MCOMD2SWE REPORT

Software engineering

Year 2 Computer Science

MCOMD2SWE | Software Engineering

Gordon Aiken | 24/03/2020

By

Gentian Gashi | Connor Dawkins | Tomas Porakis | Tovar Astradar

Table of Contents

[MCOMD2SWE REPORT 1](#_Toc35717239)

[Stories / Backlog / Estimation & Prioritisation 3](#_Toc35717240)

[Team Velocity Calculation 5](#_Toc35717241)

[Release Planning 5](#_Toc35717242)

[Sprint Backlogs & Burndown Charts of Three Sprints 5](#_Toc35717243)

[Fixed Date Planning 7](#_Toc35717244)

[Task Board 8](#_Toc35717245)

[User Manual & Code 8](#_Toc35717246)

[**References** 10](#_Toc35717247)

[**Appendix** 11](#_Toc35717248)

[Appendix A: Code 11](#_Toc35717249)

[**Sprint 1 Code:** 11](#_Toc35717250)

[**Sprint 2 Code:** 12](#_Toc35717251)

[**Sprint 3 Code:** 18](#_Toc35717252)

# Stories / Backlog / Estimation & Prioritisation

|  |  |
| --- | --- |
| Repository: | <https://github.com/CCCU-CTG/Interactive-Timetable-Map> |

Colour Code

|  |  |  |  |
| --- | --- | --- | --- |
| Sprint 1 | Sprint 2 | Sprint 3 | Future Sprints |
|  |  |  |  |

\*Most Important to Least Important\*

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **User Story** | **Acceptance Criteria** | **Estimated Points** |
| 1 | Create User Database | Must contain username | 1 |
| 2 | Create Module/Timetable Database | Must contain module name | 1 |
| 3 | Populate User Database | Must contain testable data | 1 |
| 4 | Populate Module/Timetable Database | Must contain testable data | 1 |
| 5 | Create Framework UI | UI must be capable of testing implemented functions | 1 |
| 6 | Link User Database to Program | Program must be able to read from database | 3 |
| 7 | Link Module/Timetable Database to Program | Program must be able to read from database | 3 |
| 8 | Encrypt User Passwords | Hash must not be stored in the same database as the encrypted password | 3 |
| 9 | Implement Login Functionality | User can only log in when username and password are both correct | 1 |
| 10 | Add Login UI | User must be able to enter their Username and Password | 1 |
| 11 | Link User Database to Program | Program must be able to read from database | 3 |
| 12 | Link Module/Timetable Database to Program | Program must be able to read from database | 3 |
| 13 | Merge RegForm With Login Form | Reduce code duplication | 1 |
| 14 | Update Login Code to Use Same Naming Scheme | Unify naming schemes for ease of programming | 1 |
| 15 | Fix Bug Preventing MainForm from Loading | Critical bug introduced by LoginForm must be fixed | 1 |
| 16 | Entering New Password for Username Overwrites Old Passwords | Critical security vulnerabilities must be removed | 1 |
| 17 | Make Login Draw from User Database .XML File | Login must correctly read from User Database | 2 |
| 18 | Test Database – Class Linkages | Must import information from the database correctly | 2 |
| 19 | Implement Timetable UI | Must show current week timetable | 2 |
| 20 | Design UI to Allow Admin to Edit Database | Must only be accessible by Admin Login | 3 |
| 21 | Refine UI | UI must be easy for users to navigate | 2 |
| 22 | Add a Help Button | Users should be able to easily find FAQ | 1 |
| 23 | Add a Guide for Program Usage | Users should be able to see the intended method of use for program | 1 |
| 24 | Implement Timetable Functionality | Users timetable should be correctly displayed when the user logs in | 3 |
| 25 | Implement Admin Functionality | Admin must be able to edit existing user/timetable database and add new users | 2 |
| 26 | Edit User SML/Database to Allow Admin Identification | Program must be able to identify admins | 2 |
| 27 | Delete Register Function from Login Page | Redundant features must be removed without causing bugs | 1 |
| 28 | Add Functionality To Logout Button | Logout function must return UI to neutral state with no bugs | 2 |
| 29 | Fix Bugs In Admin Functionality Forms | Admin Functionality must work completely without crashing | 2 |
| 30 | Link Google Maps to Timetable | Correct map must open when user clicks on room in Timetable | 2 |
| 31 | Fix & Refine Admin UI | Admin UI must have all necessary components for functionality | 2 |
| 32 | Fix Timetable Rows Bug | Timetable must work without major bugs | 2 |
| 33 | Test Timetable Feature | The correct timetable must be displayed | 2 |
| 34 | Test Admin User Editing Feature | User must be edited correctly, without error | 2 |
| 35 | Fix Incompatible Code In Admin Functionality - Technical | Admin must have all necessary functionalities required | 3 |
| 36 | Test Login Feature | Login and Logout features must be tested | 1 |
| 37 | Update Help Box | Help box should have information for all features | 1 |
| 38 | Create Tooltips For UI | Most UI elements should have a tooltip | 2 |
| 39 | Test Admin Database Editing | Admin Database Editing must work without critical issue | 1 |
| 40 | Implement Google Maps API |  | 5 |

# Team Velocity Calculation

Sprint 1 – 6 (Ideal work hours) \* 6 (Weeks) \* 4 (Team Members) / 3 = 48 / 3 (First Sprint) = 16

Sprint 1 Actual – 16

Sprint 2 – 6 (Ideal work hours) \* 6 (Weeks) \* 4 (Team Members) / 3 = 48 / 2 (Second sprint) = 24

Sprint 2 Actual - 26

Sprint 3 - 6 (Ideal work hours) \* 6 (Weeks) \* 4 (Team Members) / 3 = 48

Sprint 3 Actual - 30

# Release Planning

Sprint 1: Features

* Database system created and populated
* UI created and populated
* Password encryption implemented

