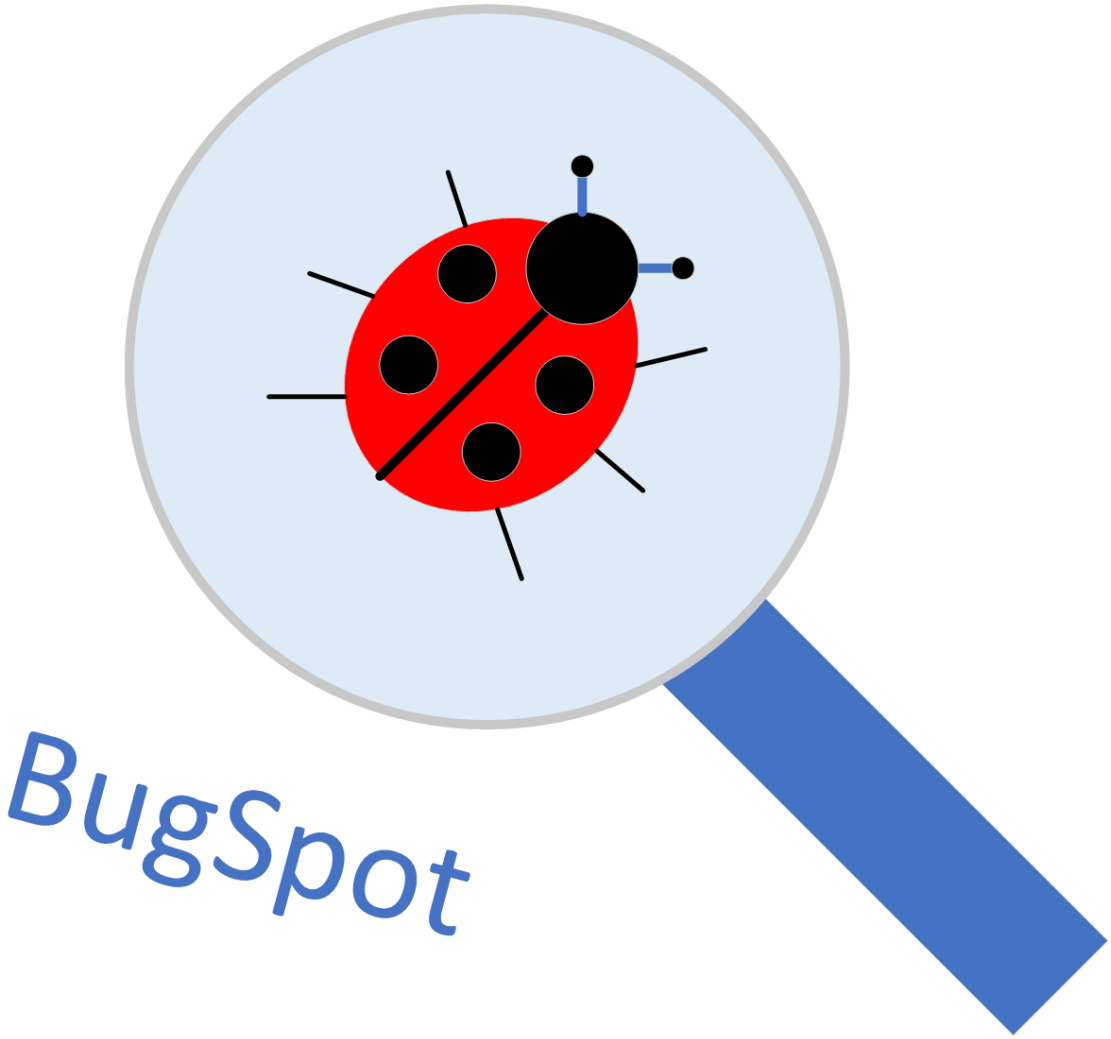
System documentation



By USN Software AS

Table of Contents

1. Introduction

This document presents and overview, requirements and design for a bug tracking system which is targeted for developers needing feedback from their customers to make improvements to the product or service they are providing. In addition to users being able to report and send in bugs to the developers of a product or service, the developers should also be able to post and show their users new and exciting features they have been working on.

2. System Overview

This basic overview shown in Figure 2-1 is a simplified overview of the system and the different modules for easy understanding of how the system is structured. The figure shows the system tied together by the cloud, which is where all information and applications will be stored. Users will be able to access the bug reporting section where they can submit bug reports, and or request features. Bugs and features are separated with different forms. The developers marked with orange will access their own module when logging in with credentials. This module will have access to read and modify reports that's been sent in. Mockups for graphical user interface can be seen in chapter 3.3 below.

Figure 2-1: Basic system overview

2.1 Architecture

The bug tracker system is built up by three modules, “Report”, “Status” and “Developer”. In Figure 2-2 the modules are shown as web applications made using ASP.NET. To differentiate between access levels there has been added color coding to the system sketch, which indicates their level of permission. Administrators or Developers are marked with Orange. Orange access level is the highest level of access for normal intended use. Normal users, bug reporters are marked in Green. Green has access to reporting bugs, viewing status, and looking at new features.

Developer module is where the developers have full access over the reports submitted. Orange access level will have CRUD (Create Read, Update, Delete) access. All reports can individually be claimed or assigned to a developer to be investigated. This will give options for freedom and flexibility when it comes to tackling a bug report.

Users will be able to access two of the three modules, with green access level they will be able to submit reports in the bug “Report” module while also being able to see the status of bugs in the “Status” module. The “Status” module is used to present the submitted reports and their current state. This sub module will also be available to those with green access level.

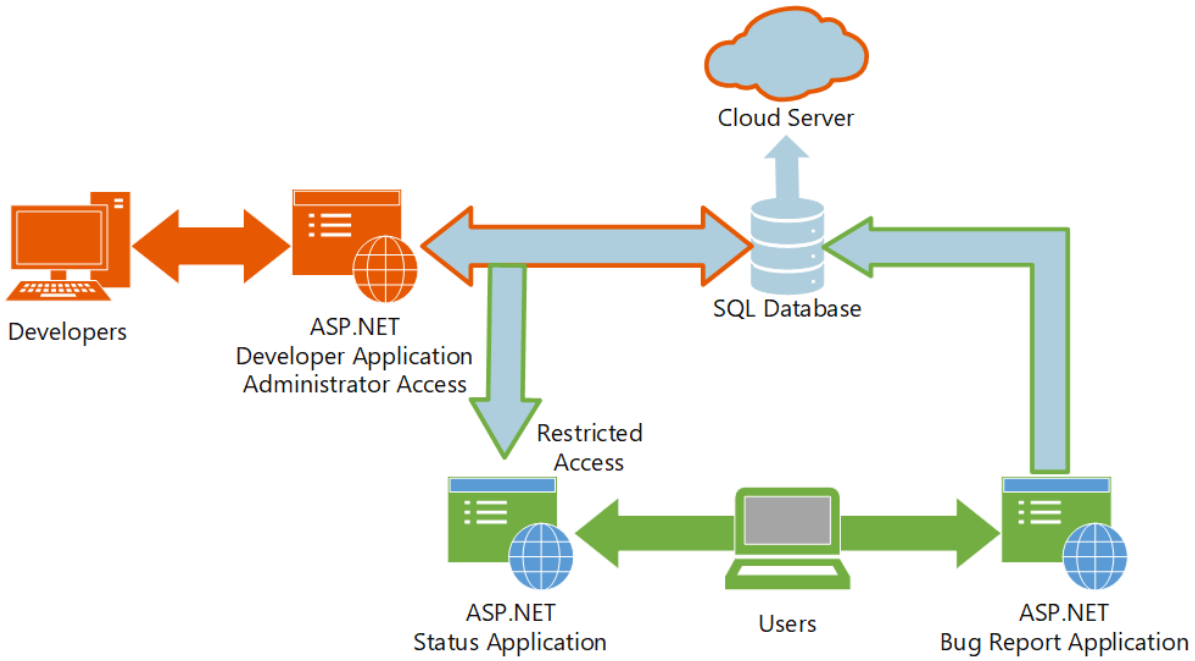


Figure 2-2: General architectural overview of the system

3. System Requirements

This chapter is about the satisfied requirements for the latest version of the product. The following subchapters describe what functional and non-functional requirements have been satisfied or why they have not. Further, an updated version of the “Graphical User Interface” (GUI) and user task flow per module are explained.

3.1 Functional Requirements

Functional requirements define the function for the system or one of its components. The following list shows how the specific requirements have been implemented:

- Data Storage
  - The data for the system is being stored on a database hosted on the Microsoft Azure cloud server. Multiple tables are used for storage of different information, see chapter 4 more detailed description on the database design.
- Data Collection
  - Data is submitted to the database using either the “Report” or the “Developer” modules. The “Report” module is the main source for all of the bug reports to keep track of while the “Developer” module allows for adding supplementary data, such as internal commentary meant to be read only by the developers.
- Data Management
  - The “Developer” module is made as the managerial module of the system where the developers can manage incoming reports by setting levels of priority and severity for each report, assigning a person to work on the reported problem, add additional software that they are working on and register other developers to expand their team. Additionally, there is a feature for internal commentary that only “Developer” module users may view for each of the reports.
- Viewing data
  - In the current version of the application there are two sets of data views, one designed for the regular user(s) and one designed for the administrative user(s). The key difference is the regular user having a more simplified view while the administrative version includes a more detailed view and options for control.
- Search Functionality
  - The final version includes search functionality by date and by software name. The date search is done from a starting date to an end date. The software search returns results where the software name starts by or includes the text string used to search with.
- Sorting of viewed data
  - The current version of the system has not yet implemented a freely sortable table as other functions and features have been prioritized due to lack of resources. Currently all views are sorted as default.

- User login
    - Due to the complexity, lack of resources, and time constraints this feature has not been prioritized and is therefore not implemented in the latest version of the system, however, the modules are designed with this feature in mind for future implementation.

