**CSC 2920 Software Engineering Report**

**11/11/2019**

**Group #2**

**Contact List**

|  |
| --- |
| **Team Members** |
| Kareem Jordan |
| Stephen Kerns |
| Chaa Loftin |
| Elias Reyes |
| Brianna Thompson |

**1. Table of Contents**

**1. Introduction 1**

1.1 Project overview 1

1.2 Problem Statement 1

1.3 Project deliverables 1

1.4 Definitions, acronyms, and abbreviations 1

**2. Project Management Plan 2**

2.1 Project organization 2

2.2 Process Model 2

2.3 Schedule 3

**3. Requirement Specifications 3**

3.1 Overall requirement specifications 3

3.2 Use Cases 3

3.3 Functional requirements 5

3.4 Non-functional requirements 6

**4. Design 7**

4.1 GUI (Graphical User Interface) design 7

4.2 Sequence Diagram 8

4.3 Class Diagram 9

**5. Implementation** **10**

5.1 Code Description 10

**6. Test Plan 10**

6.1 Techniques Used 10

6.2 Test Plan 10

**Appendix 11**

A.1 Sign-In Page 11

A.2 Contact Viewing 12

A.3 SignIn Class Summary 15

A.4 ContactViewing Class Summary 15

**1. Introduction**

**1.1 Project Overview**

We are developing a contact list that validates whether or not a contact has been submitted using the correct format. We will also provide the same parameters concerning name and email format. Our group has chosen the chief programming team structure with Brianna as our lead.

Our first step will be to plan out our project and assign given roles. We will then determine what will be required of our software. That will naturally lead into the design of software to determine the layout of our interface. Once the foundation of our program is strong, we will move into the coding/implementation phase. We will naturally bounce between the coding phase and the testing phase to determine flaws in our code and or design.

**1.2 Problem Statement**

Contact lists need a way to validate that a name and email have been given in the correct format, because the submitted data must be able to work successfully with the client’s application.

**1.3 Project Deliverables (Schedule)**

|  |  |  |
| --- | --- | --- |
| **Deliverable Number** | **Deliverable Name** | **Due Date** |
| 1 | Project Management Plan meeting minutes | 10/28/2019 |
| 2 | Software Requirements Specification  meeting minutes | 11/04/2019 |
| 3 | Software Design Description meeting minutes | 11/11/2019 |
| 4 | Code, Final Report meeting minutes | 11/18/2019 |

**1.4 Definitions, acronyms, and abbreviations (if any)**

We don’t have any.

**2. Project Management Plan**

**2.1 Project organization**

* **Brianna Thompson:** Chief Programming Lead
* **Chaa Loftin:** Implementer -Software Engineer
* **Kareem Jordan:** Implementer - Software Engineer
* **Stephen Kerns:** Implementer - Tester
* **Elias Reyes:** Implementer - Tester (2)

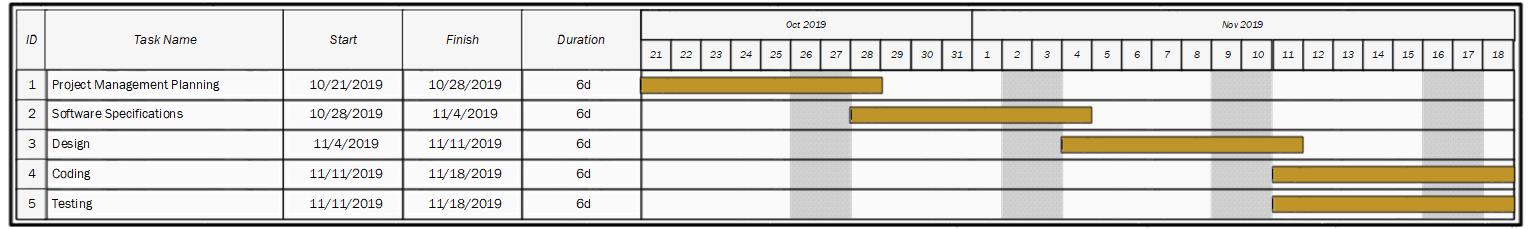
**2.2 SW Development Process Model**

The waterfall model is a linear, sequential approach to software development that starts at the system level and goes through analysis, design, coding, testing, and support. The waterfall model comprises of seven different levels:

1. **Requirements:** During this stage of the waterfall model we are defining and planning the project without mentioning anything specific.
2. **Analysis:** The system specifications are analyzed to produce a viable product model and the business logic that is needed to guide software development.
3. **Design:** The design document will outline the technical requirements needed towards software development. By choosing the programming language needed for implementation, the type of hardware used, what the data sources are, the defined architecture and the services that will be provided.
4. **Coding/Implementation:** During the coding and implementation phase, the source code will be developed using the models, the business logic and the specified requirements defined in the previous stages. The system will be designed into smaller elements prior to merging all components into a single system.
5. **Testing:** Thorough evaluations are done on the product for quality assurance. Testing will be done by ensuring the contact list functions as intended.

