Software Engineering

SID:2007201

**MOD003263**

**Submitted: December 2021**

Contents

[1.Introduction 3](#_Toc90332215)

[2. Problem Definition Document 4](#_Toc90332216)

[3. Software Design 6](#_Toc90332217)

[3.1 Wireframes 6](#_Toc90332218)

[3.2 Database design 13](#_Toc90332219)

[3.3 Software design 17](#_Toc90332220)

[3.4 Use Case Diagrams 17](#_Toc90332221)

[Class Diagrams 20](#_Toc90332222)

[Sequence diagram 21](#_Toc90332223)

[3.5 GitHub commits 25](#_Toc90332224)

[4. Software Development Documentation 28](#_Toc90332225)

[4.1 user story 1 28](#_Toc90332226)

[4.2 user story 2 34](#_Toc90332227)

[4.3 user story 3 39](#_Toc90332228)

[4.4 user story 4 40](#_Toc90332229)

[4.5 implementation 41](#_Toc90332230)

[5. Testing 48](#_Toc90332231)

[6. Plan of the software Maintenance 58](#_Toc90332232)

[7. Conclusions 59](#_Toc90332233)

[Bibliography 60](#_Toc90332234)

# 1.Introduction

This report shows the work I have done for module: Software Engineering. As a team we used git and github to create a project for our case study. The project was developed using windows form and we also created a database to store user details.

# 2. Problem Definition Document

The project aims are to create a software application using windows form for Musk Process Service, which allowed users to store site inspection details. The software application should allow an inspector to store information about working sites, e.g. number of interventions, comments, completed and action taken. The data that is entered should be stored in a database, as this will allow other inspectors to view previous inspection details. In addition, the overview of the inspection should be stored in a database and be tracked, e.g. total number of interventions made… This is important as Musk Process Service are targeted and want to monitor this data to improve safety of every site. Also, user details should be stored in a database.

System Requirements

The system requirements are:

* Computer running windows 7 and above or smart device like mobile phone and tablet
* Computer hardware, Mouse keyboard to use system/program

Our team used agile methodologies to collaborate and create a good working environment to reach and create system requirements. The diagram that represents functionally for this case study can be found within the user stories that we discussed and created, following these user stories will allow us to create a strong project in which our clients will be satisfied. Furthermore, a database was required for this project as our analysis showed us that the project cannot be fully functionally and appropriate for our clients without a database. We created an EERD diagram to demonstrate and design how the database will look.

In order to proceed with this project we required:

* Lucid chart – to create the diagrams for design.
* Github
* Visual studio 2019
* Microsoft word
* Google docs
* Google chrome
* Microsoft teams

**List of User Stories**

* As a site inspector, I want to login to my account
* As a site inspector I want to register for an account
* As a site inspector, I want to be able to go back and forth between log in form and registration, so I can register then log in to my account
* As a site inspector, I want to see all accounts that are registered to system
* As a site inspector, I want to enter the overview details of the site I have inspected
* As a site inspector, I want to get view all previous overview site inspection details
* As a site inspector, I want to log out and delete all current information that has been entered during my session which I have not saved
* As a site inspector, I want to enter information about site I am inspecting
* As a site inspector, I want to save inspections to a PDF
* As a site inspector, I want to view previous inspections as a PDF

# 3. Software Design

## 3.1 Wireframes

Graphical user interface

Description automatically generated

Figure Wireframe for log in

This diagram shows how the login form should look. This is what the user will first see when they start the program, the login system. As you can see, there are two datafield that are required, username and password. Once the user has entered these details, providing the user exists they can click the login button and they will be sent to the InspectionForm which will be shown below. Also, there is a ‘showPassword’ button this will allow user to see the password as initially the password will be covered. The cancel button allows user to quit the application. The register button will take the user to the registration wireframe.

Graphical user interface

Description automatically generated with medium confidence

Figure Wireframe for registration

This diagram shows how the registration form should look like, we designed it this way as it clear and simple to understand, as according to case study brief, it should be very straight forward. The four datafields are Name, D.O.B, username and password. The ‘back to login’ button will take the user back to the login wireframe. We have made the ‘register’ button large to enable simple registration. User will be able to register for an account once they have clicked this button.

A picture containing table

Description automatically generated

Figure Showing how user will enter data for 'Working standards'

A picture containing whiteboard

Description automatically generated

Figure Wireframe for user entering details about 'Quality'

A picture containing table

Description automatically generated

Figure Wireframe for user entering details about 'Site Rules'

Table

Description automatically generated

Figure Wireframe for user entering details about 'Environmental'

Table

Description automatically generated with low confidence

Figure Wireframe for user entering details about 'Environmental'

Table

Description automatically generated with medium confidence

Figure Wireframe for user entering details about 'Environmental'

Table

Description automatically generated

Figure Wireframe for user entering details about 'Miscellaneous'

Graphical user interface

Description automatically generated with medium confidence

Figure Wireframe for user uploading/confirming images uploaded

This diagram allows user to view all images before uploading to the system.

Graphical user interface, table

Description automatically generated

This wireframe shows what is displayed after figure 10 is complete. The user is able to look at the total interventions of the inspection he has done and add any extra comments. Once he is happy, he can click ‘finish and save’, which will complete the inspection and save the details to a PDF.