3.2 Non-functional Requirements

The non-functional requirements specify features on how the system should work, such as:

- Input errors
  - Error messages appear when the submission fields are not filled out and attempts are being made to send the submission form. There are several input fields that are missing certain check procedures, such as checking that the string length does not exceed the set maximum of characters. This is due to a lack of resources during the programming phase.
- Authentication
  - Due to the login functionality not being implemented this feature is also not yet implemented.
- Web browser support
  - The current version of the system is working as intended on the following browsers: MS Edge, Chrome, and Opera. This is tested and confirmed. Other browsers are assumed to be compatible as well.
- Cross-module navigation
  - As of the latest version, the “Report” and “Status” modules are tied to each other where a user can easily navigate between them. This is intentional as these two modules are meant to be publicly available.
- Uniform graphic design
  - The current version of the system has the same general layout and style across all the modules with minor variations, most of which are found in the “Developer” module as it requires additional fields to display the extra information.

3.3 Graphical User Interface

One of the most important requirements are the graphical user interface (GUI) with the system with a special emphasis on the graphical design, which should be simple to understand at first glance and intuitive to use. This part of document presents the final version of the graphical user interface for each of the modules.

3.3.1 Report module

Upon entering the web application, the user is greeted with a starting page where they can select what kind of a report they wish to submit, see Figure 3-1.

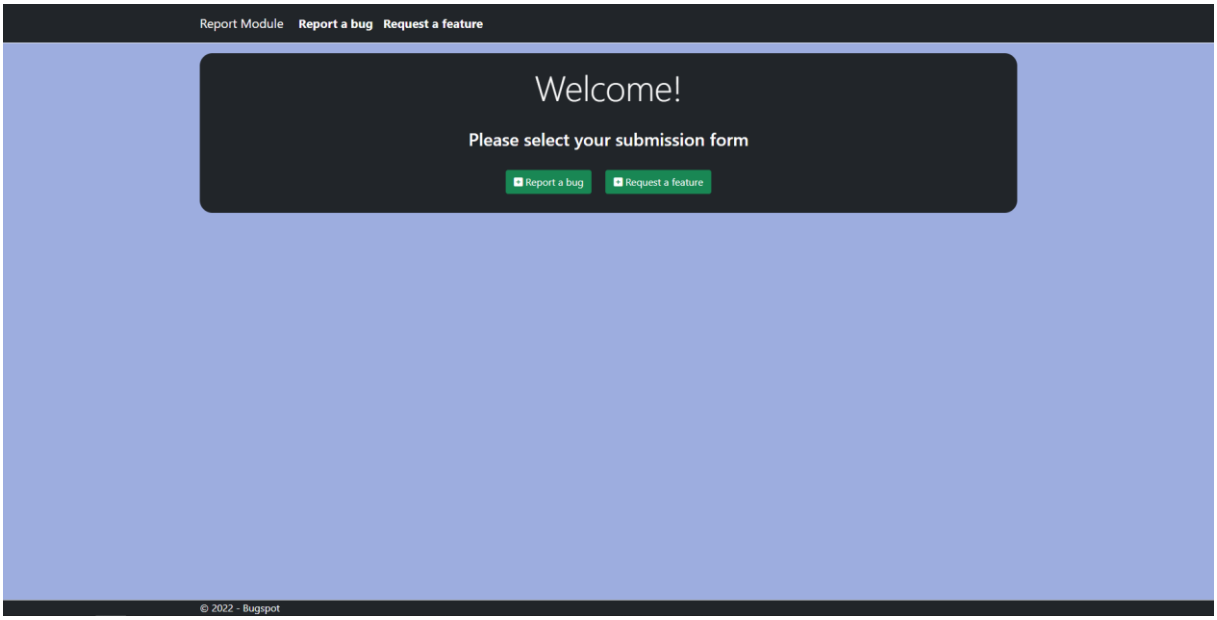


Figure 3-1: Starting page of the "Report" module (GUI) design

When the user has chosen a report type, they are redirected a simple form to fill in, shown in Figure 3-2, with relevant information and submitting it with the "Send Report" button which then takes them back to the Home-page where a message thanks the user for submitting a bug or feature. Additionally, the Home-page, shown in Figure 3-3 also offers options to submit another report or to view submitted reports in the "Status" module. The forms for both bug reports and feature requests are the same with only slight differences in the form title text.

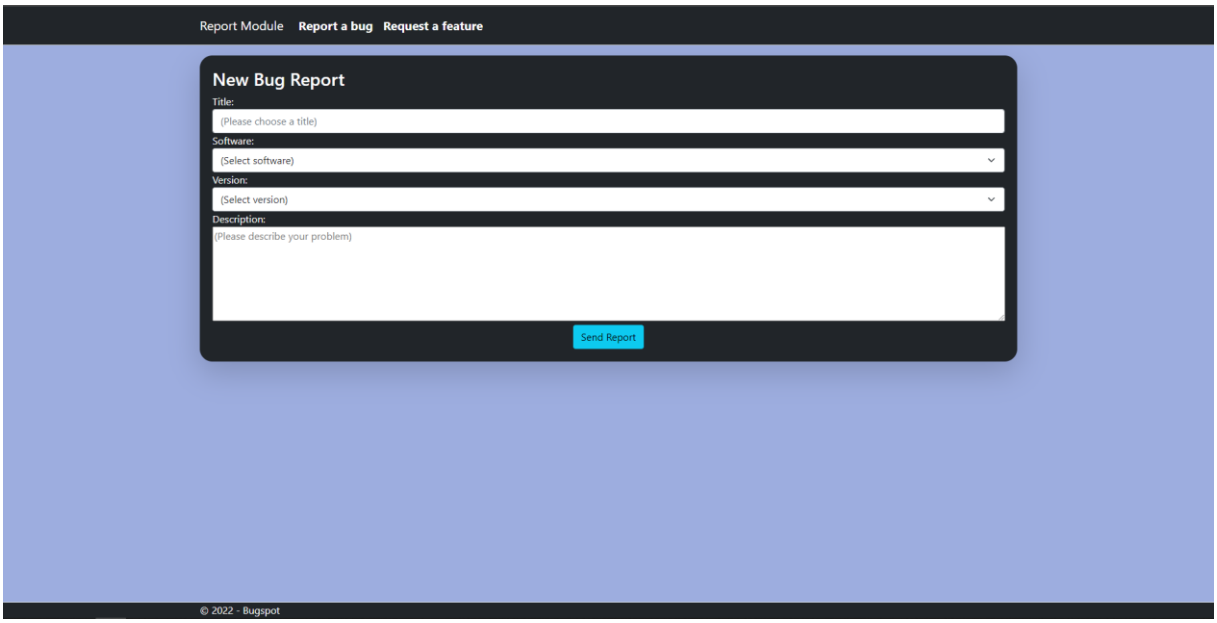


Figure 3-2: Bug Report submission form page

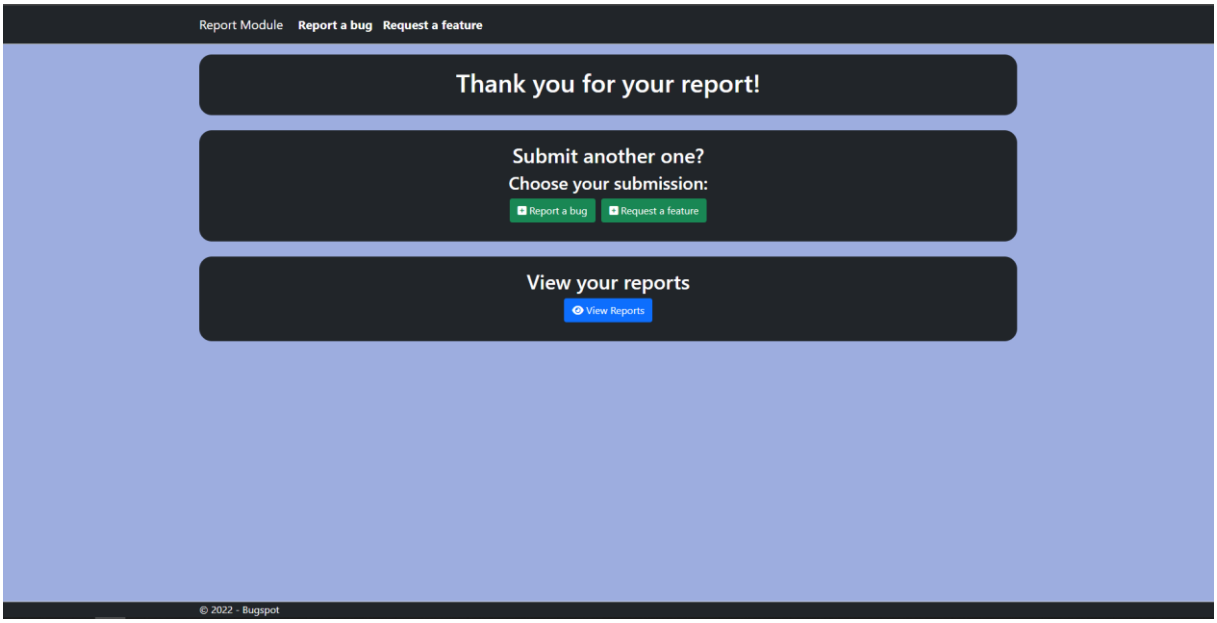


Figure 3-3: "Report" module Home-page

3.3.2 Status module

The "Status" module starts off with the initial page, shown in Figure 3-4, where a list of submitted reports is presented along with some search options. Either option takes the user to a very similar page, however this time it features the search option for date, see Figure 3-5, or the search option for software, shown in Figure 3-6. This creates a small optical illusion that the additional bar containing the search options appears on the original page whenever the user is switching between the options.