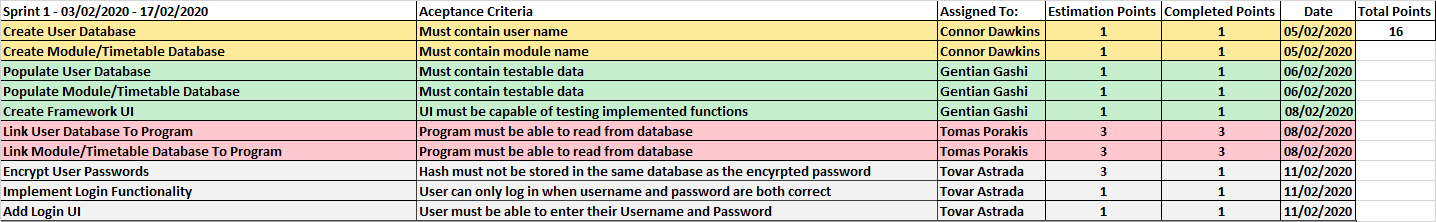
Sprint 2: Features

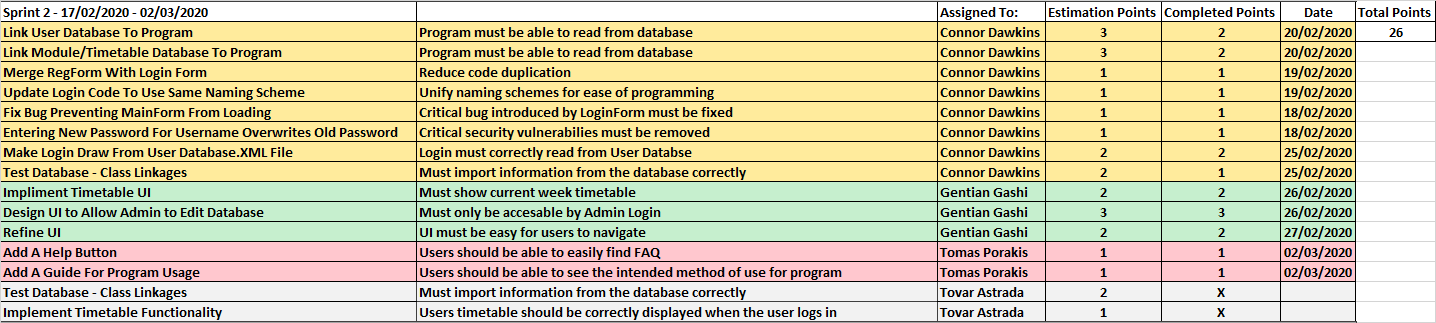
* Database system linked to program
* User login working
* UI linked to Database and User login

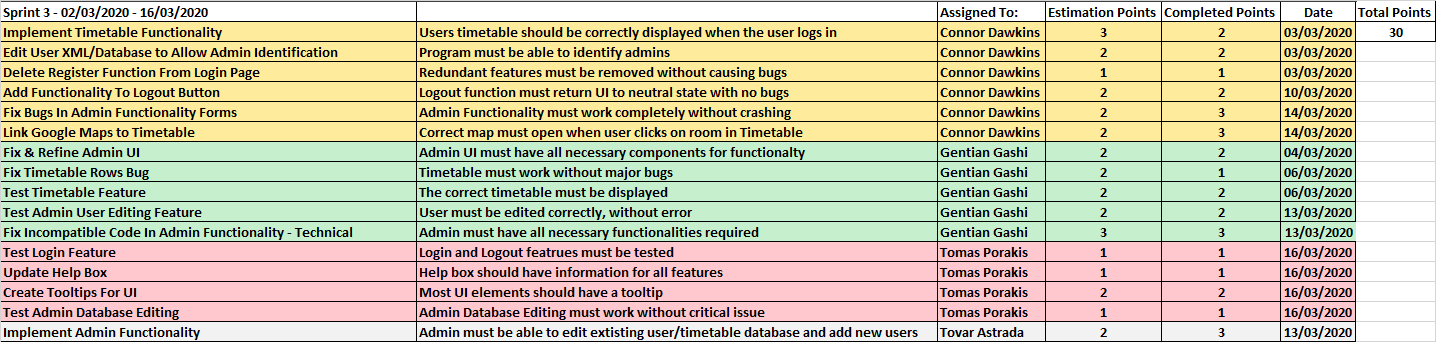
Sprint 3: Features

* When user logs in, the correct timetable is shown, and user can click on room names to be directed to a map of that location.
* Admin can edit User Database and Timetable Database in real time.