Figure Wireframe showing overview after inspection is complete

Table

Description automatically generated

This wireframe allows user to enter information about the overview of the inspection, the data fields are Site, work area, supervisor, completed by, Job description, Date, Inspector and type. The start button will enable them to begin another inspection.

Figure Wireframe for user entering overview details

Description for wireframes

Figure 3-9 all have very similar designs to keep it consistent with the system and user. They are all for taking information about the site inspection. In summary the user will be able to enter details in the blank boxes when required. The user should enter details in the rows next to each subheading (1-5) about whether there was an intervention, any comments, whether task has been completed and what action was taken, if any. Also, there is an upload image button, this is placed underneath the table and allows user to attach any photos as evidence or referencing as client requested. In addition, the next button is placed at the bottom, so the user is able to go to the next page to enter inspection data about each element of the inspection.

Above are wireframes me and my team have made, the reason we made these wireframes was allow our clients, and our team to understand and take a look at the critical structure of the project. Making these wireframes enabled us to understand our client’s vision in more detail, therefore allowing us to satisfy their needs and requirements to the highest of standards. Furthermore, while we were making these wireframes, we sent them to our client (Cristina) to provide us with feedback and their opinions, this allowed us to modify and create wireframes that our clients were happy with. The benefit of making these wireframes is that a change can be made very quickly and efficiently, rather then the user seeing the final project and requesting change to the program, as changing the final program can be difficult and time-consuming. Once our clients were happy with these wireframes, we were able to confidently develop the program to what the clients wanted and expected. We designed the wireframes based on how they will look on a smart device, such as a phone or iPad, however the designs will be very similar on a webpage. As you can see the wireframes for entering data about the inspection look very similar in the sense they will all have ‘Health, safety, quality and environmental site inspection’ as the title, this is in bold and will be the headline for each form.

## 3.2 Database design

Diagram

Description automatically generated

Description

A database is required because the project requires to store data that exists even when the program is not running.

Firstly, table ‘User’ is needed because we must store user details in a external database so that inspectors will not be required to register for an account each time they use the software, as the program use queries to identify each user and check whether they exist. Furthermore, we found out during the meeting with Musk Proccess Service that inspection details must be available to access at any given time when requested by an inspector, therefore, using a database will allow the program to fetch data whenever required. Also, through our requirement analysis we initially decided that table ‘Inspections’ will be stored in a database, however after discussions with our client (Cristina Luca), we agreed this data will be stored in a PDF. User and Inspections have a 0 to many relationship because a user can exist without having made any inspections or having made many inspections. Inspections and User have a 1 to 1 relatioship as one inspection will be linked to only one user. Inspections and inspectionDetails have a 1 to many relationship because one inspection can have many details. InspectionDetails and inspection have a 1..1 relationship because one inspectionDetails can only be linked to one inspection. We decided to create a composition for Inspections and InspectionDetails, this is because if Inspections is deleted, details and the overview of all inspections cannot exist because there will be nothing to store. (informaiton is relied on inspections)

**Database schema**

|  |  |  |  |
| --- | --- | --- | --- |
| User | | | |
| **Attribute Name** | **DataType** | **Allow Nulls** | **Description** |
| User\_ID | Int(5) | NO | Primary key, unique for each user, Auto incrementing |
| User\_Name | Varchar(50) | NO | Stores username (will use when user wishes to log in) |
| User\_DOB | Int(6) | NO | Stores user date of birth |
| User\_Password | Varchar(50) | NO | Stores users password (will use when user wishes to log in) |

|  |  |  |  |
| --- | --- | --- | --- |
| Inspections | | | |
| **Attribute Name** | **DataType** | **Allow Nulls** | **Description** |
| Inspection\_ID | Int(5) | NO | Primary Key, unique for each inspection, Auto incrementing |
| User\_ID | Int(5) | NO | Foreign key, referenced from user table |
| Working\_Standards | Varchar(50) | Yes | Stores working standard data, allow nulls as inspector may not find any problems or issues |
| Site\_Rules | Varchar(50) | Yes | Stores site rule data, allow nulls as inspector may not find any problems or issues |
| Environment | Varchar(50) | Yes | Stores environment data, allow nulls as inspector may not find any problems or issues |
| Protection\_Of\_Individuals | Varchar(50) | Yes | Stores protection of individual data, allow nulls as inspector may not find any problems or issues |
| ToolsCablesAndOtherEquipment | Varchar(50) | Yes | Stores tools cables and other equipment data, allow nulls as inspector may not find any problems or issues |
| Miscellaneous | Varchar(50) | Yes | Stores miscellaneous data, allow nulls as inspector may not find any problems or issues |

|  |  |  |  |
| --- | --- | --- | --- |
| InspectionsDetails | | | |
| **Attribute Name** | **DataType** | **Allow nulls** | **Description** |
| Inspection\_details\_ID | Int(5) | No | Primary Key, unique for each inspection detail, Auto incrementing |
| Inspection\_ID | Int(5) | No | Foreign key, referenced from inspection table |
| Positive\_Intervention | Int(5) | No | Stores number of positive intervention |
| Negative\_intervention | Int(5) | No | Stores number of negative intervention |
| Total\_Intervention | Int(5) | No | Stores total number of intervention, can be calculated |
| Action\_Taken | Varchar(50) | No | Stores whether action was taken |
| Completed | Varchar(50) | No | Store whether inspection is completed |
| Comment | Varchar(50) | No | Stores comments an inspector wants to add |

## 3.3 Software design

### 3.4 Use Case Diagrams

(Anon., n.d.)