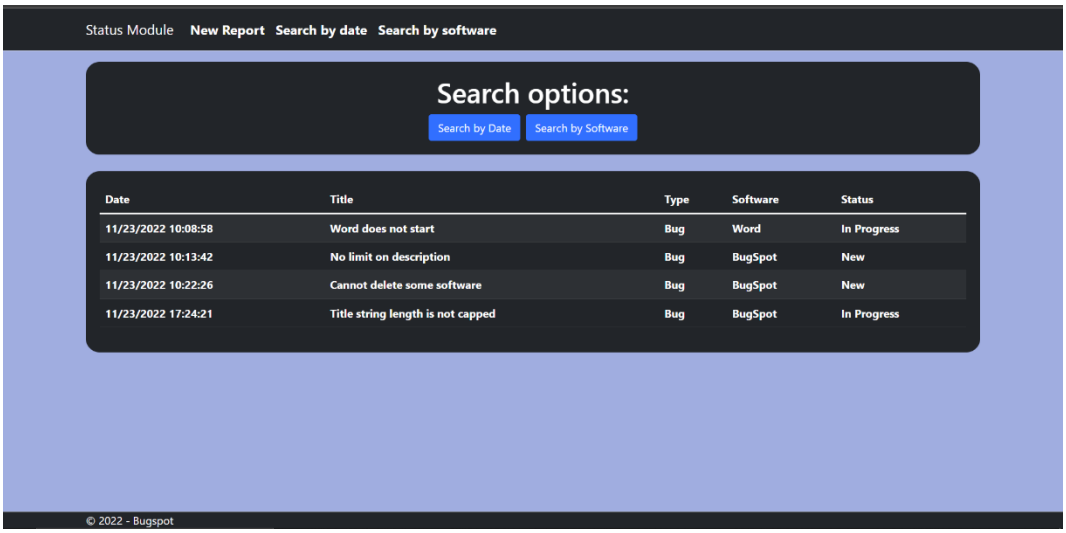


Figure 3-4: Starting page of the "Status" module

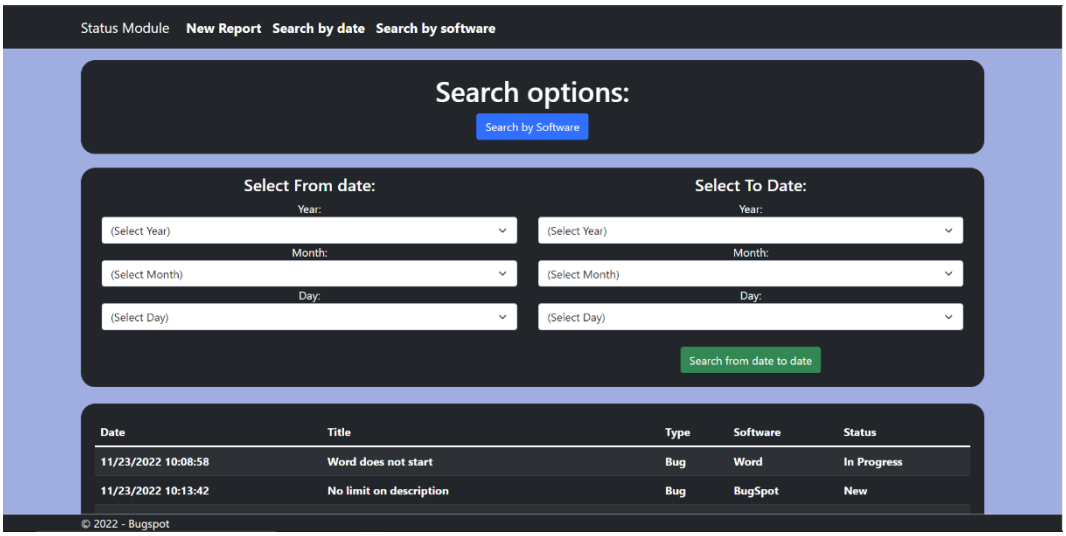


Figure 3-5: Date search option page

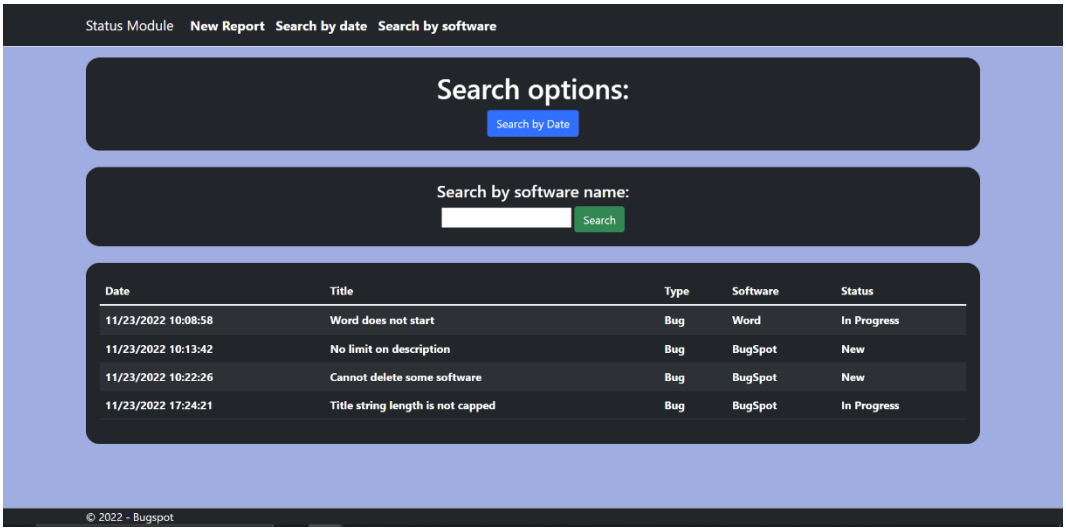


Figure 3-6: Software search option page

3.3.3 Developer module

The “Developer” module starts off with its own starting page, shown in Figure 3-7, where a list of all reports is shown. At the top of the list is a button “Add Report” which redirects the user to the “Report” module for submissions. To the side of each report are two buttons, one for detailed overview, which is the blue eye button, and the other one for deleting the report, the red trashcan button. Clicking the blue button directs the user to the developer view of the report, shown in Figure 3-8. Here the developer user has options for setting levels of priority and severity, setting a responsible person or adding internal commentary between developers.

Figure 3-7: Starting page of the "Developer" module

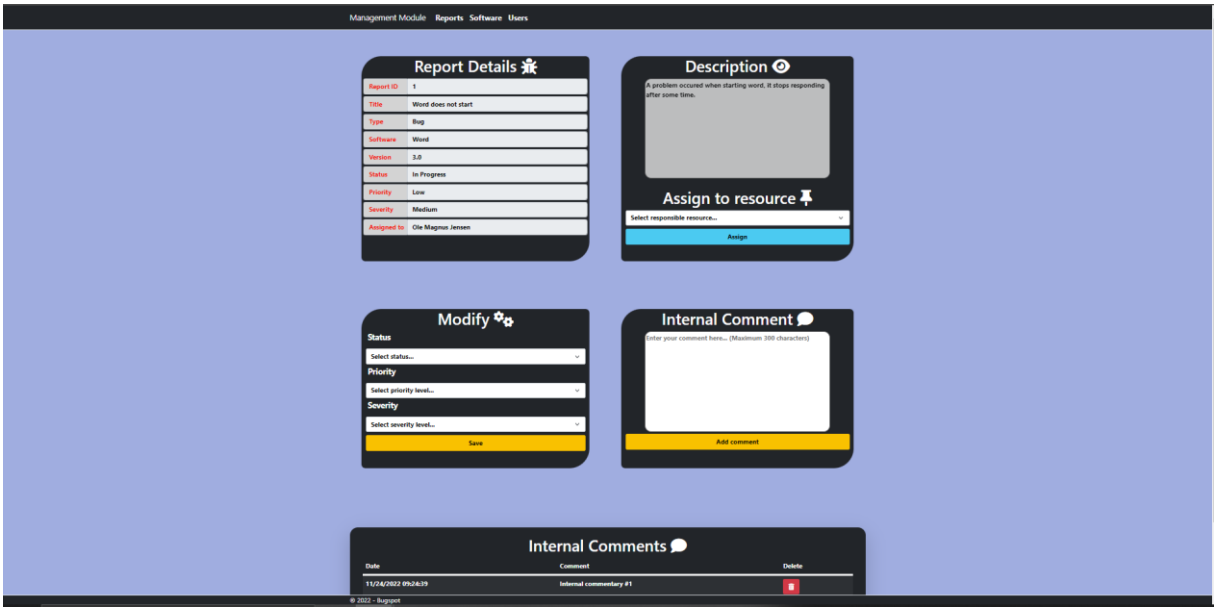


Figure 3-8: Developer report view

Next, is the software management page, shown in Figure 3-9, where the developers can add new software and edit or remove existing software in the database,

Figure 3-10: Software management Add new software page

and Figure 3-11 shows the simple forms for adding new software and editing the existing ones. Additionally, there is also a manager for the different versions for each of the software called “Version Control” which is accessed by clicking the green button next to each software. This then directs the user to the version control page, shown in Figure 3-12, where the user can view, add and remove the different versions for the selected software.