# Sprint Backlogs & Burndown Charts of Three Sprints







# Fixed Date Planning

Estimation of minimum stories each sprint

Total story points: 77

3 sprints

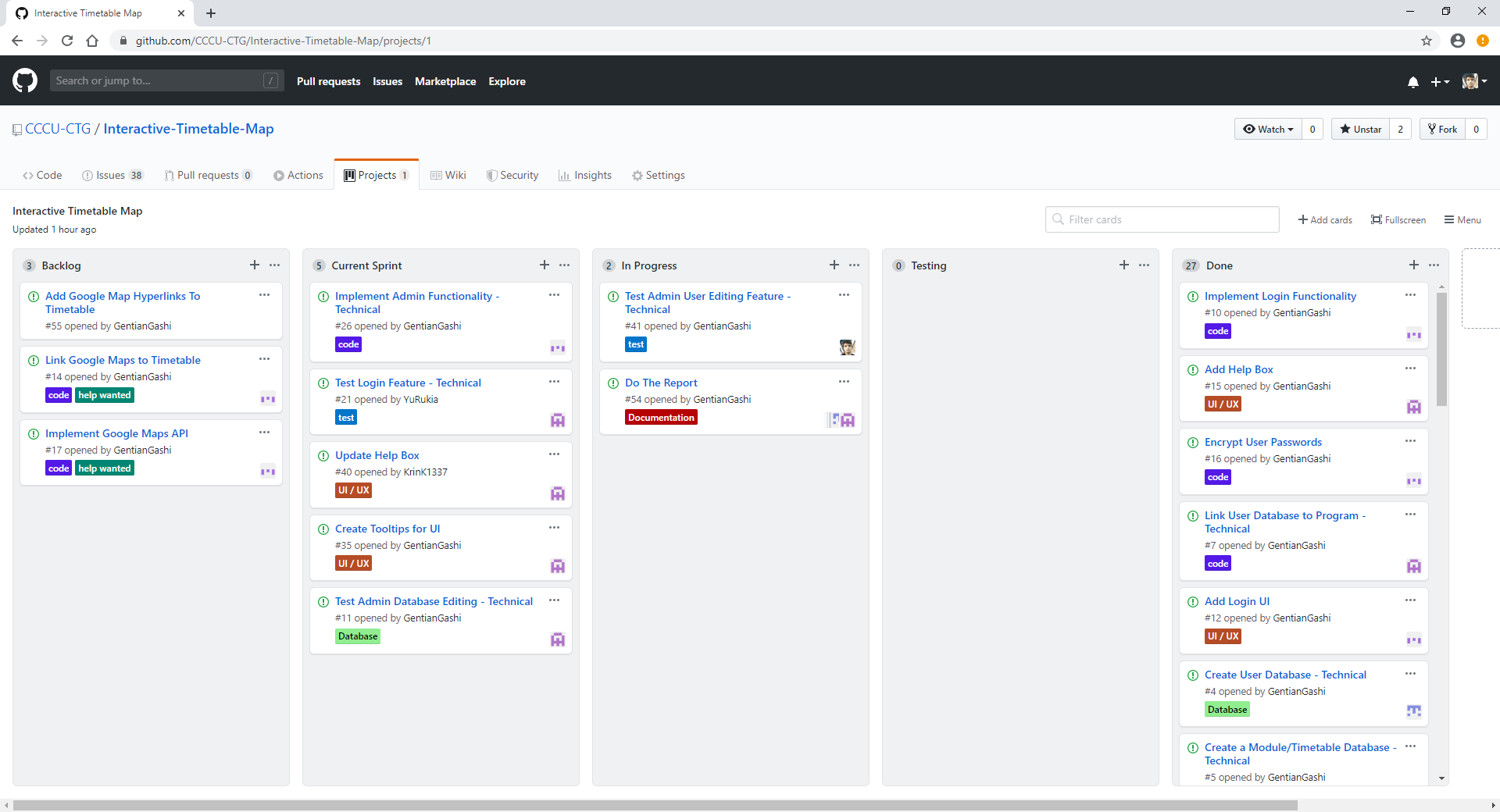
Range of Velocity: 20 - 25

Lowest (20 \* 3) = 60

Highest (25 \* 3) = 75

# Task Board

|  |  |
| --- | --- |
| Task Board: | <https://github.com/CCCU-CTG/Interactive-Timetable-Map/projects/1> |



# User Manual & Code

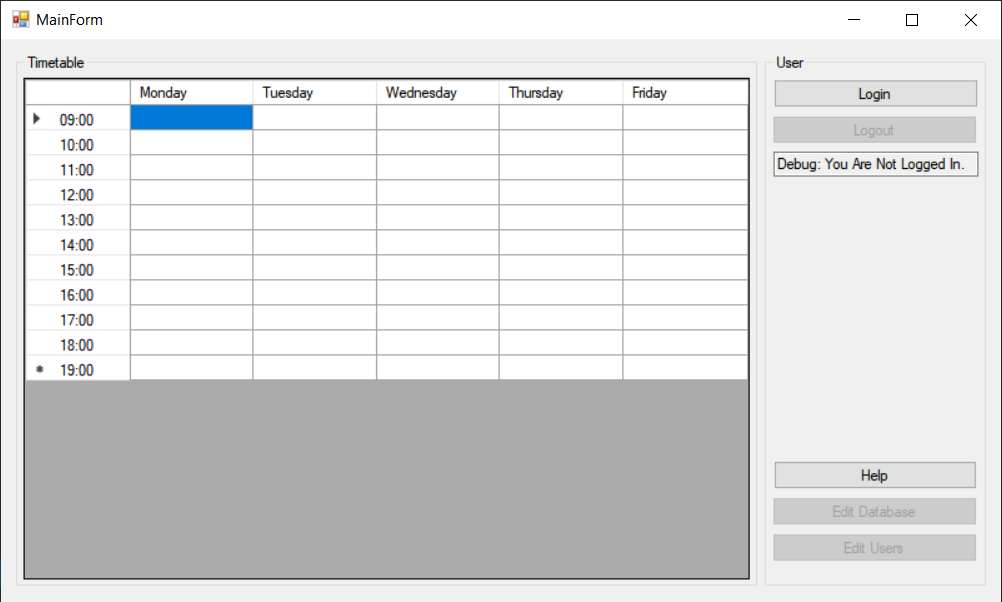
(Code in Appendix)

Sprint 1 User Manual:



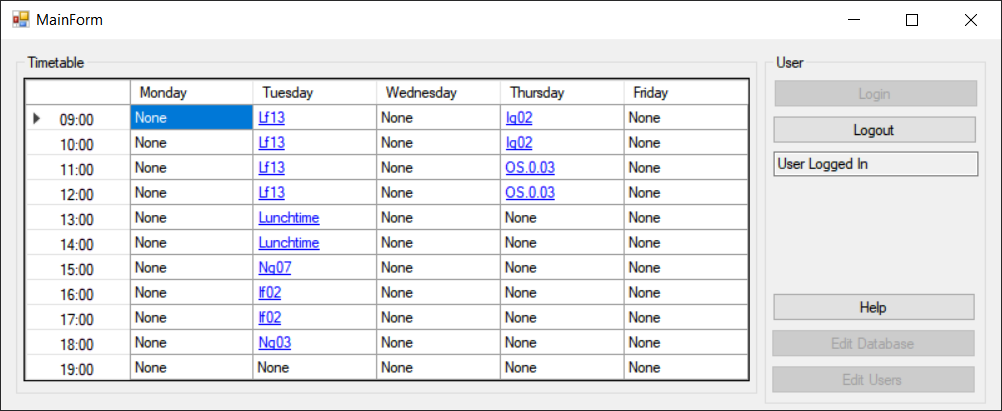
* To enter the program, type your username and password correctly. To register as a new user, type the username and password you would like to use.

Sprint 2 User Manual:



* You’ll automatically enter the program with an empty table, to login press the login button on the top-right side to view your relevant timetable. If you need additional help, press the help button located at the bottom-right screen which will open a window filled with tips and information.

Sprint 3 User Manual:



* As you open the program, an empty timetable will appear, with the option to login, you can either login as a user or admin. If you decide to log in as a user, you’ll be presented with your relevant timetable and the ability to click on your rooms which will open your browser to show you where it is. If you decide to login as the admin, you’ll unlock admin privileges such as the buttons located on the bottom-right to edit, add and remove users/timetables.

# **References**

256, E., 2020. *Encrypt & Decrypt Querystring Values Using AES 256*. [online] Stack Overflow. Available at: <https://stackoverflow.com/questions/28613831/encrypt-decrypt-querystring-values-using-aes-256> [Accessed 21 March 2020].

Aescrypt.com. 2020. *AES Crypt Users*. [online] Available at: <https://www.aescrypt.com/aescrypt_users.html> [Accessed 21 March 2020].

DataSet, A., 2020. *Add Link Label To Datagridview Cell Or Column Which Is Binded To Dataset*. [online] Stack Overflow. Available at: <https://stackoverflow.com/questions/13190590/add-link-label-to-datagridview-cell-or-column-which-is-binded-to-dataset>[Accessed 21 March 2020].

URL, P., Axel, A., Axel, A., Flynn, J., Beckers, J. and G&#243;rny, M., 2020. *Passing Base64 Encoded Strings In URL*. [online] Stack Overflow. Available at: <https://stackoverflow.com/questions/1374753/passing-base64-encoded-strings-in-url>[Accessed 21 March 2020].

