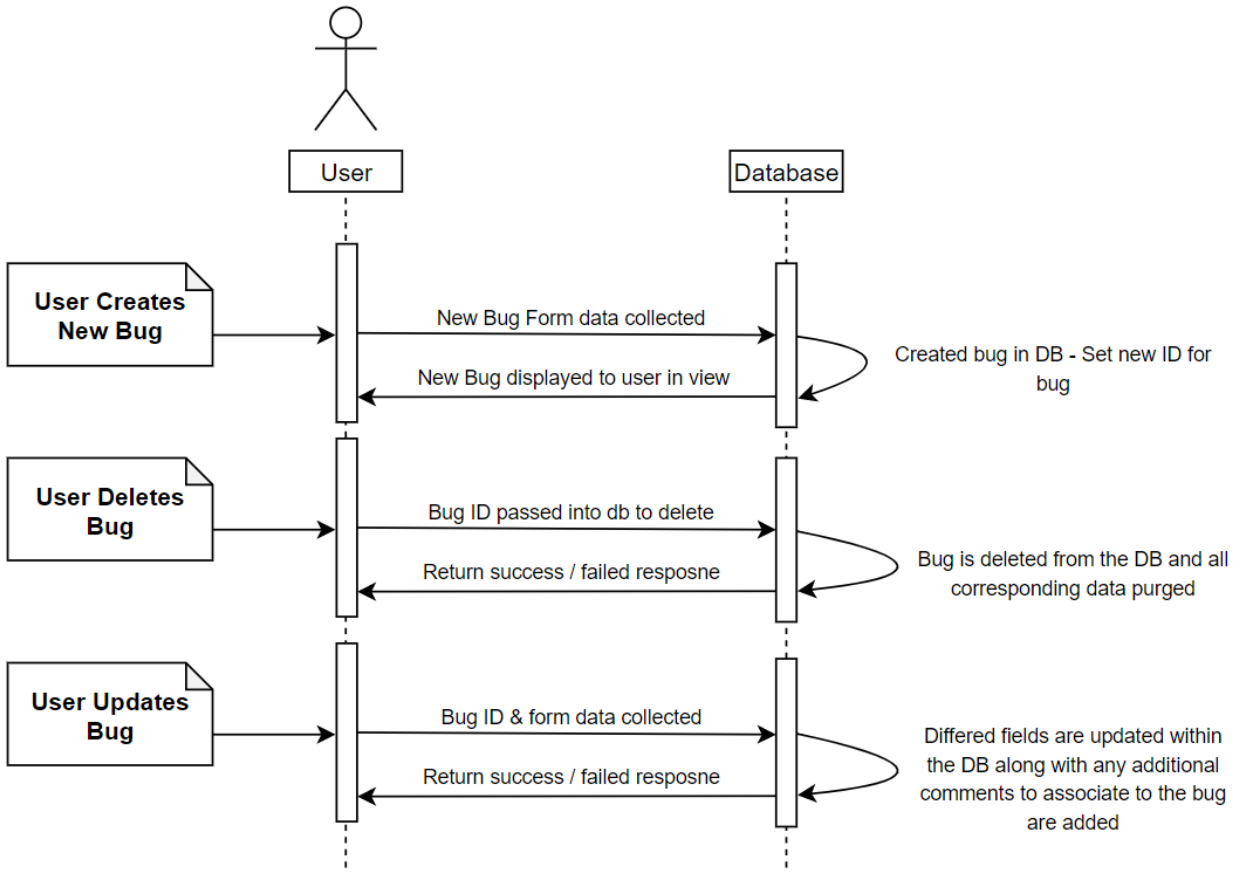
5.2 Software Quality Attributes

Flexibility, maintainability, cross functionality & user friendly.

6. Appendix A: Analysis Models







7. Appendix B: Issues List

TBD – When bug is selected unsure if it should be displayed as a new screen or a modal that is overlaid on the page.

TBD – What and how the filters on the main Bug View page will function and what data we will filter on

TBD – What methods of oAuth will be implemented to allow users to login.