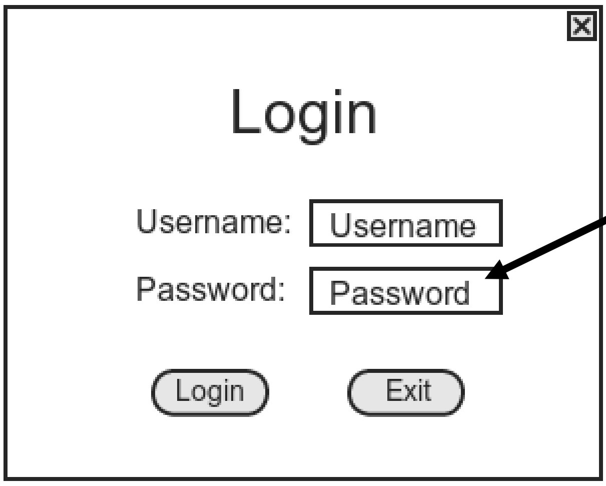
Design

Each room can be dragged around when editing is enabled

Login Form



The password text box has a password character to improve security

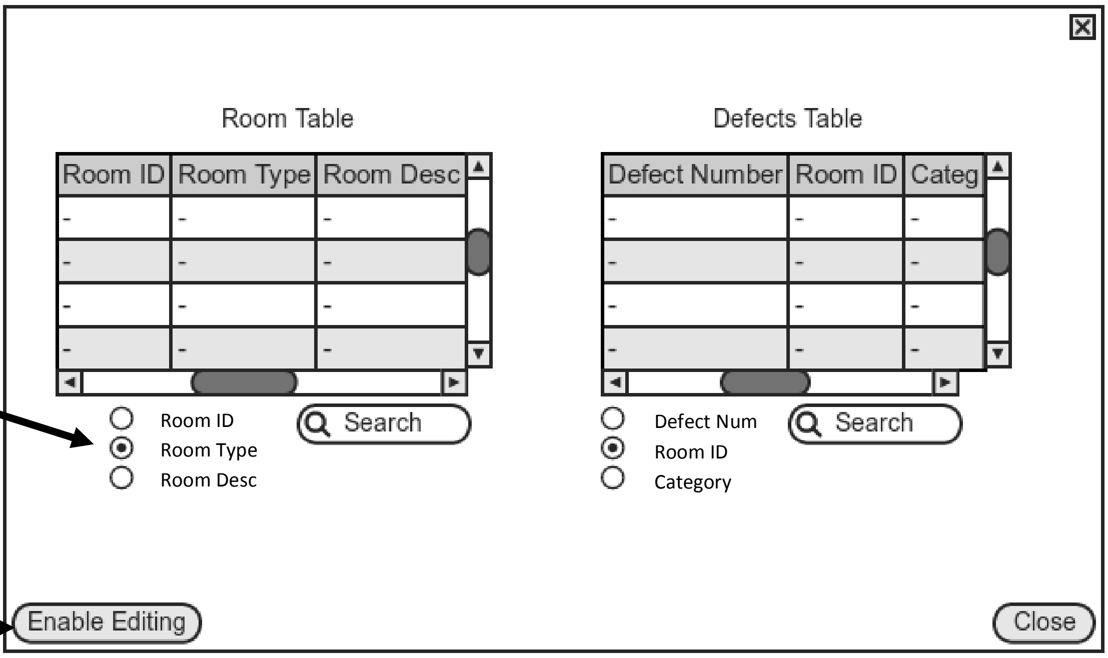
The login button takes a regular user to the Building Plan Form and an admin user to the Admin Menu Form

The exit button closes the product

Tables View Form

When the user selects a defect, the related room is selected on the room table. When a room is selected in the room table, the first defect for that room on the defects table is selected

The two tables will be relational and linked by the "Room ID" field. The Room ID field is the primary key for the room table and the foreign key for the defects table

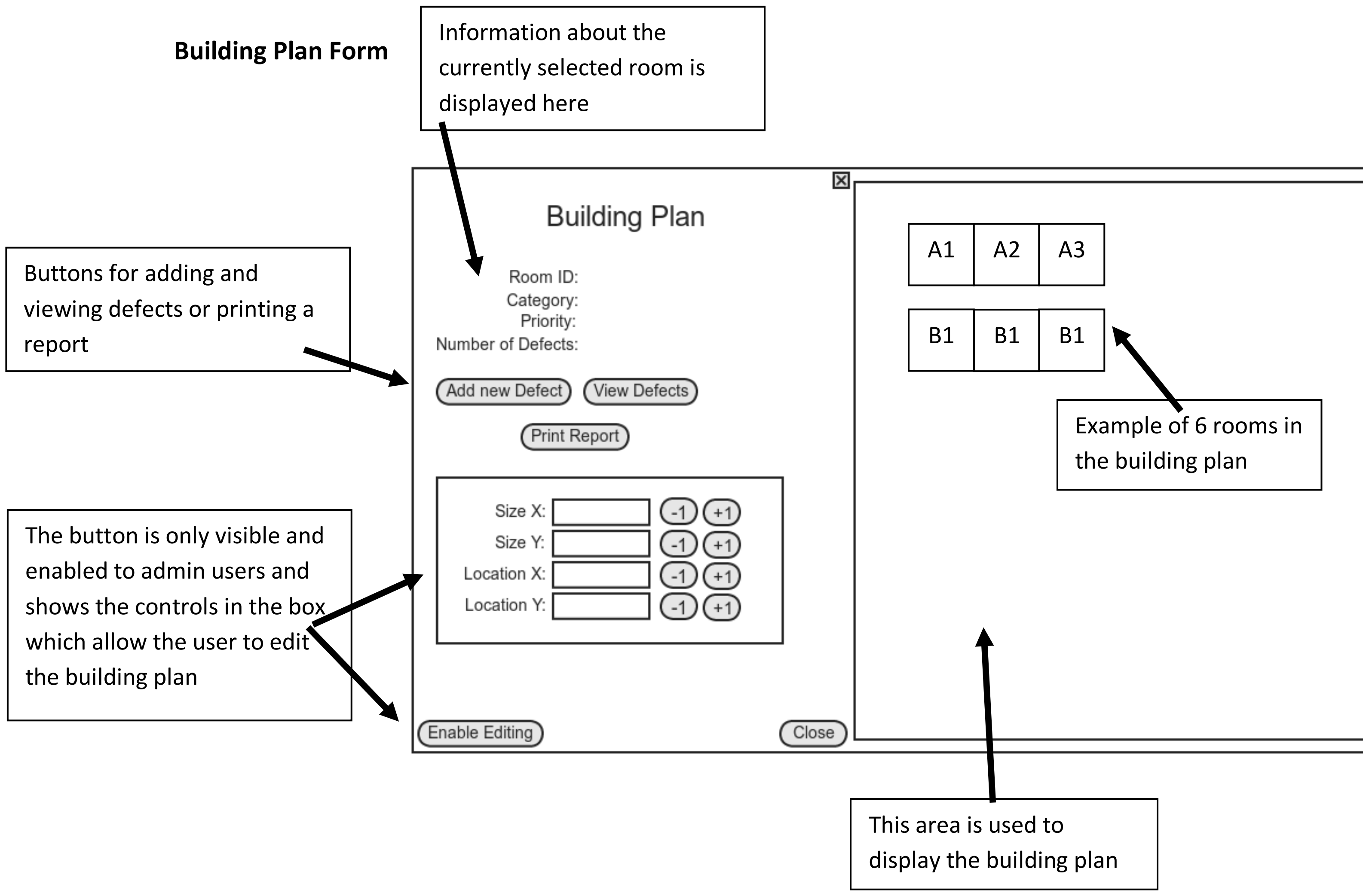


Controls for sorting and filtering

This button is only visible and enabled when the user is on an admin account

Filtering is applied as the user enters information, removing the need for an extra button