# **Appendix**

## Appendix A: Code

### **Sprint 1 Code:**

class AesCryp

{

public static string IV = "qo1lc3sjd8zpt9cx"; // 16 chars = 128 bytes

public static string Key = "ow7dxys8glfor9tnc2ansdfo1etkfjcv"; // 32 chars = 256 bytes

public static string Encrypt(string decrypted)

{

byte[] textbytes = ASCIIEncoding.ASCII.GetBytes(decrypted);

AesCryptoServiceProvider encdec = new AesCryptoServiceProvider();

encdec.BlockSize = 128;

encdec.KeySize = 256;

encdec.Key = ASCIIEncoding.ASCII.GetBytes(Key);

encdec.IV = ASCIIEncoding.ASCII.GetBytes(IV);

encdec.Padding = PaddingMode.PKCS7;

encdec.Mode = CipherMode.CBC;

ICryptoTransform icrypt = encdec.CreateEncryptor(encdec.Key, encdec.IV);

byte[] enc = icrypt.TransformFinalBlock(textbytes, 0, textbytes.Length);

icrypt.Dispose();

return Convert.ToBase64String(enc);

}

public static string Decrypt(string encrypted)

{

byte[] encbytes = Convert.FromBase64String(encrypted);

AesCryptoServiceProvider encdec = new AesCryptoServiceProvider();

encdec.BlockSize = 128;

encdec.KeySize = 256;

encdec.Key = ASCIIEncoding.ASCII.GetBytes(Key);

encdec.IV = ASCIIEncoding.ASCII.GetBytes(IV);

encdec.Padding = PaddingMode.PKCS7;

encdec.Mode = CipherMode.CBC;

ICryptoTransform icrypt = encdec.CreateDecryptor(encdec.Key, encdec.IV);

byte[] dec = icrypt.TransformFinalBlock(encbytes, 0, encbytes.Length);

icrypt.Dispose();

return ASCIIEncoding.ASCII.GetString(dec);

}

}

}

### **Sprint 2 Code:**

public partial class MainForm : Form

{

// List containing Module Timetable Class objects + User Class objects

List<User> usersList = new List<User>();

User currentUser;

List<Module\_Timetable> moduleTimetableList = new List<Module\_Timetable>();

// DEBUG, is has the login function worked?

public static bool loggedIn = false;

public static bool adminLoggedIn = false;

public MainForm()

{

InitializeComponent();

}

private void MainForm\_Load(object sender, EventArgs e)

{

UserXMLReader();

ModuleTimetableXMLReader();

LoginCheck(loggedIn, adminLoggedIn, currentUser);

editDatabaseButton.Enabled = false;

editUsersButton.Enabled = false;

}

private void UserXMLReader()

{

XmlDocument Users = new XmlDocument();

// List containing user information

List<string> userID = new List<string>();

List<string> name = new List<string>();

List<string> surname = new List<string>();

List<string> username = new List<string>();

List<string> password = new List<string>();

List<string> group = new List<string>();

List<bool> admin = new List<bool>();

string directory = Directory.GetCurrentDirectory() + "/Databases";

Users.Load(directory + @"\User\_Database.xml");

foreach(XmlNode uID in Users.DocumentElement.SelectNodes("//ID"))

{

userID.Add(uID.InnerText);

}

foreach (XmlNode uName in Users.DocumentElement.SelectNodes("//Name"))

{

name.Add(uName.InnerText);

}

foreach (XmlNode sName in Users.DocumentElement.SelectNodes("//Surname"))

{

surname.Add(sName.InnerText);

}

foreach (XmlNode userName in Users.DocumentElement.SelectNodes("//Username"))

{

username.Add(userName.InnerText);

}

foreach (XmlNode pass in Users.DocumentElement.SelectNodes("//Password"))

{

password.Add(pass.InnerText);

}

foreach (XmlNode uGroup in Users.DocumentElement.SelectNodes("//Group"))

{

group.Add(uGroup.InnerText);

}

foreach (XmlNode Admin in Users.DocumentElement.SelectNodes("//Admin"))

{

if(Admin.InnerText == "Y") { admin.Add(true); }

else { admin.Add(false); }

}

for (int i = 0; i < userID.Count; i++)

{

usersList.Add(new User(userID[i], name[i], surname[i], username[i], password[i], group[i], admin[i]));

}

}

private void ModuleTimetableXMLReader()

{

XmlDocument Modules = new XmlDocument();

// Lists containing Module names + schedules on each day

List<string> modulesList = new List<string>();

List<string> tempMonday = new List<string>();

List<string> tempTuesday = new List<string>();

List<string> tempWednesday = new List<string>();

List<string> tempThursday = new List<string>();

List<string> tempFriday = new List<string>();

string directory = Directory.GetCurrentDirectory() + "/Databases";

Modules.Load(directory + @"\Module\_Timetable\_Database.xml");

foreach (XmlNode moduleNames in Modules.DocumentElement.SelectNodes("//Name"))

{

modulesList.Add(moduleNames.InnerText);

}

for (int i = 0; i < modulesList.Count; i++)

{

tempMonday.Clear(); tempTuesday.Clear(); tempWednesday.Clear(); tempThursday.Clear(); tempFriday.Clear();

foreach (XmlNode monday in Modules.DocumentElement.SelectNodes(modulesList[i] + "/Monday/Time"))

{

tempMonday.Add(monday.InnerText);

}

foreach (XmlNode tuesday in Modules.DocumentElement.SelectNodes(modulesList[i] + "/Tuesday/Time"))

{

tempTuesday.Add(tuesday.InnerText);

}

foreach (XmlNode wednesday in Modules.DocumentElement.SelectNodes(modulesList[i] + "/Wednesday/Time"))

{

tempWednesday.Add(wednesday.InnerText);

}

foreach (XmlNode thursday in Modules.DocumentElement.SelectNodes(modulesList[i] + "/Thursday/Time"))

{

tempThursday.Add(thursday.InnerText);

}

foreach (XmlNode friday in Modules.DocumentElement.SelectNodes(modulesList[i] + "/Friday/Time"))

{

tempFriday.Add(friday.InnerText);

}

moduleTimetableList.Add(new Module\_Timetable(modulesList[i], tempMonday, tempTuesday, tempWednesday, tempThursday, tempFriday));

}

}

/// Opens Login Form

private void loginButton\_Click(object sender, EventArgs e)

{

// Checks to see if a Login Form is already open, if so, close that and open a new one

if (Application.OpenForms.OfType<loginForm>().Count() > 0) { Application.OpenForms.OfType<loginForm>().First().Close(); }

var loginForm = new loginForm(usersList);

loginForm.Show();