**2.3 Schedule**

Gantt chart

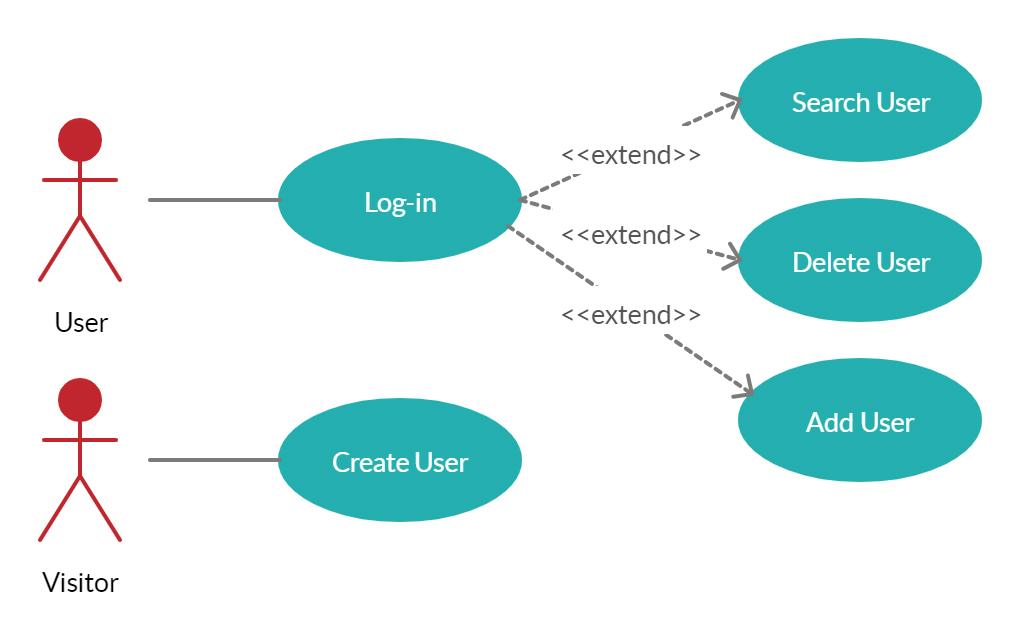


**3. Requirement Specifications**

**3.1 Overall requirement specifications**

* For the requirement specifications we have four use cases: 1. add user, 2. delete user, 3. search for user, and 4. validate user. The cases “add user” and “validate user” are initiated by the user, while “delete user” and “search for user” initiated by the user. The functional requirements are: create-contact. search-contact, delete-contact, and input. The non-functional requirements are: performance requirements, safety requirements, security requirements, and error handling.

**3.2 Use Cases**

****

|  |  |
| --- | --- |
| **Use case name:** | Add user |
| **Participating actors** | Initiated by the user |
| **Flow of events** | 1) The user inputs the required information in all text fields.  2) The system validates the required information is in the correct format.  3) The system displays text notifying the user that text is an invalid format.  4) The user’s information is added to a.CSV file. |
| **Entry conditions** | The user must be logged in and the add button must be clicked. |
| **Exit conditions** | The user has been notified by the system that all conditions have been met. |

|  |  |
| --- | --- |
| **Use case name** | Delete user |
| **Participating actors** | Initiated by the user |
| **Flow of events** | 1) The user searches for a user by inserting his/her name.  2) The system finds the user on a.CSV file.  3) The user clicks the delete button.  4) The system deletes the user. |
| **Entry conditions** | The user must be logged in, the admin must input a valid name, then click the search button, then the delete button. |
| **Exit conditions** | The system must notify the user to know that the user has been successfully deleted. |

|  |  |
| --- | --- |
| **Use case name** | Search for user |
| **Participating actors** | Initiated by the user |
| **Flow of events** | 1)  The user gives input for the name and email.  2)  The user clicks the search button.  3)  The system finds the user on.CSV file.  4)  The system notifies the user that the user has or has not been found. |
| **Entry conditions** | The user must be logged in and the search button must be clicked. |
| **Exit conditions** | The system notifies the user that the user has been found. |