Diagram

Description automatically generated

**Description**

|  |  |
| --- | --- |
| **Use Case Name** | Register to program |
| **Actors** | Site inspector |
| **Flow of events** | 1. User clicks make new account button 2. Registration form opens 3. User enters his details 4. User clicks register account 5. User account is registered 6. Message displays confirming creation of user |
| **Alternative flows** | 3a. Error message shows: ‘D.O.B must only contain integers’  3b. Error message shows: ‘Fields must be 50 characters or less’  3c. Error message shows: ‘Password and Password\_Confirmation do not match’  3d. Error message shows: ‘Password must contain minimum 7 characters’  4a. Username already exist, cannot create account  4b. Error message shows: ‘all fields must be entered’  6a. Message does not display confirmation |
| **Entry Conditions** | User enters appropriate user details and does not have a username already. |
| **Exit conditions** | User is successfully registered or cancels registration |

|  |  |
| --- | --- |
| **Use Case Name** | Login to program |
| **Actors** | Site inspector |
| **Flow of events** | 1. User enters login details 2. User clicks login button 3. ‘InspectionForm’ page opens |
| **Alternative flows** | 2a. Error message shows: “Incorrect username or password’  3a. ‘InspectionForm’ does not open  4a. User clicks on ‘Register new user’  5a. User clicks ‘cancel’ button |
| **Entry Conditions** | User exist to the system and enters correct username and password |
| **Exit conditions** | User enters incorrect login details, creates new account or cancel login process |

|  |  |
| --- | --- |
| **Use Case Name** | View/edit inspection Form |
| **Actors** | Site inspector |
| **Flow of events** | 1. Fill in ‘working standard’ details (if applicable) 2. Fill in ‘Quality’ details (if applicable) 3. Fill in ‘Site Rules’ details (if applicable) 4. Fill in ‘Environmental’ details (if applicable) 5. Fill in ‘Protection of individuals’ details (if applicable) 6. Click ‘submit’ button 7. Message shows submission successful 8. Program goes back to original inspection form |
| **Alternative flows** | 1-5a. Modify information entered  6a. Clicks on ‘cancel’ and deletes information entered  8a. Goes back to login page due to inactivity |
| **Entry Conditions** | User logs in successfully |
| **Exit conditions** | User clicks ‘Submit’ button or ‘cancel’ button |

|  |  |
| --- | --- |
| **Use Case Name** | View inspection overview |
| **Actors** | Site inspector |
| **Flow of events** | 1. User clicks ‘view inspection overview’ 2. Inspection PDF opens in new windows 3. Inspection PDF shows overview of all previous inspections |
| **Alternative flows** | 3a. System has no inspections saved |
| **Entry Conditions** | User has valid account and previous inspections have been submitted and saved |
| **Exit conditions** | User views previous inspection details or clicks ‘cancel’ button |

|  |  |
| --- | --- |
| **Use Case Name** | Log out |
| **Actors** | Site inspector |
| **Flow of events** | 1. User clicks log out 2. Message displays ‘confirm log out’ 3. User confirms and program goes back to log in page |
| **Alternative flows** | 3a. User does not confirm and stays logged in |
| **Entry Conditions** | User has logged in |
| **Exit conditions** | User wishes to log out or cancels logging out |

### Class Diagrams

Diagram

Description automatically generated

Figure Class diagram

Figure 13 shows the class diagram that we used when developing our user stories, we looked through each user story and identified which classes would be required. The class diagram contains 5 classes which include 4 front-end classes and 1 back-end class. Inside each of these classes contain the methods required to ensure the user stories are confirmed. For example, I created a ‘Register’ class which includes methods like ‘Register Account’, this was made as user story 2 states ‘As a site inspector I want to register for an account’, Therefore, is it evident this method was necessary to ensure user story 2 is met. This process was done for all user stories. The class ‘DatabaseConnection’ is linked to all other classes because this is the class that will allow the program to communicate with our database, for example the methods in this class include ‘saveToDB’ and loadOverviewData’. These methods are then called upon by other classes within the program. This will save us time as we can reuse the methods each time is it required, rather then typing out the code multiple times. InspectionForm and InspectionFormOverview are linked because the overview is derived from the data in InspectionForm. Login and register are connected because once a user registers an account they will be able to go to the login form and log in using the account in which they have just created. This class diagram allowed me and my team to understand the overall structure that our program will have and require to ensure we meet all user stories successfully.

### Sequence diagram

The sequence diagrams are built during the development of the user stories.

Diagram

Description automatically generated

User Story 1: The diagram above displays the user registering for an account

1. User enters relevent credentials; name, D.O.B (date of birth), Password and confirmation password.
2. RegisterChecker checks whether user input is valid and succificent
3. If user input is valid, program will use SqlQueries method from database class to save userinput into database
4. Register form will output, success message that registation is succesfull
5. If user input is NOT valid, program will output error message and user must re enter details.