}

/// For testing purposes, tells you if you are logged in or not

public void LoginCheck(bool loggedInTemp, bool adminLoggedInTemp, User currentUserTemp)

{

loggedIn = loggedInTemp;

adminLoggedIn = adminLoggedInTemp;

if (loggedIn) {loginButton.Enabled = false; logoutButton.Enabled = true; testingLoggedInTextBox.Text = "User Logged In"; }

else { loginButton.Enabled = true; logoutButton.Enabled = false; testingLoggedInTextBox.Text = "User Logged Out"; }

if (adminLoggedIn) { testingLoggedInTextBox.Text = "Admin Logged In"; editDatabaseButton.Enabled = true; editUsersButton.Enabled = true; }

else { editDatabaseButton.Enabled = false; editUsersButton.Enabled = false; }

currentUser = currentUserTemp;

TimetableDataGridLoad();

}

private void TimetableDataGridLoad()

{

timetableDataGrid.AllowUserToAddRows = false;

int selectedModule = 0;

string[] timetableTimes = new string[] { "09:00", "10:00", "11:00", "12:00", "13:00", "14:00", "15:00", "16:00", "17:00", "18:00", "19:00"};

if (loggedIn)

{

for (int i = 0; i < moduleTimetableList.Count; i++)

{

if (currentUser.GetModule == moduleTimetableList[i].GetModuleName) { selectedModule = i; }

}

//Resets table

timetableDataGrid.Rows.Clear();

//Creates 11 empty rows

for (int i = 0; i < timetableTimes.Length; i++)

{

timetableDataGrid.Rows.Add

(moduleTimetableList[selectedModule].GetMonday[i],

moduleTimetableList[selectedModule].GetTuesday[i],

moduleTimetableList[selectedModule].GetWednesday[i],

moduleTimetableList[selectedModule].GetThursday[i],

moduleTimetableList[selectedModule].GetFriday[i]);

timetableDataGrid.Rows[i].HeaderCell.Value = timetableTimes[i];

}

foreach (DataGridViewRow r in timetableDataGrid.Rows)

{

DataGridViewLinkCell Monday = new DataGridViewLinkCell();

if(r.Cells[0].Value.ToString() != "None")

{

Monday.Value = r.Cells[0].Value;

timetableDataGrid[0, r.Index] = Monday;

}

DataGridViewLinkCell Tuesday = new DataGridViewLinkCell();

if (r.Cells[1].Value.ToString() != "None")

{

Tuesday.Value = r.Cells[1].Value;

timetableDataGrid[1, r.Index] = Tuesday;

}

DataGridViewLinkCell Wednesday = new DataGridViewLinkCell();

if (r.Cells[2].Value.ToString() != "None")

{

Wednesday.Value = r.Cells[2].Value;

timetableDataGrid[2, r.Index] = Wednesday;

}

DataGridViewLinkCell Thursday = new DataGridViewLinkCell();

if (r.Cells[3].Value.ToString() != "None")

{

Thursday.Value = r.Cells[3].Value;

timetableDataGrid[3, r.Index] = Thursday;

}

DataGridViewLinkCell Friday = new DataGridViewLinkCell();

if (r.Cells[4].Value.ToString() != "None")

{

Friday.Value = r.Cells[4].Value;

timetableDataGrid[4, r.Index] = Friday;

}

}

}

else

{

timetableDataGrid.Rows.Clear();

for (int i = 0; i < timetableTimes.Length; i++) { timetableDataGrid.Rows.Add(); timetableDataGrid.Rows[i].HeaderCell.Value = timetableTimes[i]; }

//Displays time for each cell

}

}

private void timetableDataGrid\_CellMouseClick(object sender, DataGridViewCellEventArgs e)

{

switch (timetableDataGrid.Rows[e.RowIndex].Cells[e.ColumnIndex].Value.ToString())

{

case "Lf13":

Process.Start("https://www.canterbury.ac.uk/campusmaps/#/2ea30af3d92c45ee851a1e01/details/fa0c4599c02c468f90b2613a/");

break;

case "Ng07":

Process.Start("https://www.canterbury.ac.uk/campusmaps/#/2ea30af3d92c45ee851a1e01/details/3f97499b02244a6b8320852e/");

break;

case "If02":

Process.Start("https://www.canterbury.ac.uk/campusmaps/#/2ea30af3d92c45ee851a1e01/details/0e32e2d035c84cfd8f3ce7c7/");

break;

case "Ng03":

Process.Start("https://www.canterbury.ac.uk/campusmaps/#/2ea30af3d92c45ee851a1e01/details/0b8dbbf97e4e44b3af2780ee/");

break;

case "Ig02":

Process.Start("https://www.canterbury.ac.uk/campusmaps/#/2ea30af3d92c45ee851a1e01/details/2ae42057860e4f719e06dc9a/");

break;

case "OS.0.03":

Process.Start("https://www.canterbury.ac.uk/campusmaps/#/2ea30af3d92c45ee851a1e01/details/89418a349156467b941e428e/");

break;

case "Pg06":

Process.Start("https://www.canterbury.ac.uk/campusmaps/#/2ea30af3d92c45ee851a1e01/details/1a6730c77c8548f78b3f389c/");

break;

case "Lg27":

Process.Start("https://www.canterbury.ac.uk/campusmaps/#/2ea30af3d92c45ee851a1e01/details/18587695b38a4d8380e2ccd9/");

break;

case "If03":

Process.Start("https://www.canterbury.ac.uk/campusmaps/#/2ea30af3d92c45ee851a1e01/details/86973e61c0d34a85b732371f/");

break;

case "Lunchtime":

Process.Start("https://www.canterbury.ac.uk/campusmaps/#/2ea30af3d92c45ee851a1e01/details/c06b8595bcbd43bba93b340d/");

break;

}

}

private void EditDatabaseButton\_Click(object sender, EventArgs e)

{

if (Application.OpenForms.OfType<EditDatabaseForm>().Count() > 0) { Application.OpenForms.OfType<EditDatabaseForm>().First().Close(); }

var DataForm = new EditDatabaseForm();

DataForm.ShowDialog();

}

// reload the timetable from the file

public void UpdateDataForm()

{

moduleTimetableList.Clear();

ModuleTimetableXMLReader();

TimetableDataGridLoad();

}

private void EditUsersButton\_Click(object sender, EventArgs e)

{

// open users form and use a callback when closed to update the users list

if (Application.OpenForms.OfType<EditUsersForm>().Count() > 0) { Application.OpenForms.OfType<EditUsersForm>().First().Close(); }

var UserForm = new EditUsersForm(usersList, this, currentUser);

UserForm.ShowDialog();

}

// reload the users from the file

public void UpdateUserForm()