Building Plan Form


Information about the currently selected room is displayed here

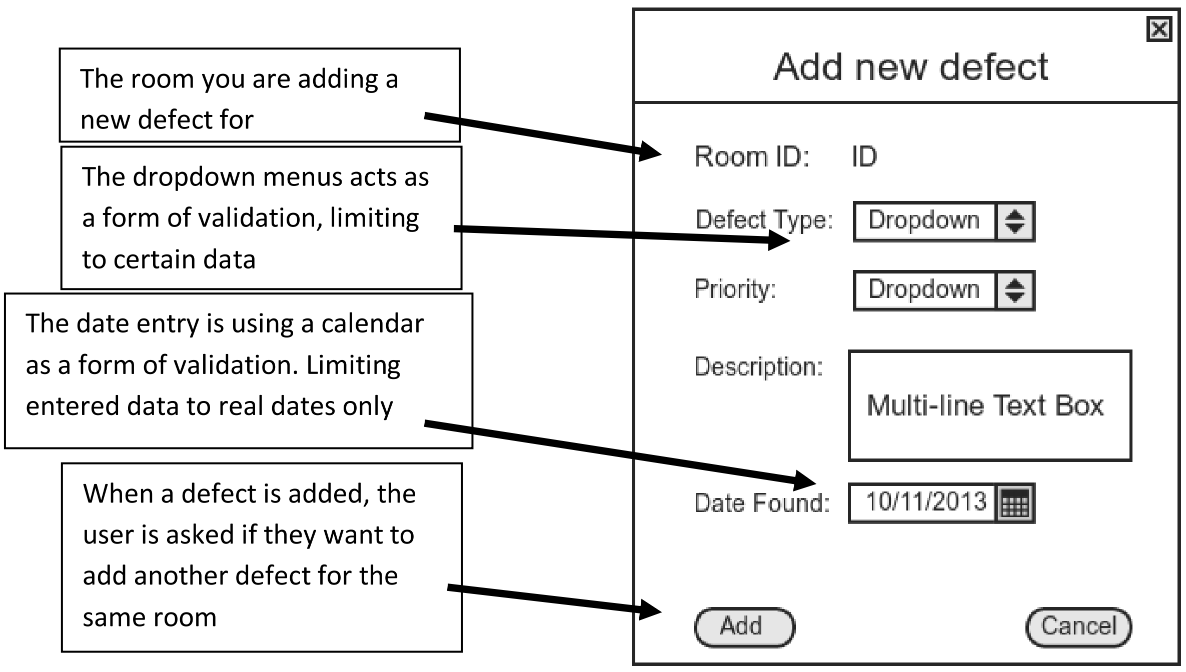
Buttons for adding and viewing defects or printing a report

The button is only visible and enabled to admin users and shows the controls in the box which allow the user to edit the building plan

Example of 6 rooms in the building plan

This area is used to display the building plan

Add New Defect Form



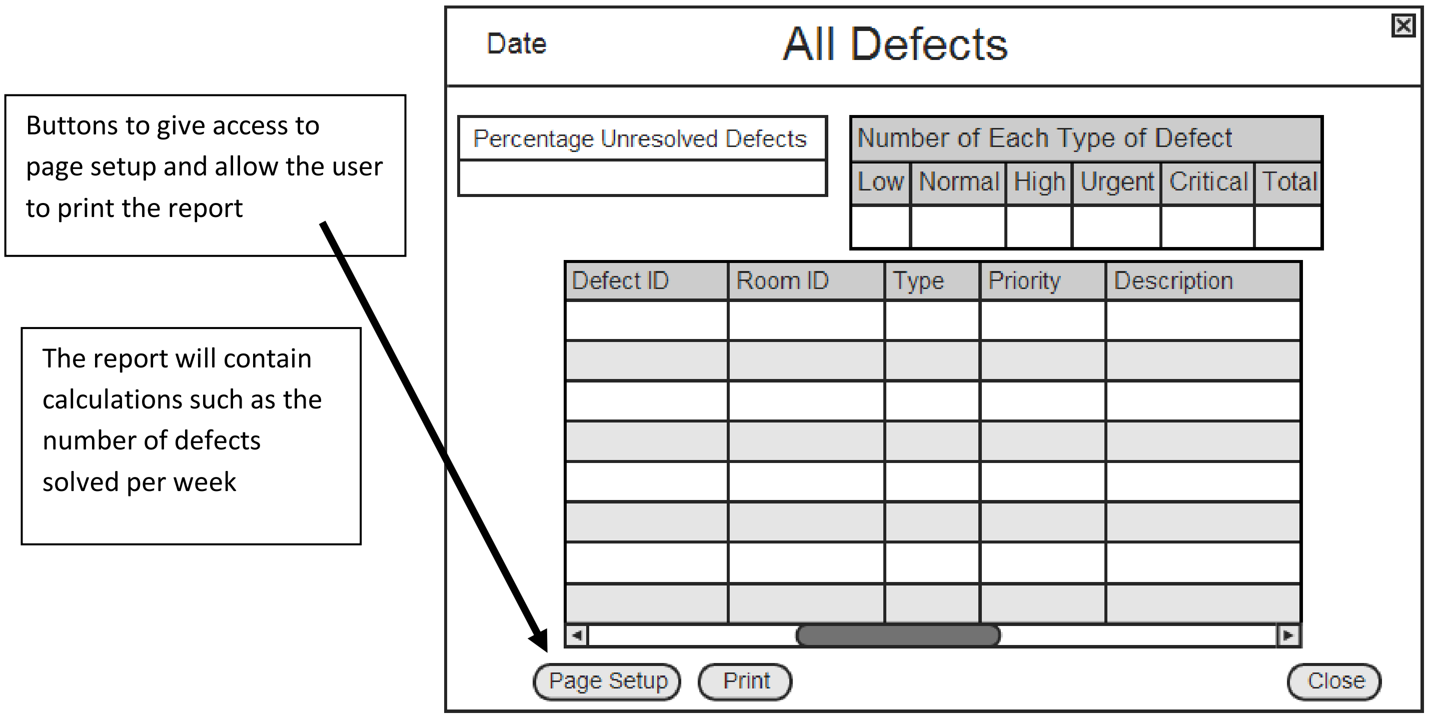
The room you are adding a new defect for

The dropdown menus acts as a form of validation, limiting to certain data

The date entry is using a calendar as a form of validation. Limiting entered data to real dates only

When a defect is added, the user is asked if they want to add another defect for the same room