A picture containing diagram

Description automatically generated

User story 2: The above diagram shows the user logging into the system

1. User enters username and password into LoginForm
2. LoginChecker will call SqlQueries and check whether the username and password user has entered exists in the database
3. If username and password exist, user will gain access to system
4. If username and password do NOT exist, error message will be displayed and user will not have access to system.

Diagram

Description automatically generated

User story 6: The diagram above shows the action of logging out of the system

1. User logs out of system
2. InspectionForm closes database connection, to ensure database is not manipluated
3. Goodbye message is outputted to user

Diagram

Description automatically generated

User story 5: The diagram above shows user viewing data about previous site inspections

1. User clicks view data
2. siteInspectionForm runs a select query to retrieve database about inspection details
3. SqlQueries successfully retrieves data
4. Inspection Details is displayed to user

A picture containing timeline

Description automatically generated

Figure user entering information about the overview inspection details

User story 7: The diagram above shows the user entering overview details about the inspection that has been made

1. User enters data and fills out the inspection site form that is displayed on screen
2. System checks whether all required data is entered, if they are data will be saved to the database
3. If required data is not filled, user will be prompted to enter again program will repeat until it is filled
4. Once data is saved, user can click ‘view’ to view data that has been entered
5. User clicks exit to leave inspection overview form.

A picture containing diagram

Description automatically generated

User story 8 and 9: The diagram above shows user saving inspection to a PDF, then viewing this data in a PDF.

1. User enters data into Form1 that is saved into AuditFormData
2. User clicks ‘savetoPDF’ button on AuditFormData, which saves to a PDF.
3. If system is unable to save to a PDF, the system will notify the user there was an error
4. User clicks View data as PDF
5. PDF viewer displays data that is stored in a PDF file

### 3.5 GitHub commits

**Personal commits**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

**These are the commits for the whole project**

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

GitHub link: <https://github.com/Nebulastar/SoftEngSiteBrief>

# 4. Software Development Documentation

## 4.1 user story 1

**‘As a site inspector I want to register for an account’**

**Description**:

Graphical user interface, application

Description automatically generated

In the above screenshot, this is what the user is displayed when starting our program, in order to register for an account, user must click the ‘Click here to create one’ button.

Graphical user interface, application

Description automatically generated

In this screenshot shows that when user clicks ‘create account’ button, they are able to access ‘Register’ form. This is where the user is able to enter data (Name, DOB, Username, Password, ConfirmPassword) and create an account.

**Testing:**

User left required field blank

Graphical user interface

Description automatically generated

Figure User leaves required field blank

In this screenshot it shows how if a user does not enter details in all the required fields, the program will not allow you to register and will ask you to re enter with all required fields filled.

Graphical user interface, application

Description automatically generated

Figure Test for password being minimum 8 characters

In this screenshot, it ensures that the user enters a password that has a minimum of 8 characters, this is to improve security. As you can see, the user has entered ‘John123’ which is only 7 characters, therefore the program will not let user register. System will prompt user to try again.

Graphical user interface, application

Description automatically generated

Figure User entered more than 50 characters

In this screenshot you can see the user has entered more than 50 characters in the required fields, I have programmed the system to not allow this because the database that is used only allows to store a maximum of 50 characters (varchar50). Therefore, program will prompt user to try again.

Graphical user interface

Description automatically generated

Figure User enters letters in DOB data field

In this screenshot, the user has entered letters in the ‘DOB’ data field, the database I have used stores the ‘DOB’ as an integer, therefore only integers can be accepted. The program will notify the user that only numbers are allowed in ‘DOB’ and will prompt to try again.

Graphical user interface, application

Description automatically generated

Figure User has entered two password that do not match

In this screenshot, the user has entered two different passwords in ‘Password’ and ‘confirmPassword’, therefore the program has not let the user to register an account as each user can only have one password for an account.

Graphical user interface, application

Description automatically generated

Figure The password cannot be seen and are hidden behind '\*' keys for security reasons

Graphical user interface

Description automatically generated

Figure user has clicked button 'Show password'

In the above screen shots, you can see initially the program hides the password the user enters; however, I have implemented a ‘show password’ button which will show the password on screen. As you can see once the button is clicked, the password is visible to the user. The user can then uncheck the button to hide it again.

Graphical user interface

Description automatically generated

Figure Before user clicks 'Clear' button

Graphical user interface

Description automatically generated

Figure After user has clicked 'clear' button

Graphical user interface, application

Description automatically generated

Figure Shows user registering for an account

In this screenshot, the user has entered in all data fields in an appropriate way, therefore the tests have passed. As this is the case, the user has successfully created an account.



Figure 25 User details has been saved to the database

As you can see, the user details have been saved to the database as ‘userID’ 32.

## user story 2

* ‘As a site inspector, I want to login to my account’ **and** ‘as a site inspector, I want to be able to go back and forth between log in form and registration, so I can register then log in to my account’

Graphical user interface, application

Description automatically generated

Figure Shows the log in form I have implemented

**Testing**

Graphical user interface, application

Description automatically generated

User has entered nothing; therefore, program lets user know that username and password must be entered and will prompt user to try again.

Graphical user interface, application

Description automatically generated

Figure Password is protected by '\*' characters

Graphical user interface

Description automatically generated

Figure user has clicked show password button, password is now visible

Graphical user interface

Description automatically generated

Figure user has clicked 'clear' therefore all input has been cleared and emptied

Graphical user interface

Description automatically generated

Figure user has clicked 'close'

In this screenshot, the user has clicked the ‘close’ button, therefore the program will close. As you can see, a goodbye message has occurred once the user has clicked the close button.