**3.3 Functional requirements**

**Create-Contact**

**Input:** Details of the new contact information.

**Action:** Displays a form to be filled in by the owner and records contact information like name and email.

**Output:** Creates a new contact bar with the given information and adds it to the contact list.

**Search-Contact**

**Input:** Name of the contact searched for.

**Action:** Displays a list of possible contacts while the user types the search keywords in the search bar.

**Output:** Displays existing details of the selected contact.

**Delete-Contact**

**Input:**

• Name of the contact searched for.

• Details of the searched contact information obtained from the result set.

**Action:** Prompts the user to confirm if he/she wants to permanently delete that contact.

Output: Updates the contact bar with the newly updated information.

**3.4 Non-functional requirements**

**3.4.1. Performance Requirements**

* The system shall accommodate 1 terabyte without fault.
* Responses to view information shall take no longer than 5 seconds to appear on the screen.

**3.4.2. Safety Requirements**

System use shall not cause any harm to human users.

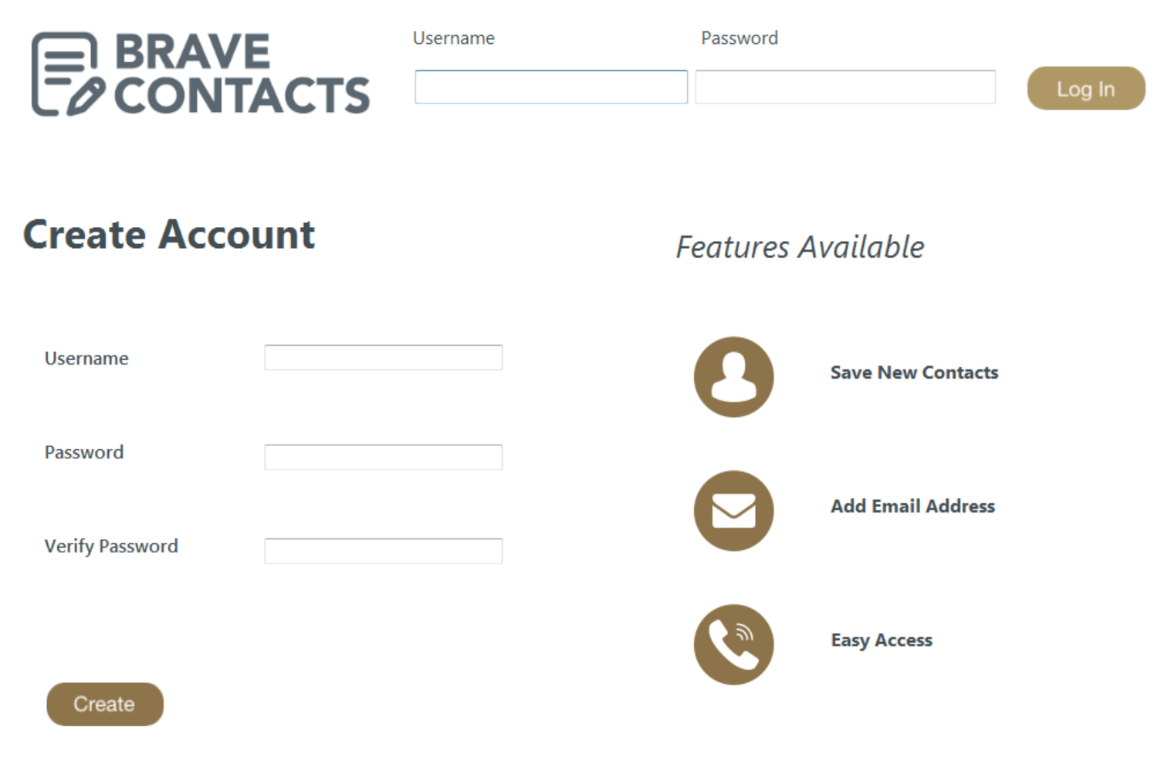
**3.4.3. Security Requirements**

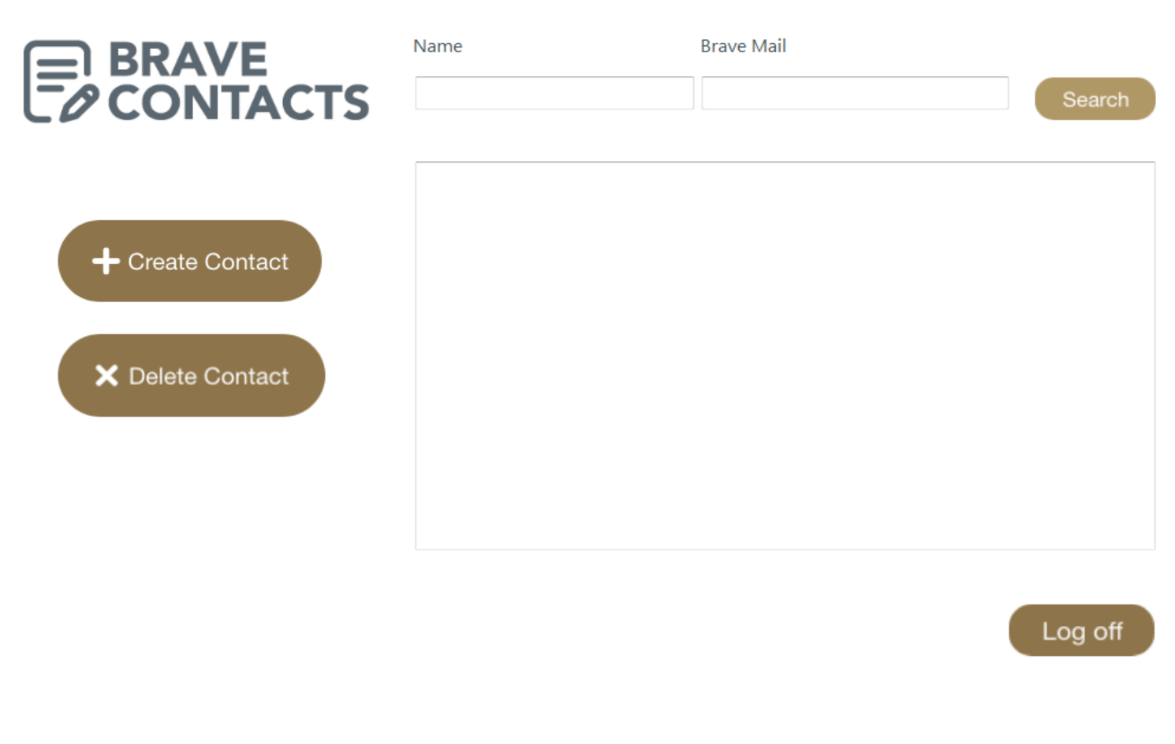
* System will validate passwords with Boolean method.
* Users should not be able to log in without having or creating an account.