Report View Form



Buttons to give access to page setup and allow the user to print the report

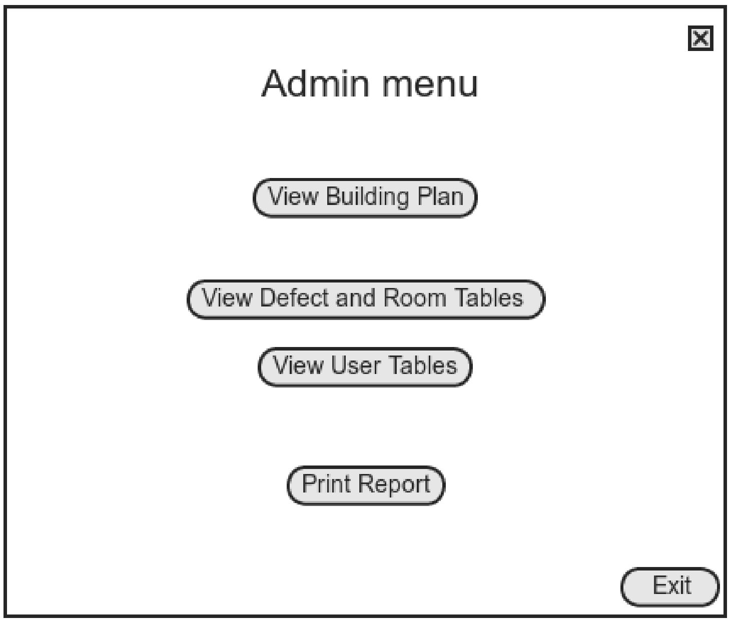
The report will contain calculations such as the number of defects solved per week

Admin Menu Form

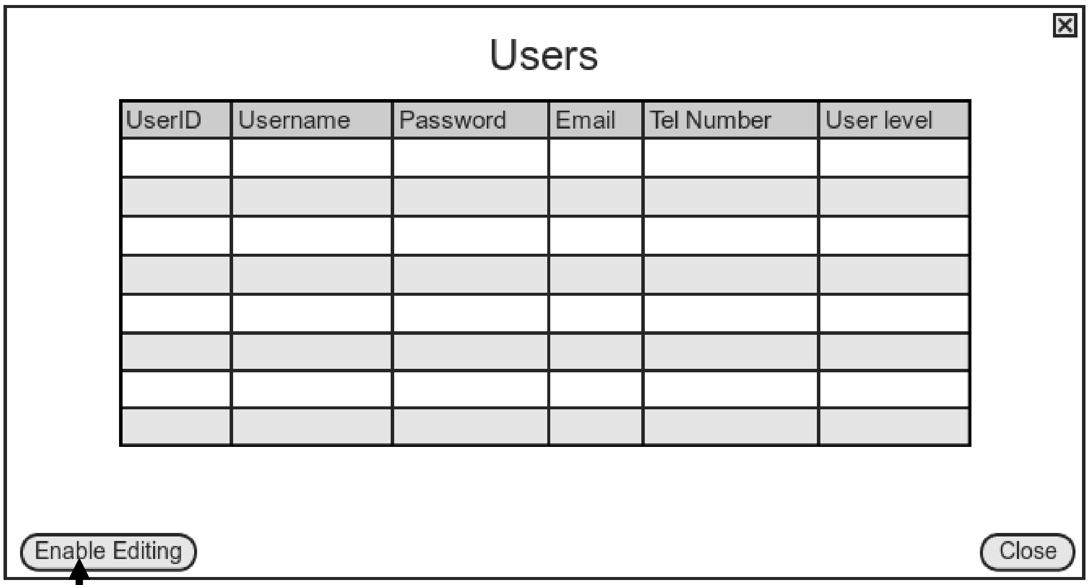
This is the form the user is first brought to after logging into an admin account.

It allows access to all the forms including access to the user table to add or edit user records.

When on other forms, when logged in as an admin, the "enable editing" buttons will be visible and enabled



User Table Form



Button to enable the editing of records

The only way to access this page is through the admin menu

Files and data structures

I will be using three relational tables. RoomTable, DefectTable and UserTable. The DefectTable will be the parent of both the RoomTable and the UserTable.

Room Table

Field Name	Data type	Description	Length	Example
(Primary Key) Room ID	String	A unique ID for each room. Will typically be the name/ number of the room	25	"E44" or "Connel Room"
Room Type	String	The type of room	30	"Bedroom", "Lab", "Office"
Room Description	String	A description of the room for further identifying qualities.	150	"The fourth room on the right with the large red doors"

Defect Table

Field Name	Data type	Description	Length	Example
(Primary Key) Defect number	Integer	A unique, auto incrementing integer for	8	1, 2, 3 etc.
(Foreign Key) Room ID	String	The ID of the room the defect is for	25	"E44" or "Connel Room"
Type	String	The type of defect the defect is	15	"Building Works", "FFE"
Priority	String	The priority of the defect	8	"Critical", "Urgent", "High", "Normal", "Low"
Description	String	A description as to what the defect is	100	"Light does not work", "Window is cracked"
Date found	Date	The date the defect was found	15	"12/11/13"
Date resolved	Date	The date the defect was resolved	15	"17/12/13"
(Foreign Key) UserID	String	The username of the user who added the defect	20	"RJones"

User Table

Field Name	**Data type**	**Description**	**Length**	**Example**
(Primary Key) UserID	String	A unique id for each user	20	“RJones”
Password	String	The password for the user to login with	25	“GreenRadiatorCamel”
Email	String	The email address of the user	40	“RJones@company.com”
Tel Number	String	The telephone number of the user	11	“01436268874”
User Level	String	The Level of the user	5	“User”, “Admin”

A JSON file will also be used. It will store 5 pieces of information: name, locationX, location, SizeX, SizeY and penColor.

Below is an example of data stored in the JSON file:

```
 {
    "name": "E1",
    "locationX": 27,
    "locationY": 250,
    "sizeX": 200,
    "sizeY": 200,
    "penColor": "default"
  },
```

The data from the JSON file is then deserialised into a list of a custom class objects with these properties:

Variable name	**Data type**
Name	String
location	Integer
location	Integer
SizeX	Integer
SizeY	Integer
penColor	Integer

So the structure of the list is such:

```
rectangleList[
          [0]
           rectangle[
                      name
                      locationX
                      locationY
                      SizeX
                      SizeY
                      penColor
                   ]
          [1]
           rectangle[
                      name
                      locationX
                      locationY
                      SizeX
                      SizeY
                      penColor
                   ]
          ]
```

The name of the class is “rectangle”
The name of the list is “rectangleList”

List for Combo Boxes – Implemented as arrays
On the “Add New Defect” form, there are two combo boxes used for input. The data structures behind these boxes are as follows:

defectTypeComboBox:

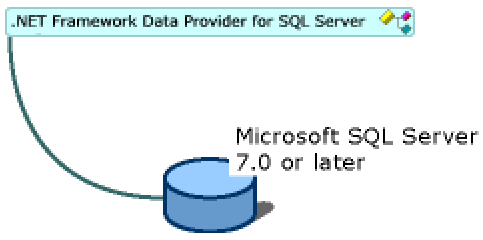
- “Building Works”
- “FFE”
- “M and E”

priorityComboBox:

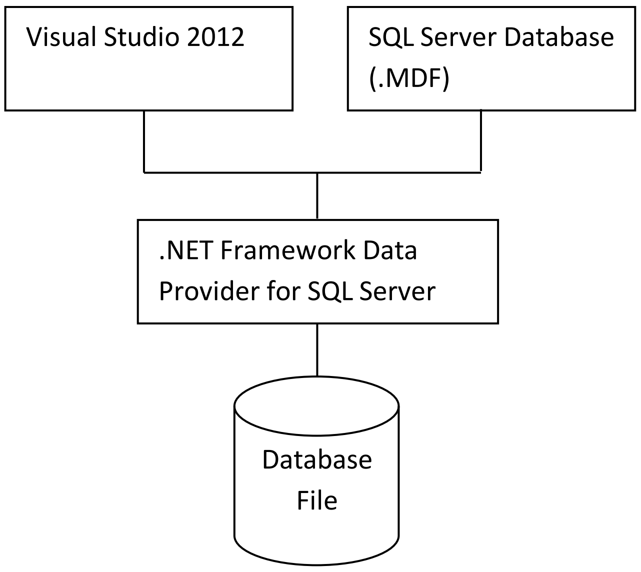
- “Critical”
- “Urgent”
- “High”
- “Normal”
- “Low”

Methods of access

I will be using an SQL Server Express database which allows for the database to be accessed across a network which will be important for my solution. The DBMS I will be using is the ".NET Framework Data Provider for SQL Server" The DBMS will assist with manipulating data in the database by providing methods for adding and editing data as well as preventing database corruption and errors. It also simplifies the otherwise complex tasks involved in connecting to the database.


.NET Framework Data Provider for SQL Server
Microsoft SQL Server 7.0 or later

The file access I will be using will be indexed sequential. This will allow the location of a file through checking each record but will also allow the location of records using indexes, speeding up access times. This will be used for all three tables – RoomTable, DefectTable and UserTable- as sorting and filtering will be useful for all tables as there will be a large number of records to sort and filter through.


Visual Studio 2012
SQL Server Database (.MDF)
.NET Framework Data Provider for SQL Server
Database File

Validation

For validating data entered into the system, I will use regular expressions(Regex). A regular expression is a text string for describing a search pattern. I will check entered data against the regular expression containing the desired format for the entered data. If the data matches the regular expression then the data is validated.

Size and Position of rooms on the building plan

I will use a regular expression to make sure the entered data is a number. An example of the Regex I will use is this: (^\d$)

```
IF(data IS number) THEN
     throw error
ENDIF
```

New defect form

A presence check will be performed on each data entry control to make sure data has been entered or selected for each. The presence check will check to see if there is at least one character (excluding spaces) entered in the field.

```
If(data IS all white space OR IS empty) THEN
     throw error
ENDIF
```

For the date entry, I will use a date picker which limits the user to entering only real dates, providing a form of validation.

There is a textbox for entering a description for the defect. I will not apply validation other than the presence check to the textbox as the description is allowed to be anything, but there must be a description.

All other data entries for adding a new defect are done by the use of dropdown lists which limit the user to entering certain values.

New Room Form

I will check that there is data entered in each textbox not including whitespaces. I will trim the contents of the textbox which ignores whitespaces and if this results in the textbox being empty then there was no data in the textbox

New User Form

I will use two Regular Expressions, one for phone numbers and one for email addresses. These regexes will only match with data that is a valid phone number or a valid email