Graphical user interface

Description automatically generated

Figure user has clicked 'create account' and they have been taken too registration form

Graphical user interface, application

Description automatically generated

Figure user has attempted to log in, however the credentials do not exist

In this screenshot, the user attempts to log in with details that do not exist, therefore they are not allowed access. Program prompts them to enter correct details.

Graphical user interface, application

Description automatically generated

As you can see, the user that we registered earlier called ‘John01’, was able to log in. A message box shown confirms that a user can log into their account.

## user story 3

* As a site inspector, I want to see all accounts that are registered to system

Graphical user interface

Description automatically generated

Figure user has logged in clicked on 'user details'

A picture containing shape

Description automatically generated

Figure user has clicked on 'load data'

Graphical user interface, application

Description automatically generated

Figure use has clicked to access database

As you can see, the user is able to access registration details. The data includes userID, name, DOB, name, username and password, therefore the site inspector is able to view all accounts that are registered to the system.

## user story 4

* Graphical user interface, text

  Description automatically generatedAs a site inspector, I want to get view all previous overview site inspection details

Figure shows GUI for entering and viewing overview of all inspections

Graphical user interface

Description automatically generated with medium confidence

Figure User has clicked 'load overview' and can see overview for previous inspections (dummy data is shown for testing purposes)

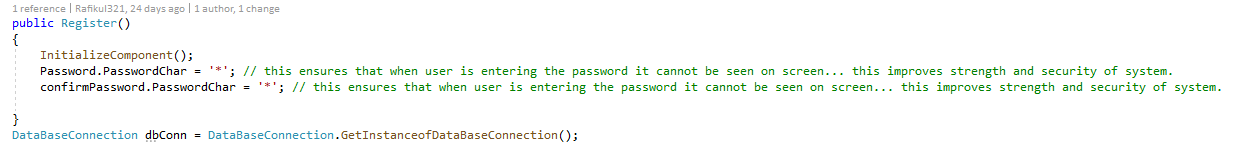
As you can see, the user is able to click ‘load overview for all inspections’ and consequently it is loaded in the DataGrid view for the user to view, therefore user can view all previous overview site inspections.

### 4.5 implementation

­­Code

These are screenshots of the implementations I have solely and personally contributed for this project.

­**Registration Form:**



This shows how I changed the password to not be displayed and be covered by ‘\*’ characters.

Text, application

Description automatically generated

This code checks whether if the user has left required data fields blank, if so, an error message is shown. Also checks if password and password confirmation match up, password has a minimum of 7 characters and username and password are above 51 characters. Also, I use INSERT query to save the data fields into ‘person’ database. In addition, if password does not match the requirements, I clear the text using “” and automatically put the keyboard onto username using ‘focus’.

Text

Description automatically generated

Figure When user wishes to go to login form, this switches the form and hides register form

Text

Description automatically generated

Figure This allows user to check and uncheck to hide password, if checked password will be displayed in English, if not checked password will be hidden by '\*' characters

A picture containing chart

Description automatically generated

Figure This occurs when user clicks clear button, all text fields are set to empty

**Login form**

A picture containing graphical user interface

Description automatically generated

Figure sets password to be hidden behind '\*' characters

Text

Description automatically generated with medium confidence

Figure Closes form and project when user clicks close button

Text

Description automatically generated with medium confidence

Figure Opens register form

Graphical user interface, text, application, email

Description automatically generated

SELECT query is used to find username and password and uses ‘getDataSet’ method from Database class. This checks whether user exists in the ‘Person’ database. Also checks if user has left required fields blank. If incorrect details are entered, username and password gets emptied.

Graphical user interface, text

Description automatically generated with medium confidence

Figure Empties text box

Graphical user interface, text, application

Description automatically generated

Figure This allows user to check and uncheck to hide password, if checked password will be displayed in English, if not checked password will be hidden by '\*' characters

**DBconnection**

Graphical user interface, text, application, email

Description automatically generated

Created attributes and a database connection string. Also initialise the DatabaseConnectionString.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

This method was created to display the database and also used to check in login form to check whether user exists.

Graphical user interface, text, application

Description automatically generated

This method was created to save data into the database. It expects four parameters, name, dob, username and password. Adds the parameters to the database.

**userDetails**

Graphical user interface, text, application, email

Description automatically generated

Allows DataB to open

**DataB**

Graphical user interface, text, application

Description automatically generated

I used getDataSet method from DBconnection. SELECT query is used to display ‘Person’ database

**InspectionForm**

Graphical user interface, application

Description automatically generated

Figure Opens userDetails form

**SiteInspectionOverview**

Text

Description automatically generated

Figure I used getDataSet method from DBconnection. SELECT query is used to display ‘SiteInspection’ database