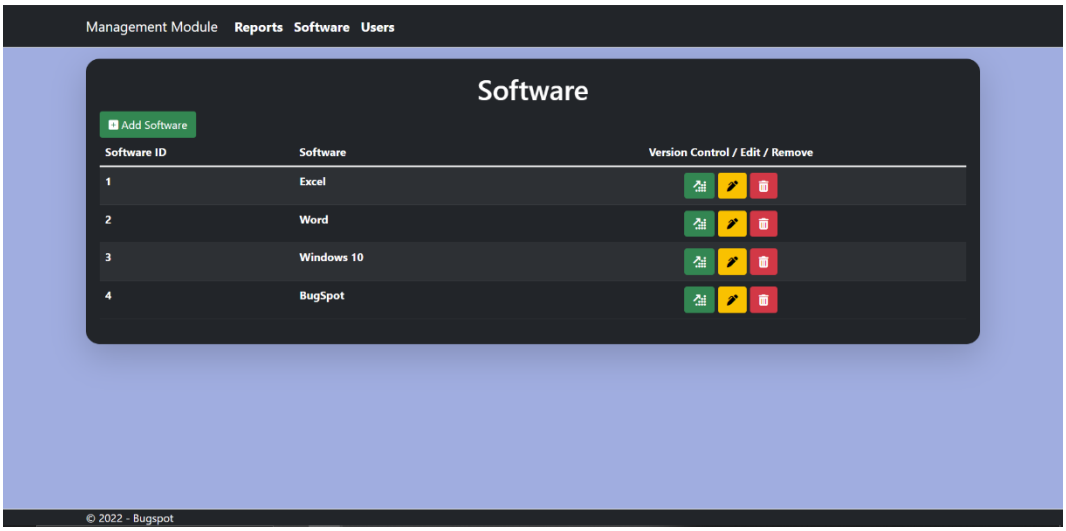


Figure 3-9: Software management main page

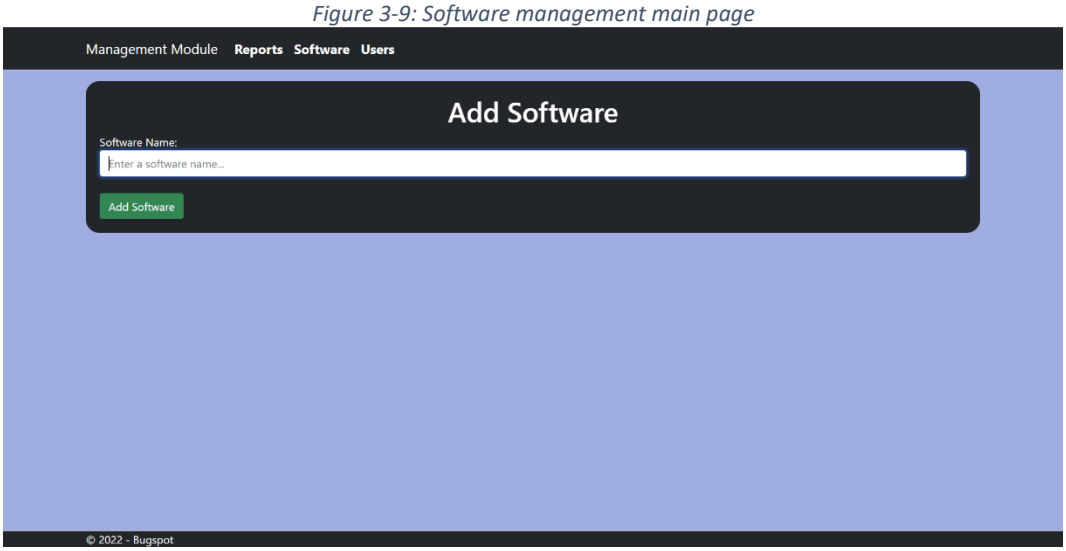


Figure 3-10: Software management Add new software page

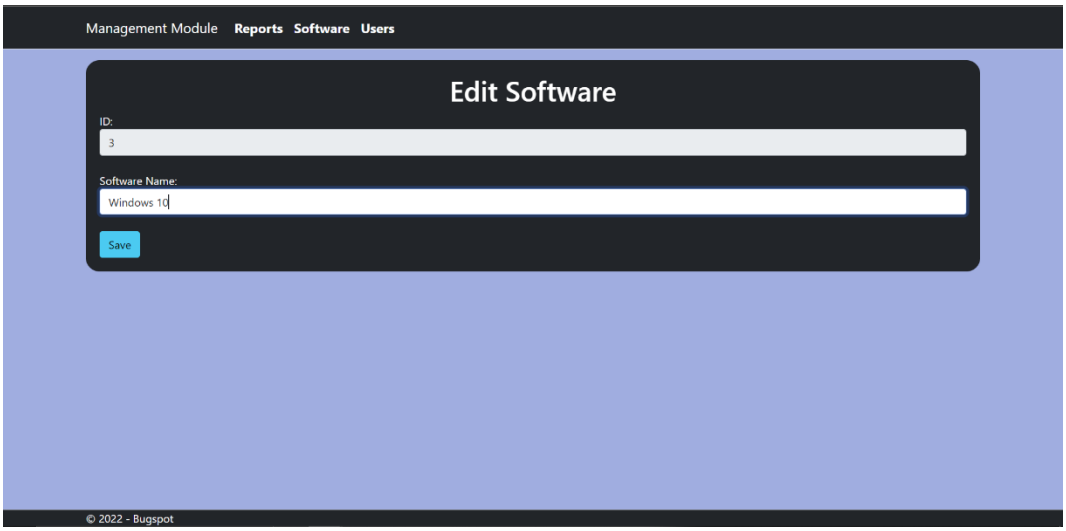


Figure 3-11: Software management Edit software

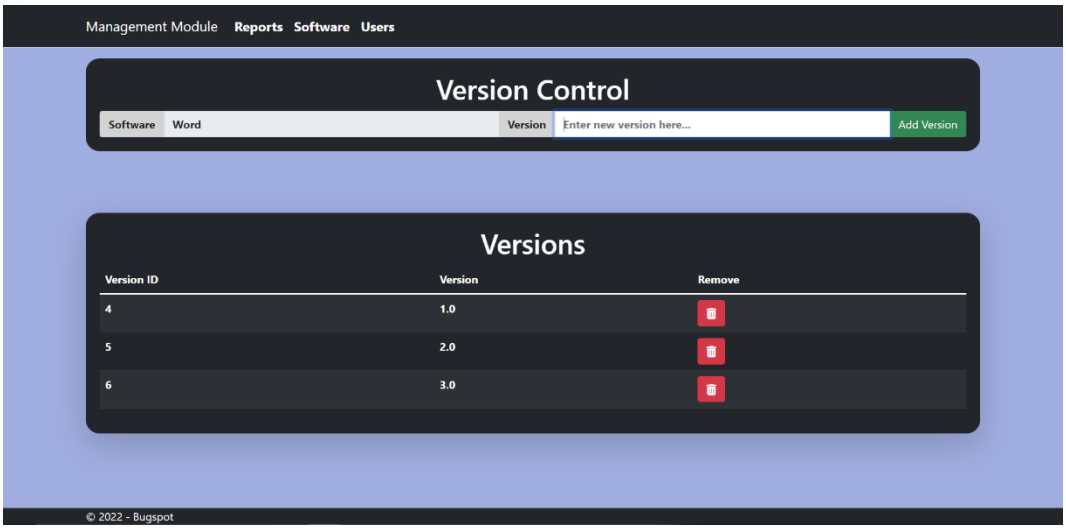


Figure 3-12: Software version control

Lastly, the “Developer” module features the User manager, shown in Figure 3-13, where the developers can add, edit and remove users of the “Developer” module, which is done in the same way as described in the software manager description earlier in the sub-chapter.

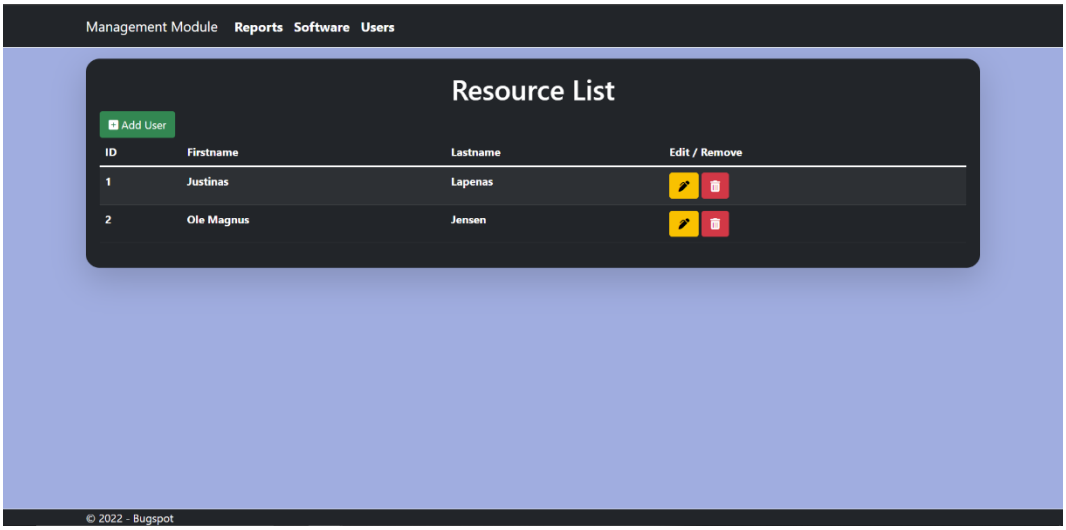


Figure 3-13: User management starting page

3.4 User Task Flow

This chapter will present how the bug report system modules work using flowcharts showing its behavior when browsing the web modules.

Starting with the “Report” module, shown in Figure 3-14, the module starts with a starting page which allows the user to choose the report submission form for either a bug report or a feature request. Once either of the two options is selected it leads to a form for the selected report submission (note that both submission forms are essentially the same, except for the form name). Here the user is required to fill out the form in its entirety before clicking the “Send Report” button. Should one or more of the fields are left out a small message box will appear, prompting the user to fill out the missing value. Upon a successful submission the user is redirected to a Home page for the module with a “Thank you” message and options to either submit another report/request or to view the reports in the “Status” module.