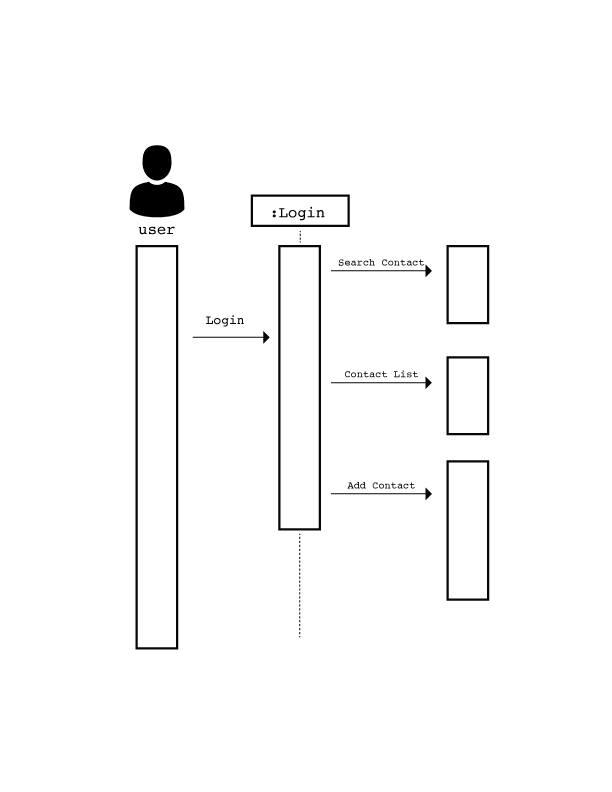
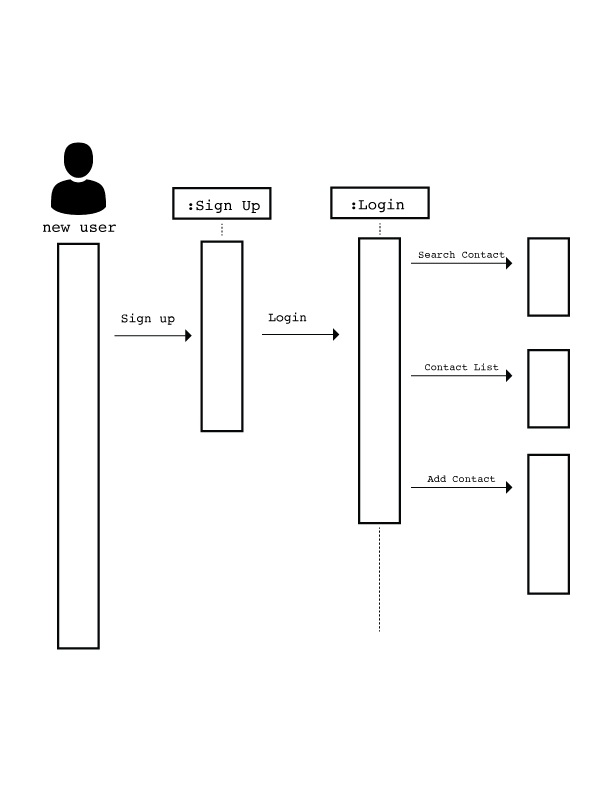
**3.4.4. Error Handling**

The Contact List shall handle all expected and unexpected errors in ways that prevent loss in information and a long downtime period. For example, if the user has issues logging in, creating an account, searching user, deleting from the contact list, or logging off.

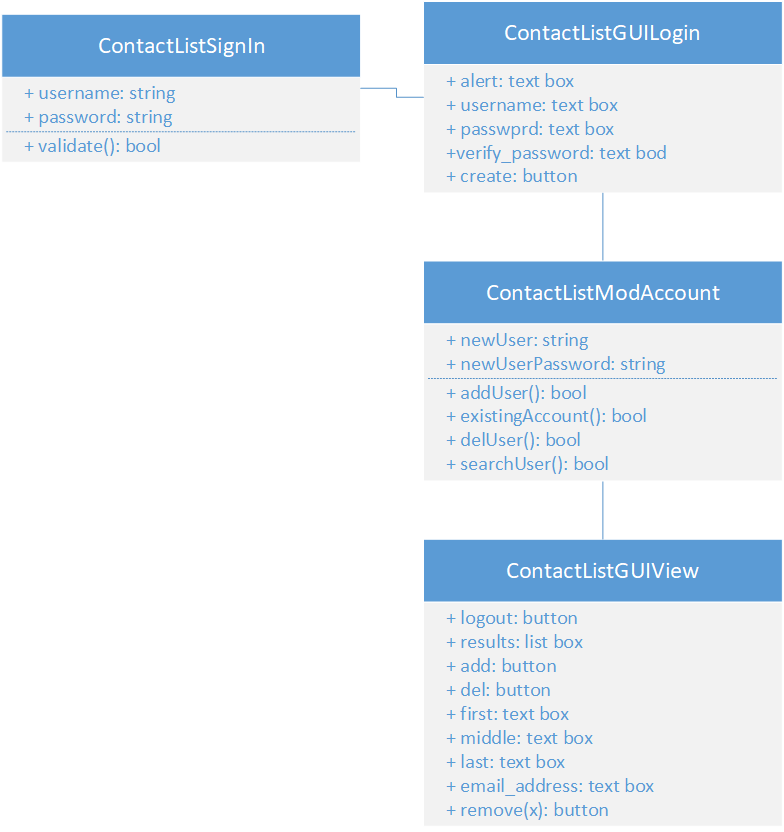
**4. Design**

**4.1 GUI (Graphical User Interface) design**

****

**4.2 Sequence Diagram**

**4.3 Class Diagram**

****

**5. Implementation**

**5.1 Code Description:** See Appendix

**6. Test Plan**

**6.1 Techniques Used:**

BlackBox Testing will be used for test generation by ensuring the proper implementation of the contact list. BlackBox Testing will be done by ensuring that a visitor can create an account, once the account has been created the visitor becomes a user. As a user, he/she can add a contact, delete a contact, and search for a contact. The Method that will be used is Error Guessing. With this technique, the software engineers will use their experience to determine the problematic areas of the application.