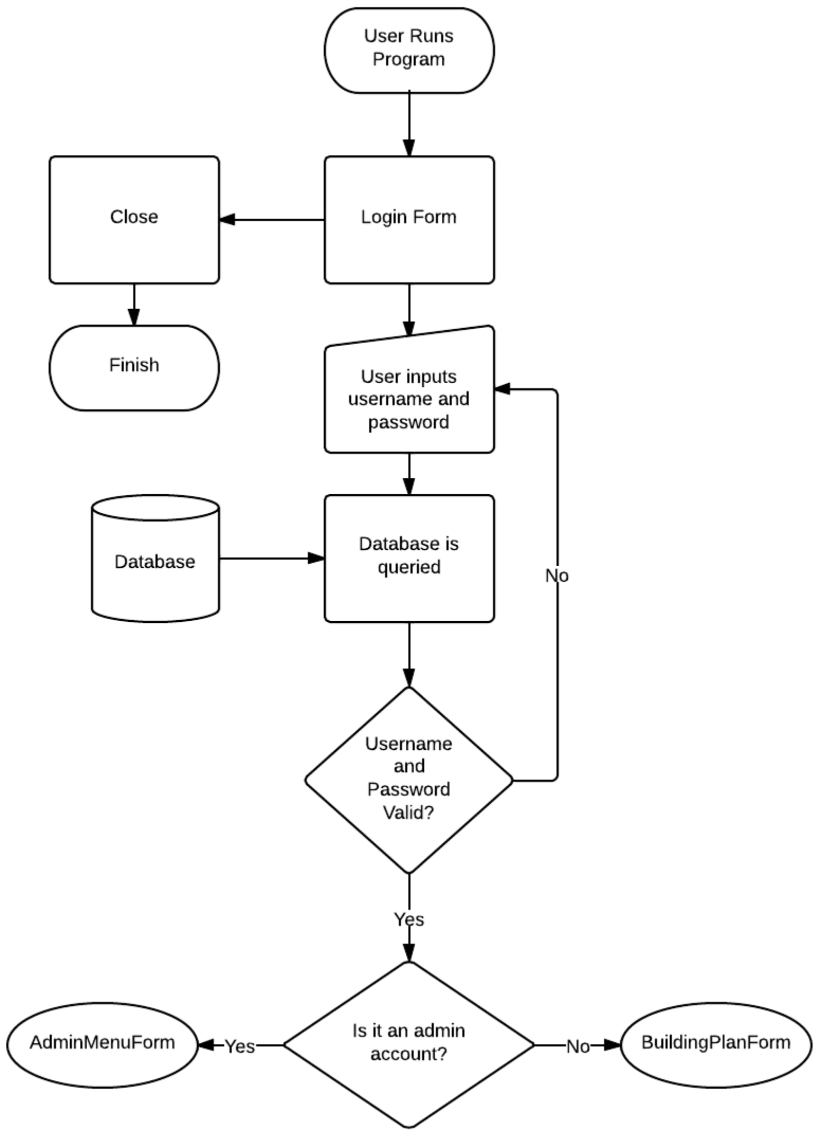
Filtering on Table View Forms

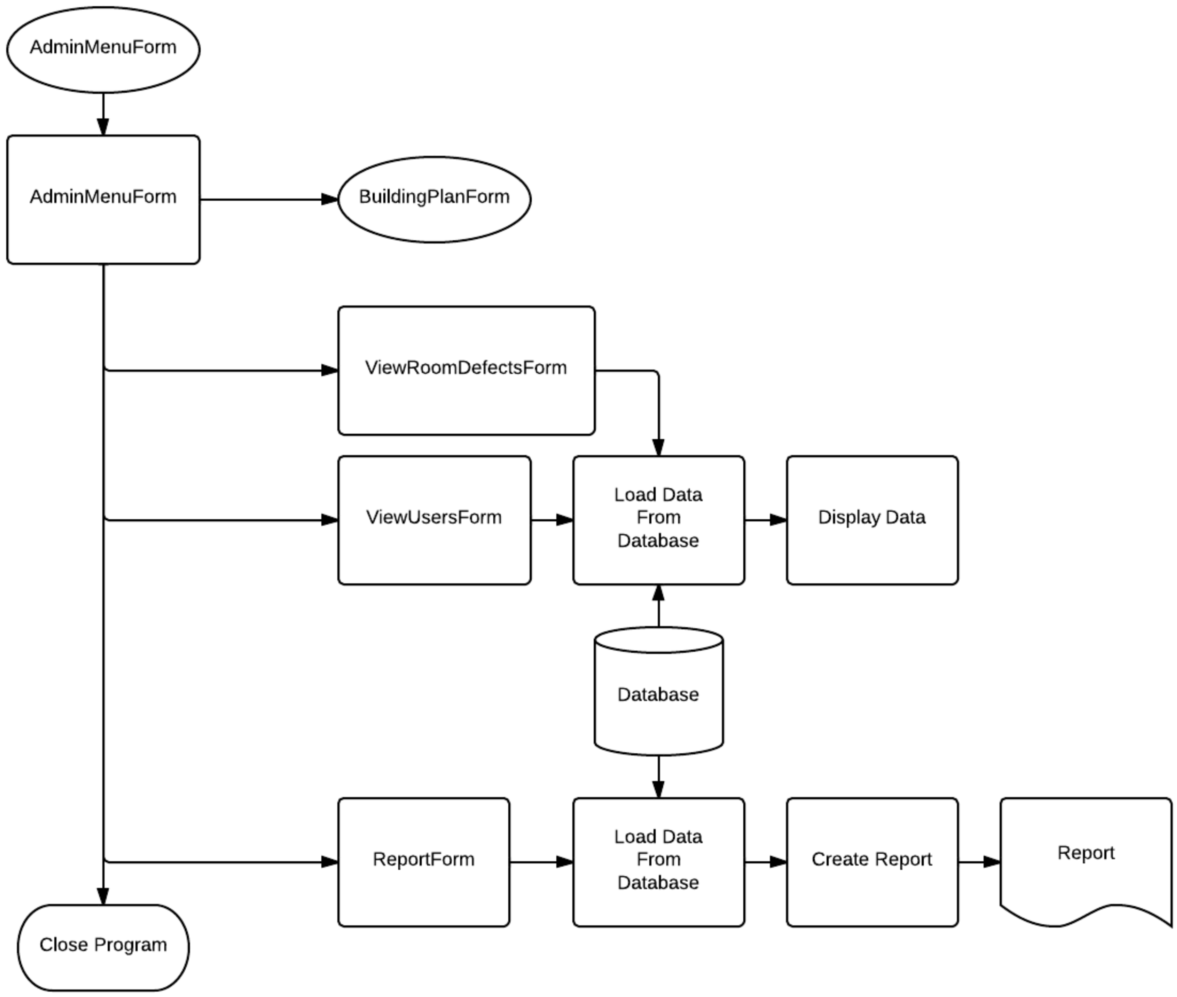
Validation will be applied to the textbox where the filter data is typed dependant on which field has been selected to be filtered by. If the field is a number-only field, I will validate to make sure the entered data contains only numbers else this can cause a runtime error.

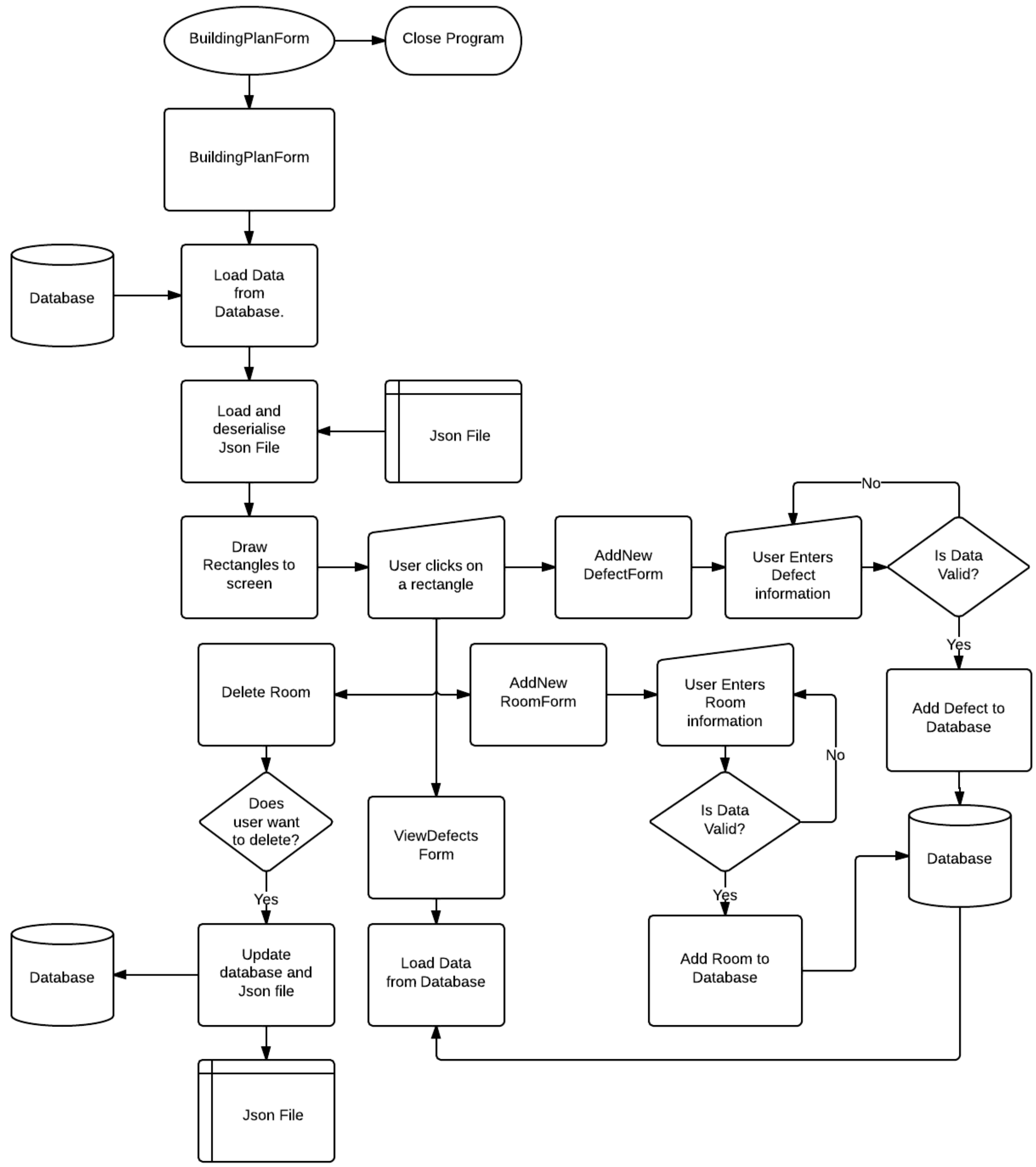
```
IF(field IS number only field AND data NOT number) THEN
     throw error
ENDIF
```

Processing Stages

System Flowchart




AdminMenuForm
AdminMenuForm
BuildingPlanForm
ViewRoomDefectsForm
ViewUsersForm
Load Data From Database
Display Data
Database
ReportForm
Load Data From Database
Create Report
Report
Close Program