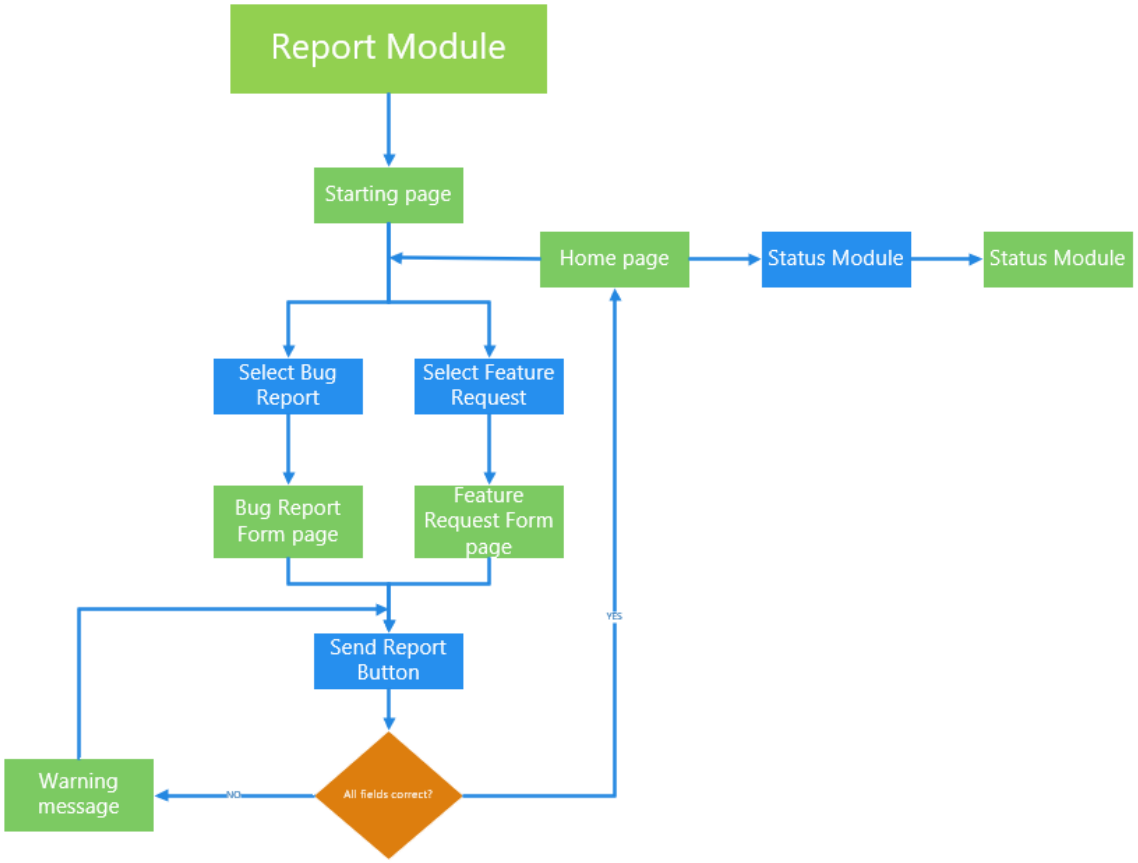


Figure 3-14: "Report" module user task flow

The next module is the “Status” module, which is used to display submitted reports to the non-developer users. The task flow for this module is shown in Figure 3-15. The module starts at a home page which has options for search in the report list as it can be quite long. Additionally, there is also an option which leads the user to the report module if they wish to report a new bug or feature.

The first search option is the “Search by date” option, where the user is required to submit a starting date and an end date. Once done, clicking the “Search” button will update the report view bellow with reports within the date period. Choosing the same date twice will display all reports submitted from midnight of that day to right before midnight the next day. The second search option is “Search by Software” option, where the user is required to enter a software name into the field. Upon clicking the “Search” button the report view is updated with matching software. Typing in the entire software name is not necessary as the search API will find all of the software that matches the input, even if it is only partial. The view is then ordered alphabetically. If the user submits something other than a software name or leaves the search field empty a warning message is prompted.

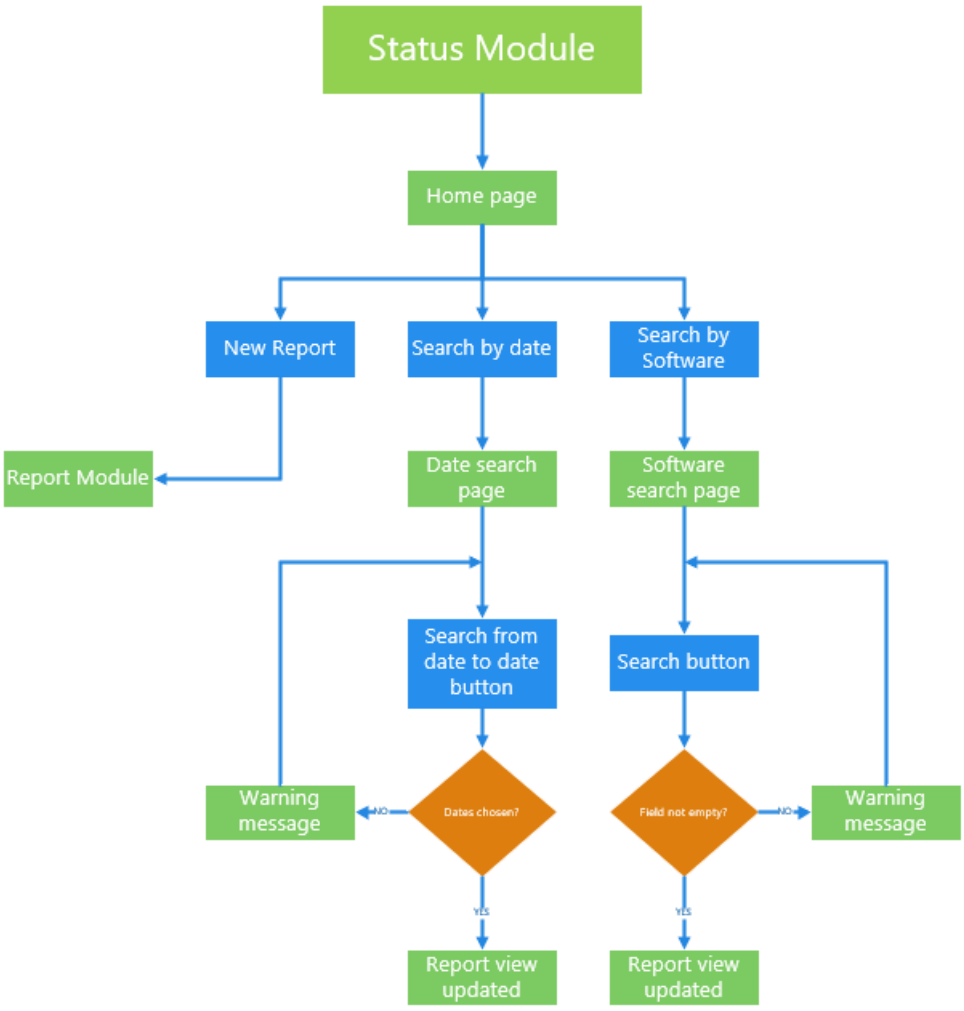


Figure 3-15: "Status" module user task flow

Lastly, the largest module is the “Developer” module. This module is meant to only be accessed by registered users (developers) however the login feature is not yet implemented, and it is a standalone module for the time being.

The current module consists of three manager units, each responsible for managing a specific part of the system. Starting with the report manager, shown in figure Figure 3-16 on the left side, which features three core functions for reports, namely “Add Report” which uses the “Report” module to create new reports, “Report View” which brings the user to a more detailed overview of a specific report, shown in Figure 3-17, and “Delete Report” which removes the selected report from the system entirely. The “Report View” has multiple functions, as illustrated by the Figure 3-17. Here, the user has five sections of additional information and options for setting and modifying the information.

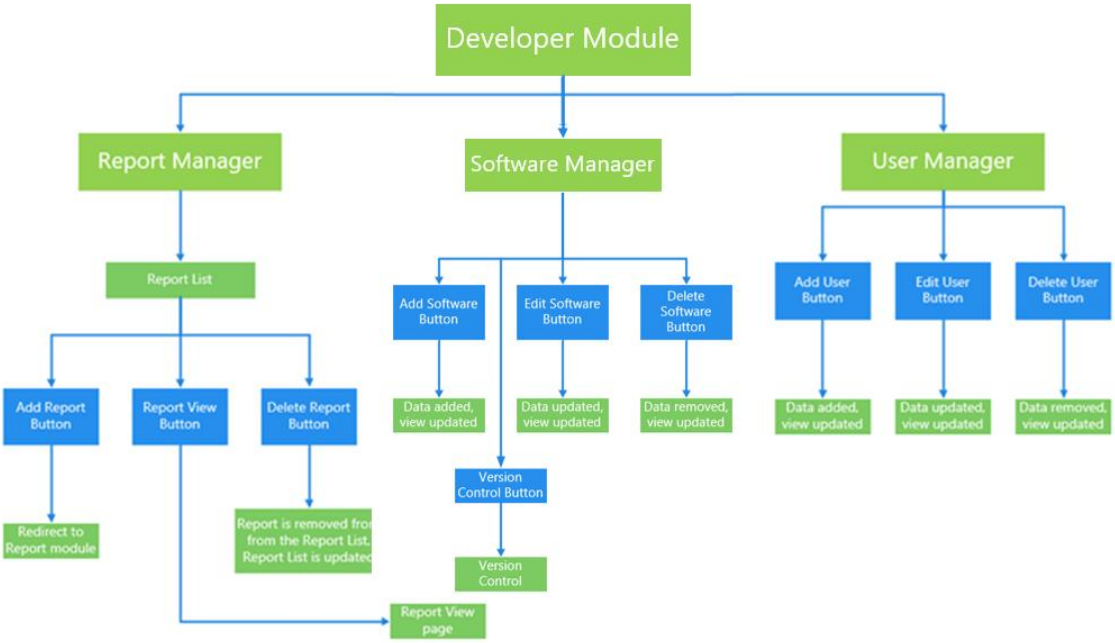


Figure 3-16: "Developer" module user task flow

The “Details” section is the main view of the most important information on the report, such as title, software name and version number, type of the report, status, priority/severity levels and who is assigned to work on the problem. There is also a description section which shows the problem description sent by the user who reported the problem, which also features the function for assigning a responsible person to work on the report. This is done by selecting a name from the dropdown list and clicking “Assign” button. Next, in the “Modify” section the user can set or change the status, priority and severity of the report by selecting an option for each from their dropdown lists and clicking “Save”. The user must select each of the options to successfully update the “Details” section with new values. In the case a single option being updated the remaining two should be selected the same values as they already are before saving the changes.

Lastly, the internal comment section, which consists of the input field and the viewing field. In the input field a developer can add internal commentary by typing text in the text field and clicking the “Add comment” button. The added comment is then displayed in the

commentary field view bellow (“Internal Comments” section) along with all the previously added comments.