# Testing

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test No.** | **Test (Description)** | **Class** | **Expected Result** | **Actual Result** | **Pass/Fail** | **Comments (if applicable)** |
| 1 | Attempt to create account with no ‘Name’ entered | RegisterForm | Program displays required fields must be entered | Program displays required fields must be entered | Pass |  |
| 2 | Attempt to create account with no ‘DOB’ entered | RegisterForm | Program displays required fields must be entered | Program displays required fields must be entered | Pass |  |
| 3 | Attempt to create account with no ‘Username’ entered | RegisterForm | Program displays required fields must be entered | Program displays required fields must be entered | Pass |  |
| 4 | Attempt to create account with no ‘Password’ entered | RegisterForm | Program displays required fields must be entered | Program displays required fields must be entered | Pass |  |
| 5 | Attempt to create account with no ‘Confirm Password’ entered | RegisterForm | Program displays required fields must be entered | Program displays required fields must be entered | Pass |  |
| 7 | Attempt to create account while entering >50 characters in ‘username’ | RegisterForm | Program displays ‘username and password do not meet requirements’ | Program displays ‘username and password do not meet requirements’ | Pass |  |
| 8 | Attempt to create account while entering >50 characters in ‘Name’ | RegisterForm | Program displays ‘Name’ cannot contain more than 50 characters | Program displays ‘Please only enter numbers in ‘DOB’ | Fail | This is a fail as even though the program does not allow more than 50 characters, the wrong error message is shown. This will be fixed in next scheduled update to prevent confusion for user |
| 9 | Attempt to create account while entering >50 characters in ‘Password’ | RegisterForm | Program displays ‘username and password do not meet requirements’ | Program displays ‘username and password do not meet requirements’ | Pass |  |
| 10 | Attempt to create account while entering a password with less than or equal to 7 characters | RegisterForm | Program displays ‘username and password do not meet requirements’ | Program displays ‘username and password do not meet requirements’ | Pass |  |
| 11 | Attemping to to create account while entering different inputs in ‘password’ and Confirm password’ | RegisterForm | Program displays ‘username and password do not meet requirements’ | Program displays ‘username and password do not meet requirements’ | Pass | Preferably next change we will make error message more precise so user knows precisely what is wrong. |
| 12 | ‘Show password’ is checked | RegisterForm | Password is displayed and shown on screen | Password is displayed and shown on screen | Pass |  |
| 13 | ‘Show password’ is unchecked | RegisterForm | Password is hidden and replaced with ‘\*’ on screen | Password is hidden and replaced with ‘\*’ on screen | Pass |  |
| 14 | ‘Clear’ button is clicked | RegisterForm | All textboxes with values will be emptied | All textboxes with values will be emptied | Pass |  |
| 15 | ‘Go back to login page’ button is clicked | RegisterForm | LoginForm will open and be shown | LoginForm will open and be shown | Pass |  |
| 16 | Attempt to register with appropriate inputs and details | RegisterForm | Program displays ‘successful registration’ | Program displays ‘successful registration’ | Pass |  |
| 17 | When registration is complete, ‘Name’ is saved to database | RegisterForm | ‘Name’ is saved to ‘person’ database | ‘Name’ is saved to ‘person’ database | Pass |  |
| 18 | When registration is complete, ‘DOB’ is saved to database | RegisterForm | ‘DOB is saved to ‘person’ database | ‘DOB is saved to ‘person’ database | Pass |  |
| 19 | When registration is complete, ‘username’ is saved to database | RegisterForm | ‘username’ is saved to ‘person’ database | ‘username’ is saved to ‘person’ database | Pass |  |
| 20 | When registration is complete, ‘Password’ is saved to database | RegisterForm | ‘Password’ is saved to ‘person’ database | ‘Password’ is saved to ‘person’ database | Pass |  |
| 21 | When registration is complete, an ‘userID’ is assigned to the user as a Primary key | RegisterForm | User is assigned unique ‘userID’ | User is assigned unique ‘userID’ | Pass |  |
| 22 | Attemping to login with wrong username but correct password | loginForm | Program displays ‘incorrect login details’ | Program displays ‘incorrect login details’ | Pass |  |
| 23 | Attemping to login with correct username but incorrect password | loginForm | Program displays ‘incorrect login details’ | Program displays ‘incorrect login details’ | Pass |  |
| 24 | ‘Show password’ is checked | loginForm | Password is displayed and shown on screen | Password is displayed and shown on screen | Pass |  |
| 25 | ‘Show password’ is unchecked | loginForm | Password is hidden and replaced with ‘\*’ on screen | Password is hidden and replaced with ‘\*’ on screen | Pass |  |
| 26 | Attempting to login while leaving ‘username’ blank | loginForm | Program displays error message ‘ All required fields must be entered’ | Program displays error message ‘ All required fields must be entered’ | Pass |  |
| 27 | Attempting to login while leaving ‘password’ blank | loginForm | Program displays error message ‘ All required fields must be entered’ | Program displays error message ‘ All required fields must be entered’ | Pass |  |
| 28 | ‘Clear’ button is clicked | loginForm | All textboxes with values will be emptied | All textboxes with values will be emptied | Pass |  |
| 29 | ‘Create account’ button is clicked | loginForm | RegisterForm will open and be displayed | RegisterForm will open and be displayed | Pass |  |
| 30 | ‘Close’ button is clicked | loginForm | Program will close and exit | Program will close and end | Pass |  |
| 31 | Attempting to login with correct username and password | loginForm | Program will accept login details and give access to site inspection | Program will accept login details and give access to site inspection | Pass |  |
| 32 | Attempting to submit data while all data fields are empty | SiteInspectionOverview | Message will display ‘No data was entered’ | Program does not realise nothing has been entered and saved empty spaces to database | Fail | My colleague will work on this to fix this bug on next update |
| 33 | Entering letters in ‘date’ | SiteInspectionOverview | Message will display ‘only integers allowed’ | Program crashes | Fail | Exception is thrown as database is can only store int for ‘date’ A try, catch will be used to fix. |
| 34 | When overview is complete, ‘Site’ is saved to database | SiteInspectionOverview | ‘site’ is saved to ‘siteInspection’ database | ‘site’ is saved to ‘siteInspection’ database | Pass |  |