**6.2 Test Plan**

|  |  |
| --- | --- |
| **Test Case ID: 01** |  |
| **Purpose** | Password Verification |
| **Test assumption** | Whenever a visitor signs up to become a user he/she password must match during profile creation. If not the user will be alerted to the error and must try again before profile creation. |
| **Tester** | Stephen, Elias |
| **Test Date** | 11/15/2019 - 11/18-2019 |
| **Test ID** | **Input Values                  Expected Output Result                                Fix** |
| **1** | pw: 1212  vpw:1212   Account created           Access to Contact List N/A |
| **2** | pw: 1212  vpw:1312   Warning message        Denied access         Passwords must match |

**Appendix**

**A.1 Sign-In Page**

Imports System.IO

Public Class ContactViewing

    Dim fileName As String = "C:\Users\christwalker\Desktop\contacts.txt"

    'Log off Button

    Private Sub LogOff\_Click(sender As Object, e As EventArgs) Handles LogOff.Click

        Me.Hide()

        Signinpage.Show()

        Signinpage.incorrect.Visible = False

        Signinpage.incorrect2.Visible = False

        Signinpage.headerpasswordinput.Text = ""

        Signinpage.headerusernameinput.Text = ""

        Signinpage.usernameinput.Text = ""

        Signinpage.passwordinput.Text = ""

        Signinpage.verifypassword.Text = ""

    End Sub

    'Delete Button

    Private Sub Delete\_Click(sender As Object, e As EventArgs) Handles Delete.Click

        ContactList.Items.RemoveAt(ContactList.SelectedIndex)

    End Sub

    'Search Button

    Private Sub loginbutton\_Click(sender As Object, e As EventArgs) Handles search1.Click

        If File.Exists(fileName) Then

            Using MyReader As New Microsoft.VisualBasic.

                        FileIO.TextFieldParser(

                          fileName)

                MyReader.TextFieldType = FileIO.FieldType.Delimited

                MyReader.SetDelimiters(",")

                Dim currentRow As String()

                While Not MyReader.EndOfData

                    Try

                        currentRow = MyReader.ReadFields()

                        Dim currentField As String

                        For Each currentField In currentRow

                            If (nameinput.Text.Length > 0 And EmailInput.Text.Length > 0) Then

                                If currentField.Contains(nameinput.Text) OrElse currentField.Contains(EmailInput.Text) Then

                                    ContactList.Items.Add(currentRow(0) + "," + currentRow(1).ToString)

                                    nameinput.Text = ""

                                    EmailInput.Text = ""

                                    Exit Sub

                                ElseIf MyReader.EndOfData Then

                                    MsgBox(nameinput.Text + " could not be be found in the contact list")

                                    Exit Sub

                                End If

                            End If

                        Next

                    Catch ex As Microsoft.VisualBasic.

                  FileIO.MalformedLineException

                        MsgBox("contact list is corrupted, please call customer support")

                    End Try

                End While

            End Using

        Else

            MsgBox("Your contact list is empty")

        End If

    End Sub

    'Create Button

    Private Sub Button2\_Click(sender As Object, e As EventArgs) Handles Create.Click

        If nameinput.Text.Length > 0 And EmailInput.Text.Length > 0 Then

            Dim file As System.IO.StreamWriter

            If My.Computer.FileSystem.FileExists(fileName) Then

                file = My.Computer.FileSystem.OpenTextFileWriter(fileName, True)

                file.WriteLine(Environment.NewLine + nameinput.Text + "," + EmailInput.Text)

                file.Close()

                MsgBox(nameinput.Text + " was successfully added")

            Else

                file = My.Computer.FileSystem.OpenTextFileWriter(fileName, False)

                file.WriteLine(nameinput.Text + "," + EmailInput.Text)

                file.Close()

                MsgBox(nameinput.Text + " was successfully added")

            End If

            nameinput.Text = ""

            EmailInput.Text = ""