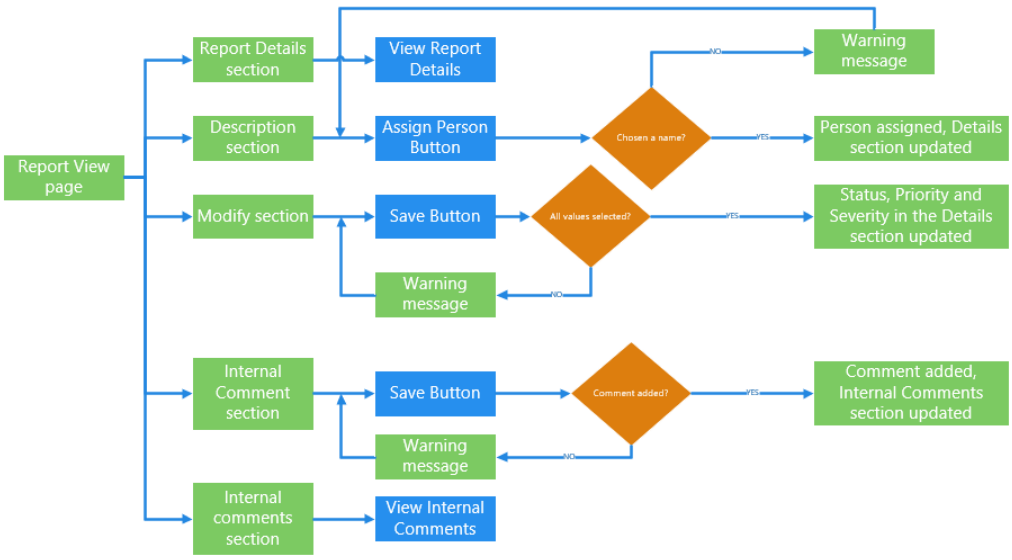


Figure 3-17: Developer page Report View user task flow

Moving away from the report manager to the software manager, the software manager features a simple CRUD functionality for software and software versions. Here, the users can see a list of software currently in the system, with options for adding a new software, editing the name, or deleting an existing software, shown in the middle part of the Figure 3-16. Additionally, an option to add different versions is present in the version sub manager, which is accessed by clicking the “Version Control” button next to a selected software. In the

version control sub manager, the developer can see all the currently registered versions for that software and either add new or remove existing versions, shown in Figure 3-18 bellow.

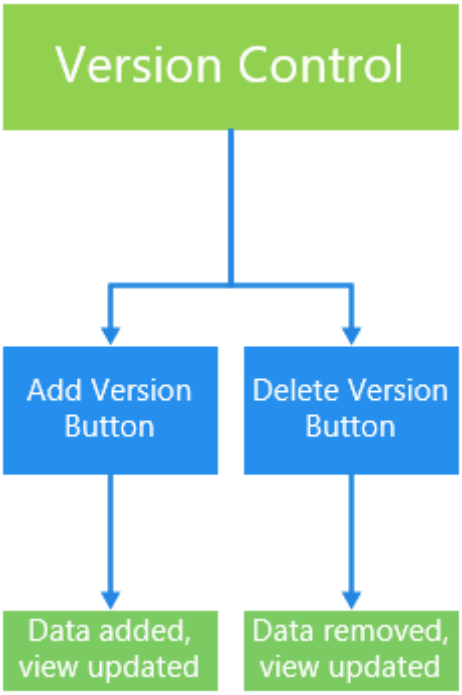


Figure 3-18: Version control sub-manager user task flow

Finally, the user manager, shown on the right side of Figure 3-16, is for management of registered users. Similar to software manager, the user manager also features a list of users with options for creating new, editing or removing existing users.

3.5 Data Specifications

To prevent unwanted interactions with the database, such as custom injection queries or browsing of private data by the users (leaks), all interactions between the database and user or management modules will transact with specified procedures that only allow for a specific interaction (stored procedures). Viewing of multiple table content will be done in a similar fashion by using custom view procedures.

4. Database design

This chapter will present the current database design, a short summary on how and why the current database has been expanded and how it is used in the system.

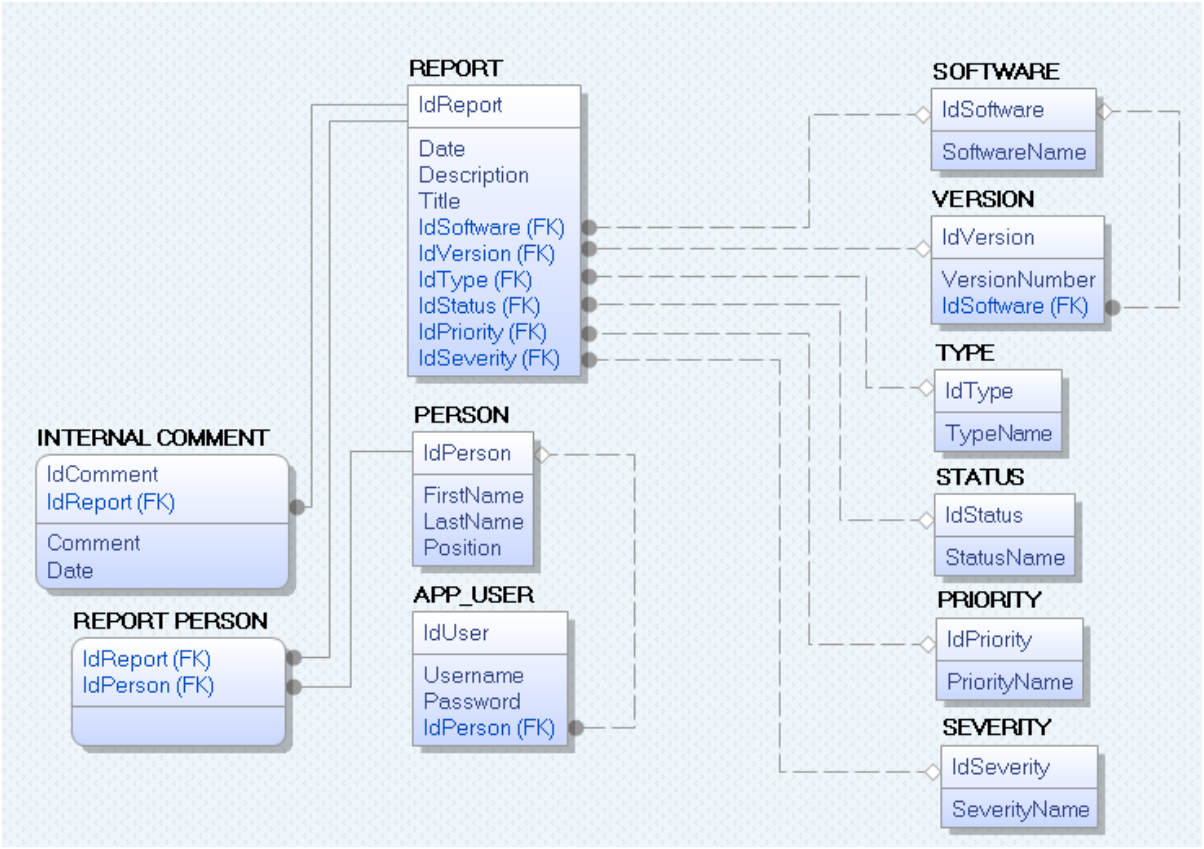


Figure 4-1: Current database table model

Figure 4-1 shows the current database table model used for the system while Figure 4-2 features the original design. The database expansion was implemented due to lack of details in the original design. This is improved by adding additional tables which hold information for software version, priority and severity levels and a table for internal developer commentary. Additionally, some structural changes were necessary, due to flaws in the original design, specifically for connections between REPORT and PERSON tables. The biggest change here is that a table RESPONSIBLE PERSON was dropped completely for being redundant and unnecessary.

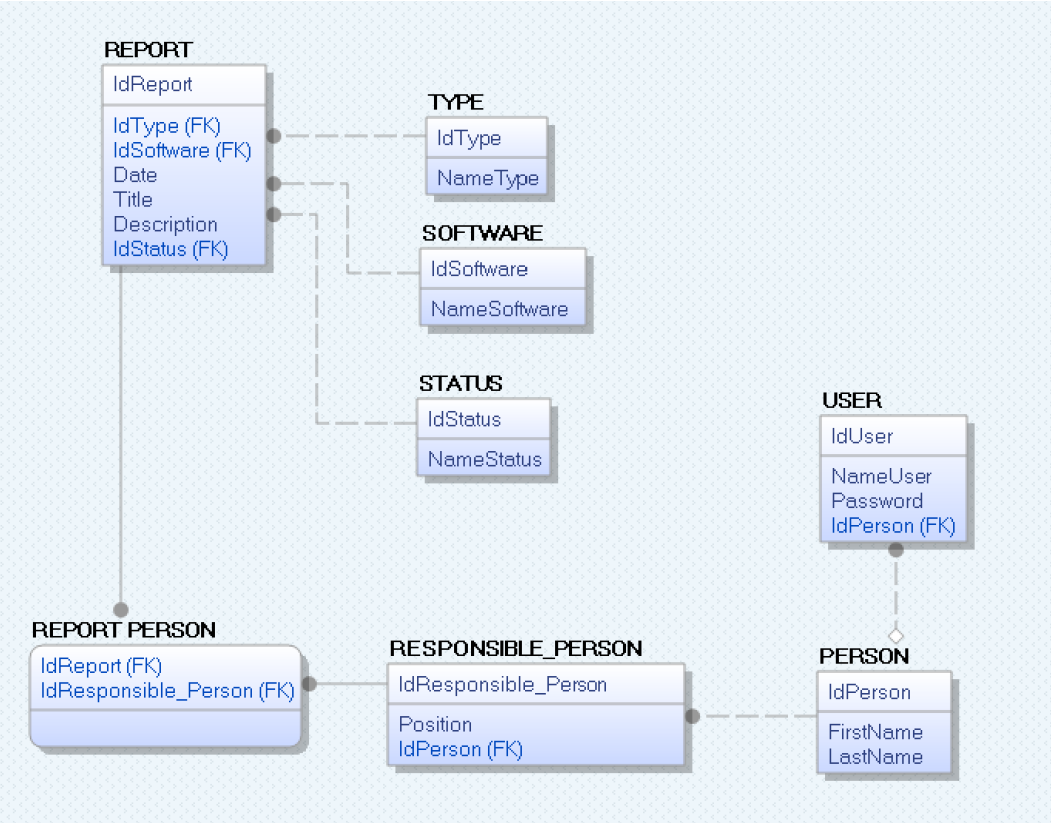


Figure 4-2: Previous database table model

4.1 Database tables

This subchapter focuses more on the functions and relations of each table in the database. Starting with the REPORT, it holds the key information for every report which is submitted by the user. Additionally, the table holds multiple foreign keys to the supplementary tables SOFTWARE, VERSION, TYPE, STATUS, PRIORITY and SEVERITY. Each of these tables serve a single purpose which is to supply the report with additional information, some of which is exclusive for the developers. In the case of SOFTWARE and VERSION tables they have an additional relation between each other, where the VERSION table also contains a foreign key to the SOFTWARE table. This is so that each existing version can be identified for a specific software, which is then done using a clever API that sorts away versions that do not match a selected software foreign key from the VERSION table.