| 35 | When overview is complete, ‘CompletedBy’ is saved to database | SiteInspectionOverview | ‘completedBy’ is saved to ‘siteInspection’ database | ‘completedBy’ is saved to ‘siteInspection’ database | Pass |  |
| 36 | When overview is complete, ‘Date’ is saved to database | SiteInspectionOverview | ‘Date’ is saved to ‘siteInspection’ database | ‘Date’ is saved to ‘siteInspection’ database | Pass |  |
| 37 | When overview is complete, ‘workArea’ is saved to database | SiteInspectionOverview | ‘workArea’ is saved to ‘siteInspection’ database | ‘workArea’ is saved to ‘siteInspection’ database | Pass |  |
| 38 | When overview is complete, ‘Supervisor’ is saved to database | SiteInspectionOverview | ‘supervisor’ is saved to ‘siteInspection’ database | ‘supervisor’ is saved to ‘siteInspection’ database | Pass |  |
| 39 | When overview is complete, ‘Inspector’ is saved to database | SiteInspectionOverview | ‘inspector’ is saved to ‘siteInspection’ database | ‘inspector’ is saved to ‘siteInspection’ database | Pass |  |
| 40 | When overview is complete, ‘Inspector’ is saved to database | SiteInspectionOverview | ‘type’ is saved to ‘siteInspection’ database | ‘type’ is saved to ‘siteInspection’ database | Pass |  |
| 41 | Clicking ‘load Data’ | SiteInspectionOverview | Database for ‘siteinspection’ is displayed | Database for ‘siteinspection’ is displayed | Pass |  |
| 42 | Clicking ‘Submit’ | SiteInspectionOverview | Program displays ‘data successfully saved’ | Program displays ‘data successfully saved’ | Pass |  |
| 43 | Clicking ‘nextForm’ | SiteInspectionOverview | Program displays inspection form | Program displays inspection form | Pass |  |
| 44 | Clicking ‘working standards’ | siteInspection | Program displays subheading of checks | Program displays subheading of checks | Pass |  |
| 45 | Clicking ‘quality’ | siteInspection | Program displays subheading of checks | Program displays subheading of checks | Pass |  |
| 46 | Clicking ‘Site Rules’ | siteInspection | Program displays subheading of checks | Program displays subheading of checks | Pass |  |
| 47 | Clicking ‘Environmental’ | siteInspection | Program displays subheading of checks | Program displays subheading of checks | Pass |  |
| 48 | Clicking ‘Protection of individuals’ | siteInspection | Program displays subheading of checks | Program displays subheading of checks | Pass |  |
| 49 | Clicking ‘Tools cables and other equipment’ | siteInspection | Program displays subheading of checks | Program displays subheading of checks | Pass |  |
| 50 | Clicking ‘Miscellaneous’ | siteInspection | Program displays subheading of checks | Program displays subheading of checks | Pass |  |
| 51 | Entering data in ‘working standards’ and clicking ‘save to pdf’ | siteInspection | Data is saved to PDF and PDF is displayed | Data is saved to PDF and PDF is displayed | Pass |  |
| 52 | Entering data in ‘Quality’ and clicking ‘save to pdf’ | siteInspection | Data is saved to PDF and PDF is displayed | Data is saved to PDF and PDF is displayed | Pass |  |
| 53 | Entering data in ‘Site Rules’ and clicking ‘save to pdf’ | siteInspection | Data is saved to PDF and PDF is displayed | Data is saved to PDF and PDF is displayed | Pass |  |
| 54 | Entering data in ‘environmental’ and clicking ‘save to pdf’ | siteInspection | Data is saved to PDF and PDF is displayed | Data is saved to PDF and PDF is displayed | Pass |  |
| 55 | Entering data in ‘protection of individuals’ and clicking ‘save to pdf’ | siteInspection | Data is saved to PDF and PDF is displayed | Data is not saved to PDF | Fail |  |
| 56 | Entering data in ‘Tools cable and other equipment’ and clicking ‘save to pdf’ | siteInspection | Data is saved to PDF and PDF is displayed | Data is not saved to PDF | Fail |  |
| 57 | Entering data in ‘Miscellaneous’ and clicking ‘save to pdf’ | siteInspection | Data is saved to PDF and PDF is displayed | Data is not saved to PDF | Fail |  |
| 58 | Entering data in ‘working standards’ and clicking ‘save to text file’ | siteInspection | Data is saved to a text file | Data is saved to a text file | Pass |  |
| 59 | Entering data in ‘Quality’ and clicking ‘save to text file’ | siteInspection | Data is saved to a text file | Data is saved to a text file | Pass |  |
| 60 | Entering data in ‘Site Rules’ and clicking ‘save to text file | siteInspection | Data is saved to a text file | Data is saved to a text file | Pass |  |
| 61 | Entering data in ‘environmental’ and clicking ‘save to text file | siteInspection | Data is saved to a text file | Data is saved to a text file | Pass |  |
| 62 | Entering data in ‘protection of individuals’ and clicking ‘save to text file | siteInspection | Data is saved to a text file | Data is not saved to a text file | Fail |  |
| 63 | Entering data in ‘Tools cable and other equipment’ and clicking ‘save to text file | siteInspection | Data is saved to a text file | Data is not saved to a text file | Fail |  |
| 64 | Entering data in ‘Miscellaneous’ and clicking ‘save to text file | siteInspection | Data is saved to a text file | Data is not saved to a text file | Fail |  |
| 65 | Clicking ‘exit’ button | siteInspection | Program closes and ends | Program closes and ends | Pass |  |
| 66 | Clicking ‘Access use details’ | siteInspection | Goes to userDetails form | Goes to userDetails form | Pass |  |
| 67 | Not entering any data and clicking ‘save to PDF’ | siteInspection | Program displays ‘no information about inspection has been entered’ | Program opens PDF and shows inspection PDF with no data inserted | Fail | This is not a true fail, but a messagebox displaying no information has been entered would be more suitable |
| 68 | Entering data under ‘working standards’ but not ‘quality’ and clicking save to PDF | siteInspection | Program saves ‘working standard’ details to PDF but not ‘quality’ details | Program saves ‘working standard’ details to PDF but not ‘quality’ details | Pass |  |
| 69 | Entering numbers in data fields and clicking save to PDF | siteInspection | Program saves user input to PDF | Program saves user input to PDF | Pass |  |
| 70 | Entering data but not clicking save to PDF | siteInspection | Program does not save and data is lost when form refreshes | Program does not save and data is lost when form refreshes | Pass |  |
| 71 | Entering data but not clicking save to text file | siteInspection | Program does not save and data is lost when form refreshes | Program does not save and data is lost when form refreshes | Pass |  |
| 72 | Clicking ‘access database’ | UserDetails | Program opens ‘DataB’ form | Program opens ‘DataB’ form | Pass |  |
| 73 | Clicking ‘load data’ | UserDetails | Program displays ‘Person’ database | Program displays ‘Person’ database | Pass |  |