        End If

    End Sub

End Class

**A.2 Contact Viewing**

Imports System.IO

Public Class ContactViewing

    Dim fileName As String = "C:\Users\christwalker\Desktop\contacts.txt"

    'Log off Button

    Private Sub LogOff\_Click(sender As Object, e As EventArgs) Handles LogOff.Click

        Me.Hide()

        Signinpage.Show()

        Signinpage.incorrect.Visible = False

        Signinpage.incorrect2.Visible = False

        Signinpage.headerpasswordinput.Text = ""

        Signinpage.headerusernameinput.Text = ""

        Signinpage.usernameinput.Text = ""

        Signinpage.passwordinput.Text = ""

        Signinpage.verifypassword.Text = ""

    End Sub

    'Delete Button

    Private Sub Delete\_Click(sender As Object, e As EventArgs) Handles Delete.Click

        ContactList.Items.RemoveAt(ContactList.SelectedIndex)

    End Sub

    'Search Button

    Private Sub loginbutton\_Click(sender As Object, e As EventArgs) Handles search1.Click

        If File.Exists(fileName) Then

            Using MyReader As New Microsoft.VisualBasic.

                        FileIO.TextFieldParser(

                          fileName)

                MyReader.TextFieldType = FileIO.FieldType.Delimited

                MyReader.SetDelimiters(",")

                Dim currentRow As String()

                While Not MyReader.EndOfData

                    Try

                        currentRow = MyReader.ReadFields()

                        Dim currentField As String

                        For Each currentField In currentRow

                            If (nameinput.Text.Length > 0 And EmailInput.Text.Length > 0) Then

                                If currentField.Contains(nameinput.Text) OrElse currentField.Contains(EmailInput.Text) Then

                                    ContactList.Items.Add(currentRow(0) + "," + currentRow(1).ToString)

                                    nameinput.Text = ""

                                    EmailInput.Text = ""

                                    Exit Sub

                                ElseIf MyReader.EndOfData Then

                                    MsgBox(nameinput.Text + " could not be be found in the contact list")

                                    Exit Sub

                                End If

                            End If

                        Next

                    Catch ex As Microsoft.VisualBasic.

                  FileIO.MalformedLineException

                        MsgBox("contact list is corrupted, please call customer support")

                    End Try

                End While

            End Using

        Else

            MsgBox("Your contact list is empty")

        End If

    End Sub

    'Create Button

    Private Sub Button2\_Click(sender As Object, e As EventArgs) Handles Create.Click

        If nameinput.Text.Length > 0 And EmailInput.Text.Length > 0 Then

            Dim file As System.IO.StreamWriter

            If My.Computer.FileSystem.FileExists(fileName) Then

                file = My.Computer.FileSystem.OpenTextFileWriter(fileName, True)

                file.WriteLine(Environment.NewLine + nameinput.Text + "," + EmailInput.Text)

                file.Close()

                MsgBox(nameinput.Text + " was successfully added")

            Else

                file = My.Computer.FileSystem.OpenTextFileWriter(fileName, False)

                file.WriteLine(nameinput.Text + "," + EmailInput.Text)

                file.Close()

                MsgBox(nameinput.Text + " was successfully added")

            End If

            nameinput.Text = ""

            EmailInput.Text = ""

        End If

    End Sub

End Class

**A.3 SignIn Class Summary**

The SignIn class has two methods.  Both of which are activated by a click event listener. The first method is the Login method, which checks a .txt file to see if a username and password matches the given values. If the values can’t be found, an error message is displayed. The second method is the Create method. This method checks whether both the password textbox and verify-password textbox contain the same values. If they don’t have the same values, an error message is displayed. If they are the same, the method checks whether a file has been created to store the users and passwords.  If it doesn’t, the file is created, and the values are stored. If it has been created, the values are appended to the .txt file.

**A.4 ContactViewing Class Summary**

The ContactViewing class contains four methods. Each of these methods are activated by a click event listener. The first method is the LogOff method, which hides the current page, resets the values of the previous page, and shows the Login page. The second method is the Delete method. It deletes a selected row within our contact list. The third method is the Search method. It essentially parses a .txt file into rows and fields, which are then searched through until the end of the file is reached. If a specified value is found in a field, the row is displayed in the contact list. If nothing is found, a message is displayed saying that the values couldn’t be found. The last method is the Create method. It checks whether a .txt file exists and either appends a name and email address to an existing file or creates a new file and writes an email address and name to it.