The INTERNAL COMMENT table supplements the REPORT table with developer commentary. Here, the foreign key from the REPORT table also serves as a partial primary key to help identify every comment for a specific registered report. Additionally, this table also contains a date field for when the comment was made as well as the commentary itself.

The last supplementary table for REPORT is the REPORT PERSON table, which serves as an identifier table for “Responsible Person” for each of the reports. This is done by placing a report key and a person key inside REPORT PERSON table which creates a relation between a person and a report without limiting on how many reports a single person can be assigned to.

Lastly the PERSON and APP USER tables. The PERSON table holds information about a registered user in the system while the APP USER table is intended to hold the login credentials for each of the users, however this last table is not in use as the login feature is not implemented.

5. UML

This chapter will present user interactions with the system via use case diagram and sequence diagram. Additionally, this chapter will also contain a class diagram which unlike the other two is focused more on the structure of the system.

5.1 Use Case Diagram

As mentioned in the system architecture, chapter 2.1, the entire system consists of 3 modules: Report module, Status Module and Administrative module. Figure 2-1 shows the general interactions between the modules, the normal end user, and the developer end user.

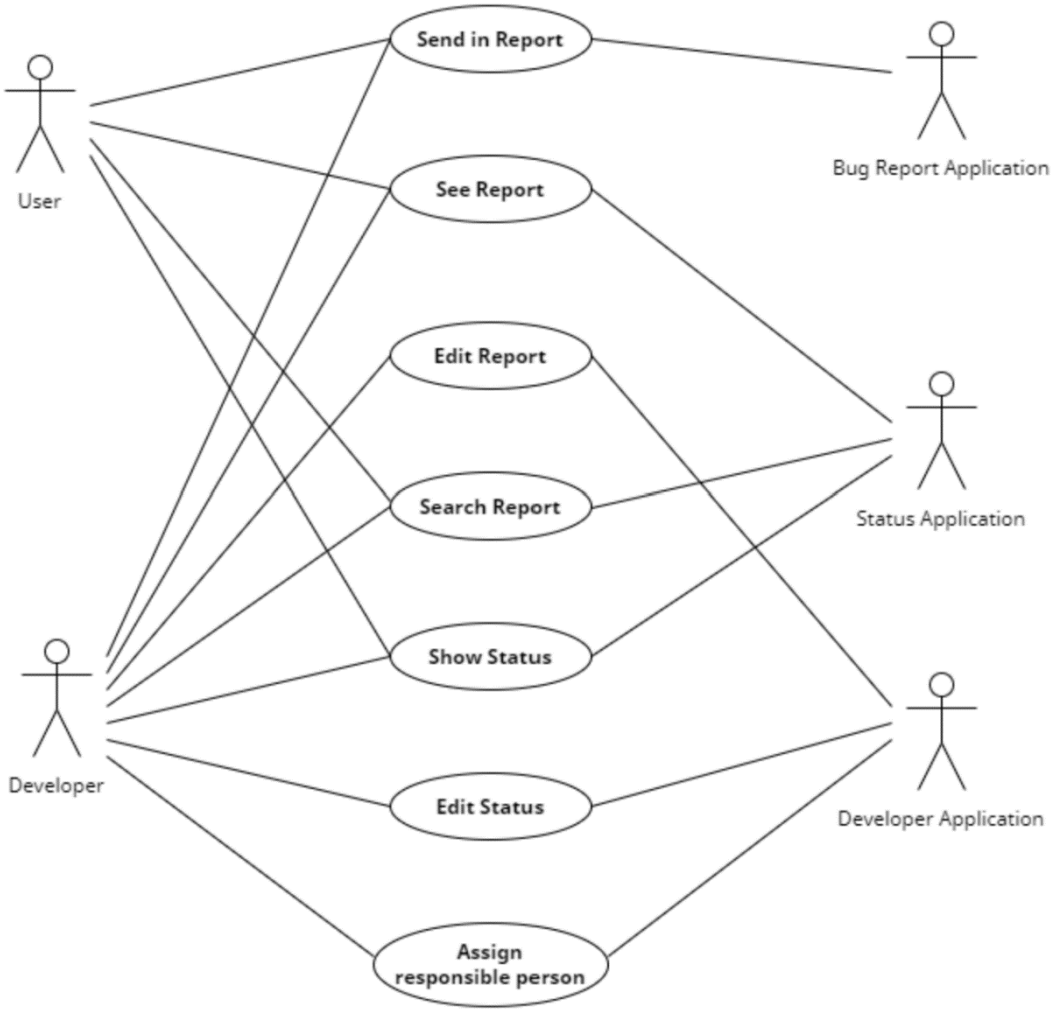


Figure 5-1: Use Case Diagram

5.2 Sequence Diagrams

This subchapter will present a closer and more detailed view of interaction for the BugSpot reporting system. The focus is to showcase how the objects interact with one another and with the user.

Figure 5-2 shows the sequence of interactions taken by a typical user who submits a report, gets a confirmation message that the report is successfully registered and is then able to view the status of said report or search for a specific report which they have submitted previously.

Additionally, the user may visit the “Features” site where they submit a request form for a feature and browse the latest features added. Figure 5-3 show the sequence diagram for administrators and developers.

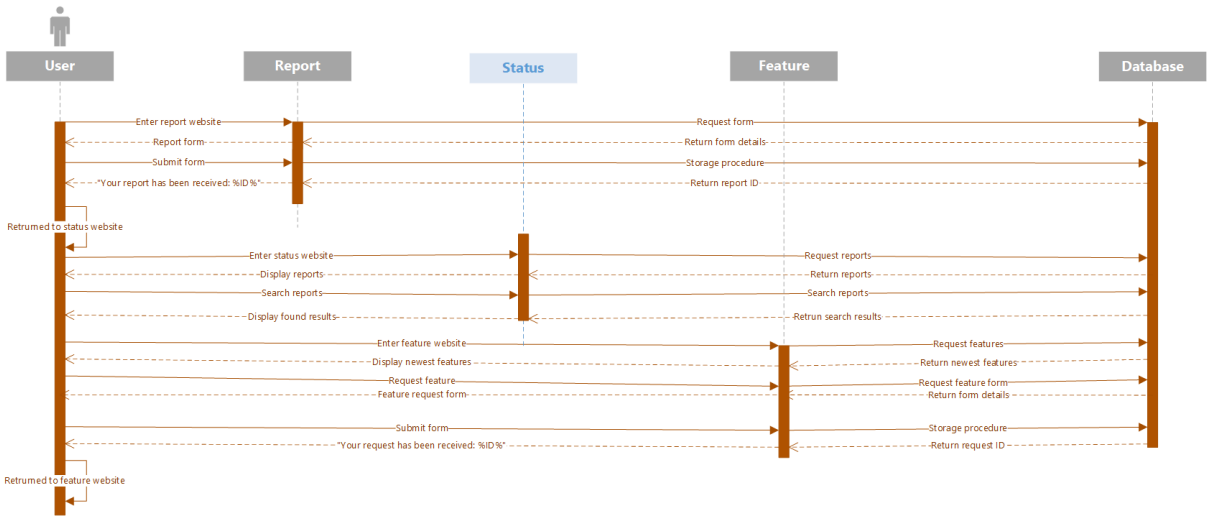


Figure 5-2: Sequence Diagram

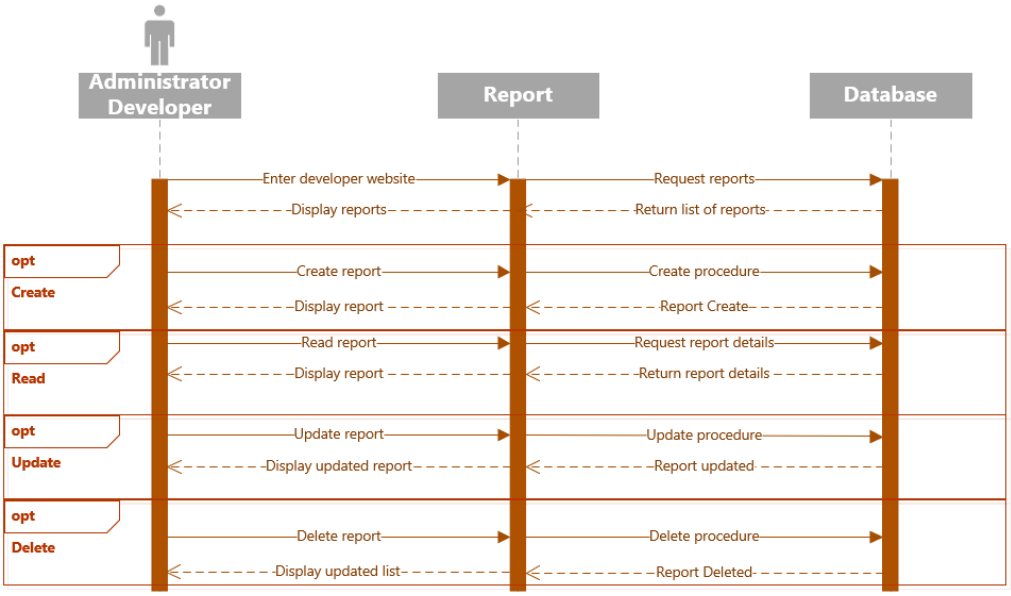


Figure 5-3: Sequence diagram for Administrators and Developers

5.3 Class Diagram

The class diagram presents a detailed view of the program regarding the actual code, specifically object classes, relationships between classes and the various properties and methods they have.