# Plan of the software Maintenance

Our program meets all the requirements that we listed, apart from saving all inspection to a PDF, as shown by the testing there are a few sections that do not save to a PDF. This part of the program was assigned to a team member that had personal issues, therefore could not complete this task and has now been assigned to other team members, if we had more time available this will be fixed. However, the rest of the program has met the user requirements but from the testing we have done, we have identified minor bugs that are being fixed now/need to be fixed. We have created a list of ways our project could be improved below:

* The database that stores username and password is currently not encrypted, this can be a huge problem because if we have a security breach, the attackers will be able to easily see our users’ personal details. This will mean we have broken many GDPR rules and will result in large fines being awarded to our us and our clients. Therefore, we would like to encrypt all sensitive data within our database. This could be done by using MD5, this hashing algorithm will protect our users. (Technopedia, n.d.)
* When the user enters letters in the ‘Date’ section in ‘InspectionOverview’ the program crashes, therefore we would like to fix this bug to ensure our program is able to display an error message to the user, rather than crashing.
* We would like to create an admin role, where admin would be able to delete and add more users to the program.
* Our program has strong validation for the registration, but we would the program to check whether the ‘username’ that is entered in the register form already exist, if so notify the user that a different username should be chosen. Currently (userID is Primary key)
* As we used windows form to create this program, it will only work on a windows computer, therefore we would like this program to be available for Mac OS, IOS and android.
* In the login form, we would like to add a ‘remember me’ button which allows users to stay logged in for a set period. This will ensure users don’t have to constantly sign in every so often.
* In the login form to improve security, we would like to add two factor authentication, this will allow the program to be more secure, this verification can be done through email or phone.
* Allow users to login into their accounts using biometrics, such as fingerprint or faceID

These improvements will allow our program to be even better and more secure for our clients.

# Conclusions

Being a part of this module has improved many skills, such as working as a team and communication. Furthermore, this module has taught me how to use git and GitHub as this was my first time using GitHub. When I read the module definition form in the first week, I was daunted as I have not programmed in C# before and have not used windows form, however after this module I feel confident using this language. My group was good to work with, even though one team member did not contribute, we were able to still complete the project and manage deadlines. This experience definitely allows me to develop to agile methodologies. Even though we could not meet up as a team due to Covid-19, we were able to communicate through texts using Microsoft teams and discord. I am very proud of myself and my teams contributions to this project.

# Bibliography

Anon., n.d. *Use Case Diagrams/Description.* [Online]   
Available at: https://computersciencesource.wordpress.com/2009/11/22/year-2-software-engineering-use-case-diagrams-descriptions/  
[Accessed Tuesday November 2021].

Luca, C., 2021. *Canvas Lecture Videos.* [Online]   
Available at: https://canvas.anglia.ac.uk/courses/22538/pages/week-05-database?module\_item\_id=1264202  
[Accessed Monday December 2021].

Technopedia, n.d. *What is MD5 hash?.* [Online]   
Available at: https://www.techopedia.com/definition/4022/md5  
[Accessed Monday December 2021].