{

// called when user details are changed

// clear the list and re read the details

usersList.Clear();

UserXMLReader();

}

private void helpButton\_Click(object sender, EventArgs e)

{

if (Application.OpenForms.OfType<HelpForm>().Count() > 0) { Application.OpenForms.OfType<HelpForm>().First().Close(); }

var HelpForm = new HelpForm();

HelpForm.Show();

}

private void LogoutButton\_Click(object sender, EventArgs e)

{

loggedIn = false;

adminLoggedIn = false;

currentUser = null;

LoginCheck(loggedIn, adminLoggedIn, currentUser);

}

}

}

### **Sprint 3 Code:**

namespace Interactive\_Timetable\_Map

{

public partial class EditDatabaseForm : Form

{

XmlDocument xtr = new XmlDocument();

string directory = Directory.GetCurrentDirectory() + "/Databases";

// global holding variables

List<string> monday = new List<string>();

List<string> tuesday = new List<string>();

List<string> wednesday = new List<string>();

List<string> thursday = new List<string>();

List<string> friday = new List<string>();

string selectedTimeSlot = "";

int groupIndex = 0;

string timeSlotDate;

string selectedModule;

public EditDatabaseForm()

{

InitializeComponent();

xtr.Load(directory + @"\Module\_Timetable\_Database.xml");

refreshDropDowns();

}

//refresh all drop downs to match new data added/removed

private void refreshDropDowns()

{

xtr.Load(directory + @"\Module\_Timetable\_Database.xml");

monday.Clear(); tuesday.Clear(); wednesday.Clear(); thursday.Clear(); friday.Clear(); moduleNameComboBox.Items.Clear();

comboBoxMon.Text = ""; comboBoxTues.Text = ""; comboBoxWeds.Text = ""; comboBoxThurs.Text = ""; comboBoxFri.Text = "";

// load in all store loops of modules with corresponding dates

// create all counter variables

int i = 0; int time = 9; int tc = 0; string dt;

foreach (XmlNode uID in xtr.DocumentElement.SelectNodes("//Monday"))

{

time = 9; tc = 0;

foreach (XmlNode slot in uID)

{

dt = string.Format("{0:00.00}", time + tc);

dt += ": ";

monday.Insert(i, dt + slot.InnerText); i++; tc++;

}

}

i = 0; tc = 0;

foreach (XmlNode uID in xtr.DocumentElement.SelectNodes("//Tuesday"))

{

time = 9; tc = 0;

foreach (XmlNode slot in uID)

{

dt = string.Format("{0:00.00}", time + tc);

dt += ": ";

tuesday.Insert(i, dt + slot.InnerText); i++; tc++;

}

}

i = 0; tc = 0;

foreach (XmlNode uID in xtr.DocumentElement.SelectNodes("//Wednesday"))

{

time = 9; tc = 0;

foreach (XmlNode slot in uID)

{

dt = string.Format("{0:00.00}", time + tc);

dt += ": ";

wednesday.Insert(i, dt + slot.InnerText); i++; tc++;

}

}

i = 0; tc = 0;

foreach (XmlNode uID in xtr.DocumentElement.SelectNodes("//Thursday"))

{

time = 9; tc = 0;

foreach (XmlNode slot in uID)

{

dt = string.Format("{0:00.00}", time + tc);

dt += ": ";

thursday.Insert(i, dt + slot.InnerText); i++; tc++;

}

}

i = 0; tc = 0;

foreach (XmlNode uID in xtr.DocumentElement.SelectNodes("//Friday"))

{

time = 9; tc = 0;

foreach (XmlNode slot in uID)

{

dt = string.Format("{0:00.00}", time + tc);

dt += ": ";

friday.Insert(i, dt + slot.InnerText); i++; tc++;

}

}

i = 0;

moduleNameComboBox.Items.Clear();

foreach (XmlNode uID in xtr.DocumentElement.SelectNodes("//Name"))

{

moduleNameComboBox.Items.Insert(i, uID.InnerText);

i++;

}

comboBoxMon.Items.Clear();

comboBoxTues.Items.Clear();

comboBoxWeds.Items.Clear();

comboBoxThurs.Items.Clear();

comboBoxFri.Items.Clear();

// add data to boxes

selectedModule = "";

selectedTimeSlot = "";

}

private void BackButton\_Click(object sender, EventArgs e)

{

Application.OpenForms.OfType<MainForm>().First().Show();

this.Close();

}

private void addModuleButton\_Click(object sender, EventArgs e)

{

bool valid = true;

string newModuleName = Interaction.InputBox("Enter the name of the new module: ", "New Module Name", "New Module Name");

// if module allready exists dont create and show warning message

foreach (XmlNode uID in xtr.DocumentElement.SelectNodes("//Name"))

{

if (uID.InnerText == newModuleName) { valid = false; }

}

if (!valid) { MessageBox.Show("Module " + newModuleName + "Already exists", "Module Allready Exists", MessageBoxButtons.OK, MessageBoxIcon.Warning); }

else

{

//create module and set it blank

XmlNode headerNode = xtr.DocumentElement.SelectSingleNode("//Modules");

XmlNode parentNode = xtr.CreateElement(newModuleName);

headerNode.AppendChild(parentNode);

XmlNode nameNode = xtr.CreateElement("Name");

nameNode.InnerText = newModuleName;

parentNode.AppendChild(nameNode);

// set all dates and times to None

XmlNode DateNode = xtr.CreateElement("Monday");

parentNode.AppendChild(DateNode);

// create all time slots and attatch to corresponding date

for (int i = 0; i < 11; i++)

{

XmlNode timeNode = xtr.CreateElement("Time");

timeNode.InnerText = "None";

DateNode.AppendChild(timeNode);

}

// repeat for tues-fri

DateNode = xtr.CreateElement("Tuesday");

parentNode.AppendChild(DateNode);

for (int i = 0; i < 11; i++) { XmlNode timeNode = xtr.CreateElement("Time"); timeNode.InnerText = "None"; DateNode.AppendChild(timeNode); }

DateNode = xtr.CreateElement("Wednesday");

parentNode.AppendChild(DateNode);

for (int i = 0; i < 11; i++) { XmlNode timeNode = xtr.CreateElement("Time"); timeNode.InnerText = "None"; DateNode.AppendChild(timeNode); }

DateNode = xtr.CreateElement("Thursday");

parentNode.AppendChild(DateNode);

for (int i = 0; i < 11; i++) { XmlNode timeNode = xtr.CreateElement("Time"); timeNode.InnerText = "None"; DateNode.AppendChild(timeNode); }