Most of functionality of BugSpot revolves around some type of form to fill in, whether it is a registration, login, or report, all the fill-out forms can be generalized simply as “Form”. Figure 5-4 shows that all the different forms for use of the system are inherited from the abstract class “Form”.

On the left side of Figure 5-4 the user and developer classes are shown to inherit from the class “Login”. This is to differentiate between the two levels of access. An interface requiring all sub classes who inherit from “Login” to also implement a parameter called “UserType”, which defines the access level.

To the right side of Figure 5-4 is the report form inheritance structure. All report forms can be generalized as “Report”. Bug reports and feature requests are very similar to each other and as such only minor differences are noticeable. Additionally, several interfaces are bound to both “Report” sub classes, requiring the parameters for type of report, status of the report and responsible person for the report.

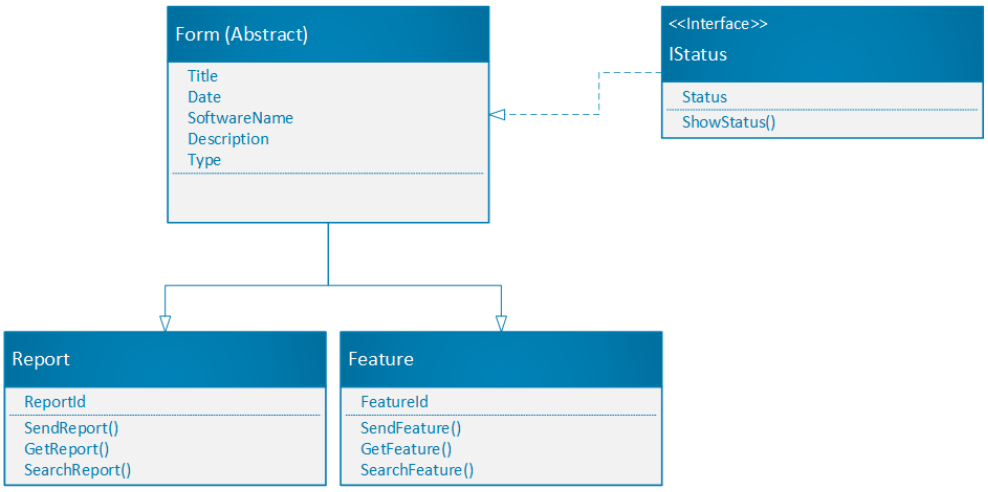


Figure 5-4: Class Diagram

6. Unit Testing

Unit testing is used to test out functions and methods and catch potential bugs and problems before they get released into the wild. A unit test can be created as shown in Figure 6-1 by adding a new project to your current solution.

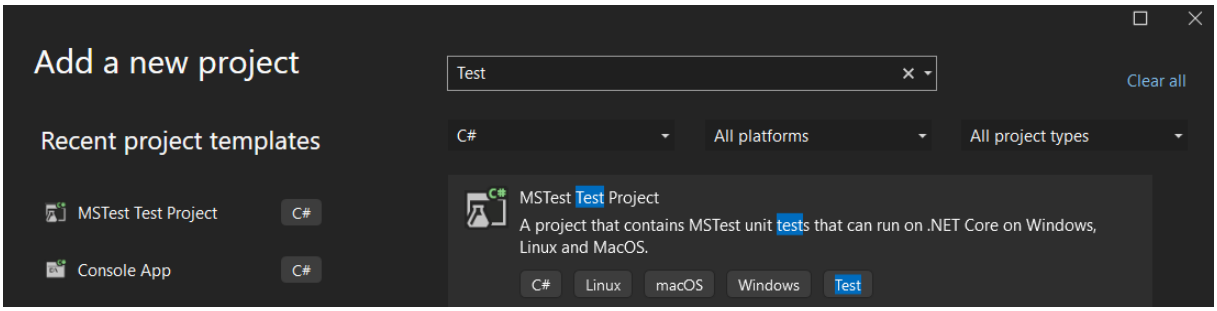


Figure 6-1 Showing the creation of a unit test

Figure 6-2 shows a unit test. Unique to the unit test is the two modifiers [TestClass] and [TestMethod], this are necessary to run a test. The test in the figure shows a TestMath method that checks if the calculation formula is correct and returns the correct result when being used.
The TestDate method checks if the date is of the datatype DateTime and have the correct formatting. If the datatype or formatting is not correct, the test will fail.

```
[TestClass]
0 references
public class UnitTestDev
{
    [TestMethod]
    0 references
    public void TestMath()
    {
        Software software = new Software();
        double x = 10;
        double _x;
        double real = 100;

        _x = software.Math(x);

        Assert.AreEqual(real, _x, 0.001, "Not correct value");
    }
    [TestMethod]
    0 references
    public void TestDate()
    {
        bool result;
        Report report = new Report();
        DateTime date = DateTime.Now;
        string dateString = date.ToString();
        result = report.CheckIfDate(dateString);
        Assert.IsTrue(result);

    }
}
```

Figure 6-2 Showing an example unit test

To run a unit test, click on Test in the header and click run all tests as shown in Figure 6-3. The test will be performed on the methods marked [TestMethod] you have made in the unit test environment.

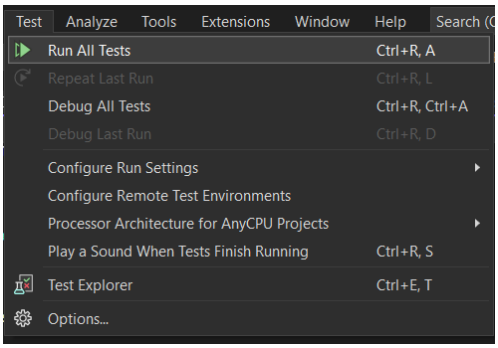


Figure 6-3 Showing how to run a unit test

A successful unit test can be seen in Figure 6-4, In this test we can see what a passed test looks like and see what methods passed and which failed. An example of a failed unit test can be seen in Figure 6-5 and show the red mark with a cross, indicating it failed.

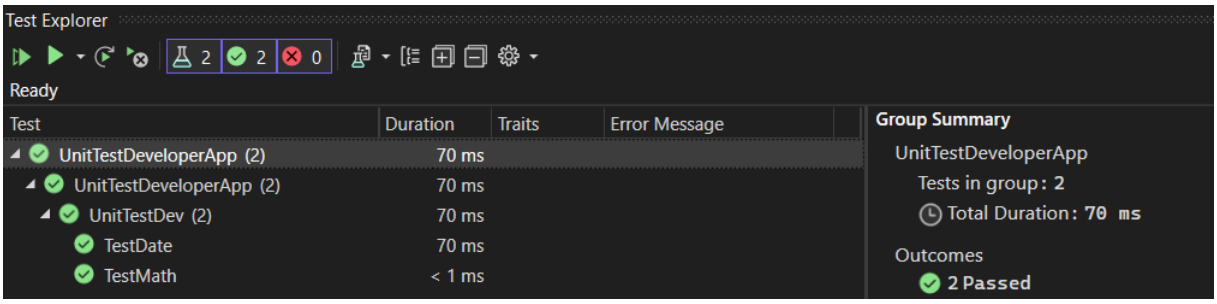


Figure 6-4 Successful unit test

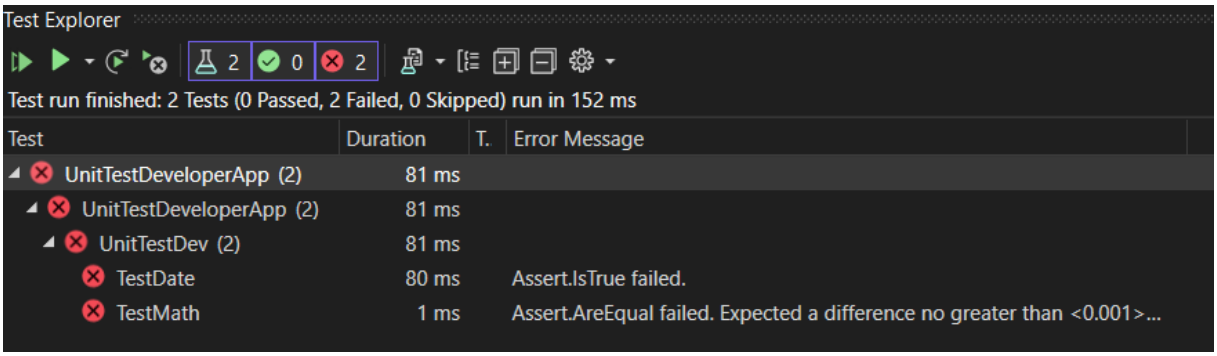


Figure 6-5 Failed unit test

7. Deployment and Maintenance

The installation of BugSpot consists of four parts. The installation demo shows how to use Microsoft Azure to deploy the system using App Containers and how to create a database and deploy BugSpot's SQL script.

The installation process:

- Create App Containers for the web application in Azure
- Publish to the Azure App Container using Visual Studios Publish functionality.
- Create a database and run the SQL script using Microsoft SQL Server Management Studio or similar service.

The installation process is the same for all modules of the system.

Maintenance of BugSpot can be done by republishing the web application using a newer version of the system. Publishing over an existing web application will not affect the data stored in the database.

8. Security

Data security is a very important aspect of any web-based application which collects user input data. This application is no exception as certain laws and regulations, namely the GDPR apply in this case. The core aspects of GDPR are:

- Lawful, fair, and transparent data processing
- Limitation of purpose, data, and storage
- Rights to information, access, and erasure of personal data
- Consent for data collection
- Personal data breaches, data protection, transfers, and privacy

To quickly summarize these points, the company running a data collecting website must ensure that the user is fully aware how their shared data, such as name, phone number, email address, location etc. will be used by the company whether it is for identification, case study or other uses they must be transparent for the user. Data collection must be clearly consented to and documented for both parties, such as a contract like for example a checkbox that must be manually ticked off by the user. The end user also has the right to see what data is collected, demand access and use of their own data and demand to erase parts or all collected data about them. Additionally, the company who collects the data from users must ensure that sensitive data they have collected, such as information which can help identify the user must be secured from potential hacker attacks or leaked due to human error, example being sending contact information of a user to the wrong email.