Pseudocode

Login Form

```
Procedure LoginFormLoad

        Boolean fileIsFound = false
        CREATE Regular Expression called pathRegex to match with text that contains "rectanglesOut.json"
        FOREACH String in Global path variable
                CREATE variable currentString of type String = currentString from Foreach
                IF pathRegex matches with currentString THEN
                        Global variable JSONPath = currentString
                        fileIsFound = true
```

```
            ENDIF
        END FOREACH
        IF !fileIsFound THEN
            SHOW MessageBox stating the file could not be found and asking if the user wants to generate one
            IF MessageBox result = yes THEN
                CREATE a List of type rectangle called tempRectangles
                Int I = 0
                WHILE I <= number of rows in RoomTable DO
                    CREATE a new rectangle with name from current row’s RoomID value
                    ADD rectangle to tempRectangles
                    i++
                END WHILE
                Serialise tempRectangles into JSON and write it to a new JSON File
                SHOW MessageBox stating that the file has been created and that the program will CLOSE
                CLOSE Program
            ENDIF
        ENDIF
END Procedure
```

```
Procedure LoginButtonClick

        Variable Row = Find user record from database
        IF Password Text Box = Password from Record THEN
            String userlevel = Userlevel from Record
            IF Userlevel from Record = “Admin” THEN
                SHOW AdminMenuForm
                Hide LoginForm
            ENDIF
            IF Userlevel from Record = “User” THEN
                SHOW AdminMenuForm
                Hide LoginForm
            ENDIF

        ENDIF
        ELSE

            SHOW error message “The username or password was incorrect”
        ENDIF

END Procedure
```

Building Plan Form

```
Procedure BuildingPlanForm

Boolean editingEnabled = false
Class rectangle
        String name
        Integer locationX
        Integer locationY
        Integer sizeX
        Integer sizeY
        String penColor
END Class

List of type rectangle rectangles
Integer     mouseOffsetX
Integer mouseOffsetY
String currentRectangleName = “”
CREATE variable numberRegex of type RegularExpression of regular expression “^[0-9]+$”

Procedure BuildingPlanForm Load
        IF userlevel IS NOT “admin” THEN
                Hide and disable controls
        ENDIF
        IF userlevel IS “admin” THEN
                SET controls.enabled = editingEnabled
        ENDIF
        Disable viewDefectsBtn
        Disable newDefectBtn
        pictureBox1.WaitToLoadImage = true
        CREATE variable sr of type StreamReader with path of rectangle information json file
        String rectangleJSON = sr.ReadToEnd
        CREATE variable serialiser of type JavaScriptSerialiser
        rectangles = serialiser.Deserialise rectangleJSON Into form of List of type rectangles
        CLOSE sr
END Procedure

Procedure SaveButtonClick
        FOREACH rectangle in rectangles
                CREATE variable thisrectangle of type rectangle = currentrectangle from Foreach
                thisrectangle.penColor = “default”
        END FOREACH
        String json = serialise the contents of rectangles
        Write json to original json file
END Procedure

Procedure timerTick
        CREATE graphics object for pictureBox1
        CREATE variable defaultPen of type Pen of colour Black and thickness 3.0
        CREATE variable defaultPen of type Pen of colour HotPink and thickness 3.0
        CREATE variable brush of type Brush of colour Black
        CREATE variable font of type Font of font Ariel , size of 13 and style of Bold

        CREATE variable bm of type Bitmap of width pictureBox1.Width and height pictureBox1.Height
        CREATE  variable offScreenGraphics of type Graphics fromImage bm
        offScreenGraphics.Clear with colour White
        USING CREATE variable gr of type Graphics fromImage bm
                FOREACH rectangle in rectangles
                        CREATE variable thisRectangle of type rectangle = currentrectangle from Foreach
                        IF thisRectangle.sizeX <= 10 THEN
                                thisRectangle.sizeX = 10
                        ENDIF
                        IF thisRectangle.sizeY <= 10 THEN
                                thisRectangle.sizeY = 10
                        ENDIF
                        IF thisRectangle.penColor == “default” THEN
                                offScreenGraphics.DrawRectangle with values Pen of defaultPen, Position of locationX
and locationY with Size of sizeX and sizeY from thisRectangle
                                offScreenGraphics.DrawString with values Text of name, Font of font, Brush of brush,
Location of locationX + (sizeX /2.5) and locationY + (sizeY/2.5)
                        ENDIF

                        IF thisRectangle.penColor == “pink” THEN
                                offScreenGraphics.DrawRectangle with values Pen of selectedPen, Position of locationX
                and locationY with Size of sizeX and sizeY from thisRectangle
```

```
                                        offScreenGraphics.DrawString with values Text of name, Font of font, Brush of brush,
                    Location of locationX + (sizeX /2.5) and locationY + (sizeY/2.5)
                              ENDIF
                    END FOREACH

          END USING
pictureBox1.Image = bm

END Procedure

Procedure picturebox1MouseDown sender variable e

          Disable viewDefectsBtn
          Disable newDefectBtn
          IF e.Button == Left Mouse Button THEN
                    FOREACH rectangle in rectangles
                              CREATE variable thisRectangle of type rectangle = currentrectangle from Foreach
                              thisRectangle.penColor = “default”
                              IF currentRectangleName == “” THEN
                                        IF e.X >= thisRectangle.locationX AND e.X <= thisRectangle.locationX   +thisRectangle.sizeX
AND e.Y >= thisRectangle.locationY AND e.Y <= thisRectangle.locationY +thisRectangle.sizeY THEN
                                                  currentRectangleName = thisRectangle.name
                                                  var row = Find row in RoomTable by Room_ID using the variable
currentRectangleName
                                                  roomIDLabel.Text = row.Room_ID
                                                  roomTypeLabel.Text = row.Room_Type
                                                  roomDescLabel.Text = row.Room_Description
                                                  MyGlobals.selectedRoom = thisRectangle.name;
                                                  Integer defectNum = 0
                                                  Integer i = 0
                                                  WHILE i < number of rows in DefectTable DO
                                                            IF DefectTable[i].Value == thisRow.Room_ID THEN
                                                                      defectNum++
                                                            ENDIF
                                                            i++
                                                  END WHILE
                                                  numberDefectsLabel.Text = defectNum
                                                  mouseOffsetX = e.X – thisRectangle.locationX
                                                  mouseOffsetY = e.Y – thisRectangle.locationY
                                                  thisRectangle.penColor = “pink”
                                                  RectangleWidthBox.Text = thisRectangle.sizeX as type String
                                                  RectangleHeightBox.Text = thisRectangle.sizeY as type String
                                                  rectangleLocationBoxX.Text = thisRectangle.locationX as type String
                                                  rectangleLocationBoxY.Text = thisRectangle.locationY as type String
                                                  Set viewDefectsBtn to enabled
`                                                 Set newDefectBtn to enabled
                                        ENDIF
                                        ELSE
                                                  thisRectangle.penColor = “default”
                                        ENDELSE
                              ENDIF
                    ENDIF
          END FOREACH
END Procedure

Procedure pictureBox1MouseMove sender variable e

IF e.Button == Left Mouse Button AND editingEnabled != false THEN
          FOREACH rectangle in rectangles
```

```
                CREATE variable thisRectangle of type rectangle = currentrectangle from Foreach
                IF thisRectangle.Name == currentRectangleName THEN
                        IF e.X >= thisRectangle.locationX AND e.X <= thisRectangle.locationX  +thisRectangle.sizeX AND e.Y
>= thisRectangle.locationY AND e.Y <= thisRectangle.locationY +thisRectangle.sizeY THEN
                                thisRectangle.locationX = e.X – mouseOffsetX
                                thisRectangle.locationY = e.Y – mouseOffsetY
                                rectangleLocationBoxX.Text = thisRectangle.locationX as type String
                                rectangleLocationBoxY.Text = thisRectangle.locationY as type String
                        ENDIF
                ENDIF
        END FOREACH
END Procedure

Procedure pictureBox1MouseUp

        currentRectangleName = “”
        mouseOffsetX = 0
        mouseOffsetY = 0
END Procedure

Procedure rectangleIncrementXClick

        FOREACH rectangle in rectangles
                CREATE variable thisRectangle of type rectangle = currentrectangle from Foreach
                IF thisRectangle.penColor == “pink” THEN
                        thisRectangle.locationX  += 1
                        rectangleLocationBoxX.Text = thisRectangle.locationX as type String
                ENDIF
        END Foreach
END Procedure
```