DateNode = xtr.CreateElement("Friday");

parentNode.AppendChild(DateNode);

for (int i = 0; i < 11; i++) { XmlNode timeNode = xtr.CreateElement("Time"); timeNode.InnerText = "None"; DateNode.AppendChild(timeNode); }

xtr.Save(directory + @"\Module\_Timetable\_Database.xml");

MessageBox.Show("Module: " + newModuleName + "succesfully added", "Module Added", MessageBoxButtons.OK, MessageBoxIcon.Information);

refreshDropDowns();

}

}

private void moduleNameComboBox\_SelectedIndexChanged(object sender, EventArgs e)

{

// feed data from dateLists into respective drop downs

// is not separated by module

// 11 per module so can be found at pos

// ( index \* 11) + i

// reset all drop downs

comboBoxMon.Items.Clear();

comboBoxTues.Items.Clear();

comboBoxWeds.Items.Clear();

comboBoxThurs.Items.Clear();

comboBoxFri.Items.Clear();

// add data to boxes

groupIndex = moduleNameComboBox.SelectedIndex \* 11;

for (int i = 0; i < 11; i++)

{

comboBoxMon.Items.Add(monday[(groupIndex + i)]);

comboBoxTues.Items.Add(tuesday[(groupIndex + i)]);

comboBoxWeds.Items.Add(wednesday[(groupIndex + i)]);

comboBoxThurs.Items.Add(thursday[(groupIndex + i)]);

comboBoxFri.Items.Add(friday[(groupIndex + i)]);

}

selectedModule = moduleNameComboBox.Text;

}

private void buttonTimeSlot\_Click(object sender, EventArgs e)

{

if (selectedTimeSlot == "") { MessageBox.Show("No time slot selected", "No Time Slot Selected", MessageBoxButtons.OK, MessageBoxIcon.Warning); }

else

{

// propmt the user to add a new time for the slot and update the list

string newTimeSlot = Interaction.InputBox("Enter the new content for timeslot: " + timeSlotDate + ": " + selectedTimeSlot, "Alter Time Slot", "Alter Time Slot");

string datestub = selectedTimeSlot.Substring(0, 6);

int writeIndex = 0;

if (timeSlotDate == "Monday")

{

monday[(groupIndex + comboBoxMon.SelectedIndex)] = datestub + newTimeSlot;

writeIndex = comboBoxMon.SelectedIndex;

}

if (timeSlotDate == "Tuesday")

{

tuesday[(groupIndex + comboBoxTues.SelectedIndex)] = datestub + newTimeSlot;

writeIndex = comboBoxTues.SelectedIndex;

}

if (timeSlotDate == "Wednesday")

{

wednesday[(groupIndex + comboBoxWeds.SelectedIndex)] = datestub + newTimeSlot;

writeIndex = comboBoxWeds.SelectedIndex;

}

if (timeSlotDate == "Thursday")

{

thursday[(groupIndex + comboBoxThurs.SelectedIndex)] = datestub + newTimeSlot;

writeIndex = comboBoxThurs.SelectedIndex;

}

if (timeSlotDate == "Friday")

{

friday[(groupIndex + comboBoxFri.SelectedIndex)] = datestub + newTimeSlot;

writeIndex = comboBoxFri.SelectedIndex;

}

// use set variables from dropdowns to build path

XmlNode slot = xtr.SelectNodes("//" + selectedModule + "/" + timeSlotDate + "/Time")[writeIndex];

// alter text and save to path, providing a callback message for clarity

slot.InnerText = newTimeSlot;

xtr.Save(directory + @"\Module\_Timetable\_Database.xml");

MessageBox.Show("timeslot changed", "Timeslot Changed", MessageBoxButtons.OK, MessageBoxIcon.Information);

Application.OpenForms.OfType<MainForm>().First().UpdateDataForm();

refreshDropDowns();

}

}

// when any combo box is selected change the drop down path variable and the date in the information box

private void comboBoxMon\_SelectedIndexChanged(object sender, EventArgs e)

{

selectedTimeSlot = monday[(groupIndex + comboBoxMon.SelectedIndex)];

textBoxTimeSlot.Text = "Monday:\n " + selectedTimeSlot;

timeSlotDate = "Monday";

}

private void comboBoxTues\_SelectedIndexChanged(object sender, EventArgs e)

{

selectedTimeSlot = tuesday[(groupIndex + comboBoxTues.SelectedIndex)];

textBoxTimeSlot.Text = "Tuesday:\n " + selectedTimeSlot;

timeSlotDate = "Tuesday";

}

private void comboBoxWeds\_SelectedIndexChanged(object sender, EventArgs e)

{

selectedTimeSlot = wednesday[(groupIndex + comboBoxWeds.SelectedIndex)];

textBoxTimeSlot.Text = "Wednesday:\n " + selectedTimeSlot;

timeSlotDate = "Wednesday";

}

private void comboBoxThurs\_SelectedIndexChanged(object sender, EventArgs e)

{

selectedTimeSlot = thursday[(groupIndex + comboBoxThurs.SelectedIndex)];

textBoxTimeSlot.Text = "Thursday:\n " + selectedTimeSlot;

timeSlotDate = "Thursday";

}

private void comboBoxFri\_SelectedIndexChanged(object sender, EventArgs e)

{

selectedTimeSlot = monday[(groupIndex + comboBoxFri.SelectedIndex)];

textBoxTimeSlot.Text = "Friday:\n " + selectedTimeSlot;

timeSlotDate = "Friday";

}

private void removeModuleButton\_Click(object sender, EventArgs e)

{

string moduleToDelete = Interaction.InputBox("Enter the name of the module to remove, note any users in this module will need to be manually changed to other modules ", "Remove Module", "Remove Module");

DialogResult removeModule = MessageBox.Show("Do you want to remove the currently selected module? this action cannot be undone.", "Remove User", MessageBoxButtons.YesNo);

if (removeModule == DialogResult.Yes)

{

foreach (XmlNode module in xtr.DocumentElement.SelectNodes("//Modules/\*"))

{

if (moduleToDelete == module.FirstChild.InnerText)

{

XmlNode parent = module.ParentNode;

//userName.ParentNode.RemoveAll();

parent.RemoveChild(module);

xtr.Save(directory + @"\Module\_Timetable\_Database.xml");

}

}

MessageBox.Show("User: " + moduleToDelete + "succesfully removed", "User Removed", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

else if (removeModule == DialogResult.No) { }

refreshDropDowns();

}

}

}

namespace Interactive\_Timetable\_Map

{

public partial class EditUsersForm : Form

{

List<User> user = new List<User>();

User currentUser;

XmlDocument xtr = new XmlDocument();

string directory = Directory.GetCurrentDirectory() + "/Databases";