Repeat code for decrementX but subtract 1 from thisRectangle.locationX
Repeat both increment and decrement code replacing X by Y
Repeat all 4 for size adjustment buttons changing X and Y for Width and Height

```
Procedure rectangleLocationBoxXKeyPress sender variable e

        IF e.KeyChar== Return Key THEN
                FOREACH rectangle in rectangles
                        CREATE variable thisRectangle of type rectangle = currentrectangle from Foreach
                        IF thisRectangle.penColor == “pink” THEN
                                IF numberRegex.IsMatch with rectangleLocationBoxX.Text THEN
                                        thisRectangle.locationX = rectangleLocationBoxX.Text as type Integer
                                ENDIF
                        ENDIF
                END FOREACH
        ENDIF
END Procedure
```

Repeat for BoxYKeyPress changing X to Y
Repeat both changing X and Y to width and height

```
Procedure editButtonClick
        IF editingEnabled == false THEN
                editingEnabled = true
                editControlsGroup.Enabled = editingEnabled
                editBtn.Text = “Disable Editing”
                return
        ENDIF
        IF editingEnabled == true THEN
                editingEnabled = false
                editingControlsGroup.Enabled = editingEnabled
                editBtn.Text = “Enable Editing”
                Return
        ENDIF
END Procedure

Procedure editBtnClick
        IF editingEnabled == false THEN
                editingEnabled = true
                Set enabled state of editControlsGroup to value of editingEnabled
                editBtn.Text = “Disable Editing”
                return
        ENDIF
        IF editingEnabled == true THEN
                editingEnabled = false
                Set enabled state of editControlsGroup to value of editingEnabled
                editBtn.Text = “Enable Editing”
                return
        ENDIF
END Procedure

Procedure viewDefectsBtnClick

        Hide buildingPlanForm
        Disable the timer
        SHOW ViewDefectsForm
        SHOW buildingPlanForm
        Enable the timer
END Procedure

Procedure newDefectBtnClick

        Hide buildingPlanForm
        Disable the timer
        SHOW NewDefectForm
        SHOW buildingPlanForm
        Enable the timer
        Reload Data from the Tables
END Procedure

Procedure buildingPlanFormOnClosing

        Disable the timer
END Procedure

Procedure addRoomBtnClick

        Hide buildingPlanForm
        Disable the timer
        SHOW NewRoomForm
        Enable the timer
        Reload Data from the Tables
END Procedure

Procedure deleteBtnClick
```

```
        IF SHOW Warning MessageBox Result == Yes THEN
                Remove rectangles with name == roomIDLabel
                var row = Find row in RoomTable by Room_ID using roomIDLabel
                Delete row from table
                Update Table in database

                Int i =0
                WHILE i < number of rows in DefectTable DO
                        IF Row[i] Cell[1] == roomIDLabel THEN
                                Remove Row[i]
                        ENDIF
                        i++
                END WHILE
                Update Table in database



                FOREACH rectangle in rectangles DO
                        CREATE variable thisRectangle of type rectangle = currentrectangle from Foreach
                        pencolor of thisRectangle = "default"
                END FOREACH
                String JSON = Serialise rectangles to JSON
                Write JSON to file
                SHOW MessageBox for confirm deletion
                roomIDLabel = ""
                roomDescLabel = ""
                roomTypeLabel = ""
                numberDefectsLabel = ""
        ENDIF
```

----------------------------More goes here?-------------------------------

END Form Procedure

New Defects Form

Procedure NewDefectsForm

```
Procedure NewDefectsFormLoad

        Load data from Table
        RoomIDLabel.Text = MyGlobals.selectedRoom
        Set selected item in DefectTypeBox to first item
        Set selected item in PriorityBox to second item
        Set max date of Date picker to current date
END Procedure

Procedure DescriptionTBoxTextChanged

        Set back colour of DescriptionTBox to default colour
        IF Length of DescriptionTBox >= 90 THEN
                Set back colour of DescriptionTBox to Coral
        ENDIF
        IF Length of DescriptionTBox == 100 THEN
                Set back colour of DescriptionTBox to OrangeRed
        ENDIF
END Procedure

Procedure DescriptionTBoxEnter
```

```
        CREATE tooltip object with name TT
        Remove all tooltips from screen
        SHOW new tooltip with text “Character Limit is 100” on the DescriptionTBox

END Procedure

Procedure DescriptionTBoxMouseClick

        CREATE tooltip object with name TT
        Remove all tooltips from screen
        SHOW new tooltip with text “Character Limit is 100” on the DescriptionTBox

END Procedure

Procedure DescriptionTBoxLeave

        Set back colour of DescriptionTBox to default colour
END Procedure

Procedure cancelBtnClick

        SHOW MessageBox with text “Are you sure you want to cancel?” with title “Cancel?” with Yes and No buttons
        IF Result of MessageBox is Yes THEN
                CLOSE this Form
        ENDIF
END Procedure

Procedure addBtnClick

        Insert new row into table with values from data entered by user
        Update table
        SHOW MessageBox with text “New defect has been added” with title “Success” with OK button
        CLOSE this Form
END Procedure

END Form Procedure
```

ViewDefectsForm

```
Procedure ViewDefectsForm

Int SelectedRowIndex

Procedure ViewDefectsFormLoad
        Load data from Database
        Disable ResolveBtn
        String filter
        String token = selectedRoom
        filter = RoomID Like SelectedRoom
        DefectTable Filter = filter
END Procedure

Procedure CLOSEBtnClick
        CLOSE form
END Procedure

Procedure resolveBtnClick
        Int i = 0
        FOREACH row in DefectTable DO
                CREATE variable thisRow of type Row = currentRow from Foreach
                IF RoomID of thisRow == selectedRoom THEN
                    break
                ENDIF
                i++
        END FOREACH
        Var row = DefectTable Row[selectedRowIndex + i]
        IF isResolved cell value of row == null THEN
                DateTime previousDate = resolveDate
                SHOW ResolveForm
                IF resolveDate != previousDate THEN
                    dateResolved Cell of row = resolveDate
                    Update DefectTable
                    Reload data from Table
                ENDIF
        ENDIF
END Procedure

Procedure dataGridViewCellClicked
        Enable resolveBtn
        selectedRowIndex = Index of row that was clicked on
END Procedure
END Form Procedure
```

ResolveForm

```
Procedure ResolveForm
Procedure ResolveFormLoad
        Maxdate of dateTimePicker = current date
END Procedure

Procedure resolveBtnClick
        resolveDate = dateTimerPicker Value
        CLOSE ResolveForm
END Procedure
END Form Procedure
```

ViewUsersForm

```
Procedure ViewUsersForm
Procedure newUserBtnClick
        HIDE ViewUsersForm
        SHOW NewUserForm
        SHOW ViewUsersForm
        Reload data from Table
END Procedure
END Form Procedure
```

NewUserForm

```
Procedure NewUserForm
CREATE Regular Expression called EmailRegex for that matches with valid emails
CREATE Regular Expression called TelRegex for that matches with valid telephone numbers

Procedure NewUserFormLoad
        DISABLE newUserBtn
        SET selected index of UserLevel combo box to 0
END Procedure
Procedure BookBtnEnable
        IF Length of UserIDBox != 0 AND password1Box == pasword2Box AND Length of password1Box != 0 THEN
            IF Length of EmailBox == 0 AND Length of TelBox == 0
```

```
                    OR EmailBox matches EmailRegex AND Length of TelBox == 0
                    OR Length of EmailBox == 0 AND TelBox matches TelRegex
                    OR EmailBox matches EmailRegex AND TelBox matches TelRegex
                    THEN
                    ENABLE newUserBtn
                    ValidationErrorLabel = “”
            ENDIF
            ELSE
                    DISABLE newUserBtn
                    ValidationErrorLabel = “One or more fields contain incorrect information”
            ENDELSE
    ENDIF

    ELSE
            DISABLE newUserBtn
            ValidationErrorLabel = “One or more fields contain incorrect information”
    ENDELSE

END Procedure

Procedure ValidateableTextBox_Leave
    IF BackColor of Sender  == White OR BackColor of Sender == Green OR Length of text in Sender  == 0 THEN
            BackColor of Sender = White
    ENDIF
    ELSE
            BackColor of Sender  = OrangeRed
    ENDELSE
    Call BookButtonEnable
END Procedure

Procedure userIDBox_TextChanged

    IF TextLength of Sender != 0 THEN
            BackColor of Sender = Green
    ENDIF
    ELSE
            BackColor of Sender = OrangeRed
    ENDELSE
END Procedure

Procedure password1Box_TextChanged

    IF Text of Sender == password2Box Text THEN
            BackColor of Sender = Green
    ENDIF
    ELSE
            BackColor of Sender = OrangeRed
    ENDELSE

    IF password2Box Text == password1Box Text THEN
            BackColor of password2Box = Green
    ENDIF
    ELSE
            BackColor of password2Box = OrangeRed
    ENDELSE
    Call BookButtonEnable
END Procedure

Procedure cancelbtnClick
```

```
        SHOW MessageBox with text “Are you sure you want to cancel?” with title “Cancel?” with Yes and No buttons
        IF Result of MessageBox is Yes THEN
                CLOSE this Form
        ENDIF
END Procedure

Procedure newUserBtnClick

        INSERT new Row into UserTable using the values from the controls
        UPDATE Database
        SHOW MessageBox with text “A new user has been added” with title “Success” with OK button
        CLOSE this Form

END Procedure
END Form Procedure
```

NewRoomForm

```
Procedure NewRoomForm

CREATE Regular Expression called whiteSpaceRegex for that matches with whitespace only

Procedure AddRoomBtnClick

        IF RoomTable does not contain Row with RoomID of roomIDBox AND roomIDBox Trimmed Text != “” THEN
                IF roomIDBox Trimmed Text != “”AND roomTypeBox Trimmed Text != “” AND roomDescripBox Trimmed Text
                != “” THEN
                        INSERT new Row in RoomTable WITH values roomIDBox, roomTypeBox and roomDescripBox
                        CREATE new rectangle WITH
                                Name = roomIDBox
                                locationX = 0
                                locationY = 0
                                sizeX = 100
                                sizeY = 100
                                penColor = “default”
                        ADD Rectangle to rectangles
                        FOREACH rectangle in rectangles DO
                                CREATE variable thisRectangle of type rectangle = currentrectangle from Foreach
                                pencolor of thisRectangle = “default”
                        END FOREACH
                        String JSON = Serialise rectangles to JSON
                        Write JSON to file
                        SHOW MessageBox with text “New room was successfully added” with title “Success” with OK
        button
                        CLOSE this Form
                ENDIF
                ELSE
                        SHOW MessageBox with text “Could not add new room. Please check there is data entered in all
        boxes” with title “Error” with OK button
                ENDELSE
        ENDIF
        ELSE
                SHOW MessageBox with text “Could not add new room. Please check there is not already a room with the
        same ID and the RoomID field is not empty” with title “Error” with OK button
        ENDELSE

END Procedure

Procedure cancelbtnClick
```

```
        SHOW MessageBox with text “Are you sure you want to cancel?” with title “Cancel?” with Yes and No buttons
        IF Result of MessageBox is Yes THEN
                CLOSE this Form
        ENDIF
END Procedure
```

Reporting Pseudocode

```
Procedure PercentageDefects

        Int count = 0
        FOR i = 0, I <= TotalRecords, i++ DO
                        if Record[i] Resolved == Null THEN
                                count++
                        ENDIF
        END FOR

        OUTPUT ((count/TotalRecords)*100)

END Procedure

Procedure DateFormat

        OUTPUT DateField as dateformat dd/mm/yyyy

END Procedure

Procedure calculatePriorityDefectsAll

        Int count = 0
        FOR i = 0, I <= TotalRecords, i++ DO
                        if Record[i] priority is Like “low” THEN
                                count++
                        ENDIF
        END FOR

        OUTPUT count

END Procedure

Procedure SetRowColour

        Int count = 0
        FOR i = 0, I <= TotalRecords, i++ DO
                String SetColor = “No Color”
                IF Record[i] priority is Like “Low” THEN
                        SetColor = “LightGreen”
                ENDIF
```

This IF statement is repeated for each priority type with SetColor set to a different colour for each different priority

```
        END FOR
        OUTPUT SetColor

END Procedure
```

Evaluation

In evaluating my solution, I will consider three main areas: Usability, Suitability and Performance.

Usability:
I shall use a questionnaire to ascertain how user friendly my solution is and how easy it is to navigate and use by users of various skill levels from both my computing class and from the rest of my school.
The questionnaire will contain several questions for which the usability of the system will be rated against from 1-5.
I shall consider the usability of the system to be acceptable if most of the questions get a 4 or 5 rating.
The questionnaire I shall use is below:

Task	1 (Very hard)	2 (Difficult)	3 (Average difficulty)	4 (Easy)	5 (Very easy)
Navigation					
Adding a new Defect					
Viewing a rooms defect					
Resolving a defect					
Adding a Room					
Printing a report					

Suitability:
I will consider the solution suitable if it meets all the aims set out for it. The solution must meet the set requirements of the problem. It must be suitable for use in a professional environment

Performance:
I will run tests on many different aspects of the solution to see whether the program functions as intended and does not break when used. The tests will use normal, extreme, incompatible and non-existent data.
Calculations must be made correctly.
If no major parts of the system fail the testing process or are very simply fixed, I will consider the performance of the system successful.