//XmlTextReader xtr = new XmlTextReader(@"..\Debug\Databases\User\_Database.xml");

// List containing user information

List<string> userID = new List<string>();

List<string> name = new List<string>();

List<string> surname = new List<string>();

List<string> username = new List<string>();

List<string> password = new List<string>();

List<string> group = new List<string>();

List<bool> admin = new List<bool>();

public string newUsername = null;

public string newPassword = null;

public string newFirstname = null;

public string newSurname = null;

public string newGroup = null;

public string newAdmin = null;

public string newID = null;

public bool sentUser = false;

User loginUser;

public EditUsersForm(List<User> usersList, MainForm parent, User loggedInAs)

{

xtr.Load(directory + @"\User\_Database.xml");

MainForm parentForm = parent;

loginUser = loggedInAs;

user = usersList;

InitializeComponent();

CreateUsers();

AddUsersUI();

}

public void AddUsersUI()

{

int i = 0;

comboBoxID.Items.Clear();

foreach (User user in user)

{

comboBoxID.Items.Insert(i, user.GetPatientID);

i++;

}

}

private void CreateUsers()

{

for (int i = 0; i < user.Count; i++)

{

string dir = user[i].GetUsername;

// Checks to see if the username already exists, if so, abort registration

if (!Directory.Exists("data\\" + dir))

{

Directory.CreateDirectory("data\\" + dir);

var sw = new StreamWriter("data\\" + dir + "\\data.ls");

string encryptedusername = AesCryp.Encrypt(user[i].GetUsername);

string encryptedpassword = AesCryp.Encrypt(user[i].GetPassword);

sw.WriteLine(encryptedusername);

sw.WriteLine(encryptedpassword);

sw.Close();

}

}

}

private void BackButton\_Click(object sender, EventArgs e)

{

Application.OpenForms.OfType<MainForm>().First().Show();

this.Close();

}

private void addUserButton\_Click(object sender, EventArgs e)

{

sentUser = false;

// create new user form and add call back function to it

if (Application.OpenForms.OfType<AddUser>().Count() > 0) { Application.OpenForms.OfType<AddUser>().First().Close(); }

var adduserForm = new AddUser(this, user);

adduserForm.ShowDialog();

}

public void AddUserFunction()

{

if (sentUser)

{

XmlNode headerNode = xtr.DocumentElement.SelectSingleNode("//Users");

XmlNode finalChild = xtr.DocumentElement.LastChild;

XmlNode parentNode = xtr.CreateElement("User");

headerNode.AppendChild(parentNode);

XmlNode infoNode = xtr.CreateElement("ID");

infoNode.InnerText = newID;

parentNode.AppendChild(infoNode);

infoNode = xtr.CreateElement("Name");

infoNode.InnerText = newFirstname;

parentNode.AppendChild(infoNode);

infoNode = xtr.CreateElement("Surname");

infoNode.InnerText = newSurname;

parentNode.AppendChild(infoNode);

infoNode = xtr.CreateElement("Username");

infoNode.InnerText = newUsername;

parentNode.AppendChild(infoNode);

infoNode = xtr.CreateElement("Password");

infoNode.InnerText = newPassword;

parentNode.AppendChild(infoNode);

infoNode = xtr.CreateElement("Group");

infoNode.InnerText = newGroup;

parentNode.AppendChild(infoNode);

infoNode = xtr.CreateElement("Admin");

infoNode.InnerText = newAdmin;

parentNode.AppendChild(infoNode);

xtr.Save(directory + @"\User\_Database.xml");

MessageBox.Show("User: " + newUsername + "succesfully added", "User Added", MessageBoxButtons.OK, MessageBoxIcon.Information);

Application.OpenForms.OfType<MainForm>().First().UpdateUserForm();

}

}

private void removeUserButton\_Click(object sender, EventArgs e)

{

if(currentUser != null)

{

if (currentUser.GetUsername == loginUser.GetUsername) { MessageBox.Show("You cannot remove yourself from the users list", "User Cannot Be Removed", MessageBoxButtons.OK, MessageBoxIcon.Warning); }

else

{

DialogResult removeUser = MessageBox.Show("Do you want to remove the currently selected user? this action cannot be undone.", "Remove User", MessageBoxButtons.YesNo);

if (removeUser == DialogResult.Yes)

{

string removeName = currentUser.GetUsername;

foreach (XmlNode userName in xtr.DocumentElement.SelectNodes("//Username"))

{

if (removeName == userName.FirstChild.Value)

{

XmlNode parent = userName.ParentNode;

//userName.ParentNode.RemoveAll();

parent.ParentNode.RemoveChild(parent);

xtr.Save(directory + @"\User\_Database.xml");

}

}

MessageBox.Show("User: " + removeName + "succesfully removed", "User Removed", MessageBoxButtons.OK, MessageBoxIcon.Information);

Application.OpenForms.OfType<MainForm>().First().UpdateUserForm();

AddUsersUI();

}

else if (removeUser == DialogResult.No) { }

}

}

else

{

MessageBox.Show("User must be selected.");

}

}

private void changePasswordButton\_Click(object sender, EventArgs e)

{

if(currentUser != null)

{

// input for new data

string newPassword = Interaction.InputBox("Enter the new password for User: " + currentUser.GetUsername + Environment.NewLine + "Must be 5 characters or longer", "Enter New Password");

if(newPassword.Length == 0) { return; }

if (newPassword.Length < 5) { MessageBox.Show("Password must be 5 characters or longer."); }

else

{

// search for user and change detail

foreach (XmlNode userName in xtr.DocumentElement.SelectNodes("//Username"))

{

if (currentUser.GetUsername == userName.FirstChild.Value)

{

XmlNode parent = userName.ParentNode;

foreach (XmlNode node in parent)

{

if (node.Name == "Password") { node.InnerText = newPassword; }

}

}

}

xtr.Save(directory + @"\User\_Database.xml");

MessageBox.Show("User: " + currentUser + "has had their password changed", "Password Changed", MessageBoxButtons.OK, MessageBoxIcon.Information);

Application.OpenForms.OfType<MainForm>().First().UpdateUserForm();

}

}

else

{

MessageBox.Show("User not selected.");

}

}

private void textBoxFirstName\_TextChanged(object sender, EventArgs e)

{

}

private void comboBoxID\_SelectedIndexChanged(object sender, EventArgs e)

{

int index = comboBoxID.SelectedIndex;

currentUser = user[index];

textBoxFirstName.Text = user[index].GetFirstName;

textBoxLastName.Text = user[index].GetSurname;

textBoxUsername.Text = user[index].GetUsername;

textBoxGroup.Text = user[index].GetModule;

}

}

}