



**TSA National
Conference
Atlanta Georgia
2018**

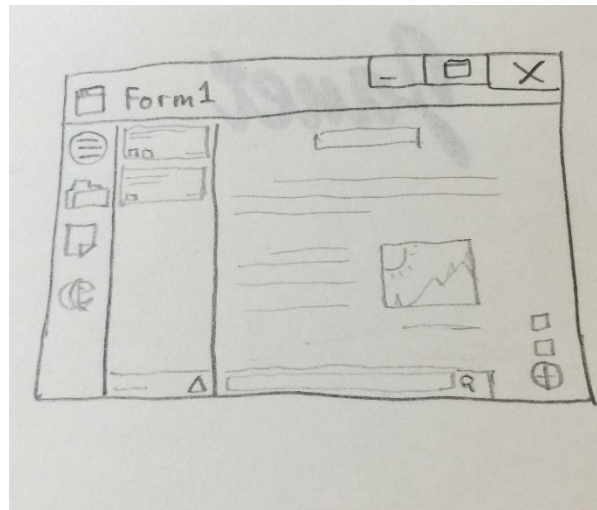
Table of contents

- Research – Page 2
- Project description – Page 4
- Work Logs – Page 5
- Project Requirements – Page 11
- High-Level Software Design – Page 12
- Testing – Page 13
- End-User Documentation – Page 22
- Team Self Evaluation – Page 29
- References – Page 30
- Copyright Checklist – Page 31

Research

We designed a program that helps teachers and administrators record meetings while they occur. Specifically, PLC meetings. This fills the criteria of educational value theme for the software design project. Therefore, for our research, we wanted to focus on who we are really designing the program for. We interviewed teachers at the GTI, and administrators who would also benefit from this program. The next page includes from the meeting we had with a teacher and two administrators.

We learned what kind of features a normal user would expect from this program. The program was designed from the teacher's input, both the design and the functionality. Here is a picture of the original design that we created, although we kept some key ideas, some concepts also changed.

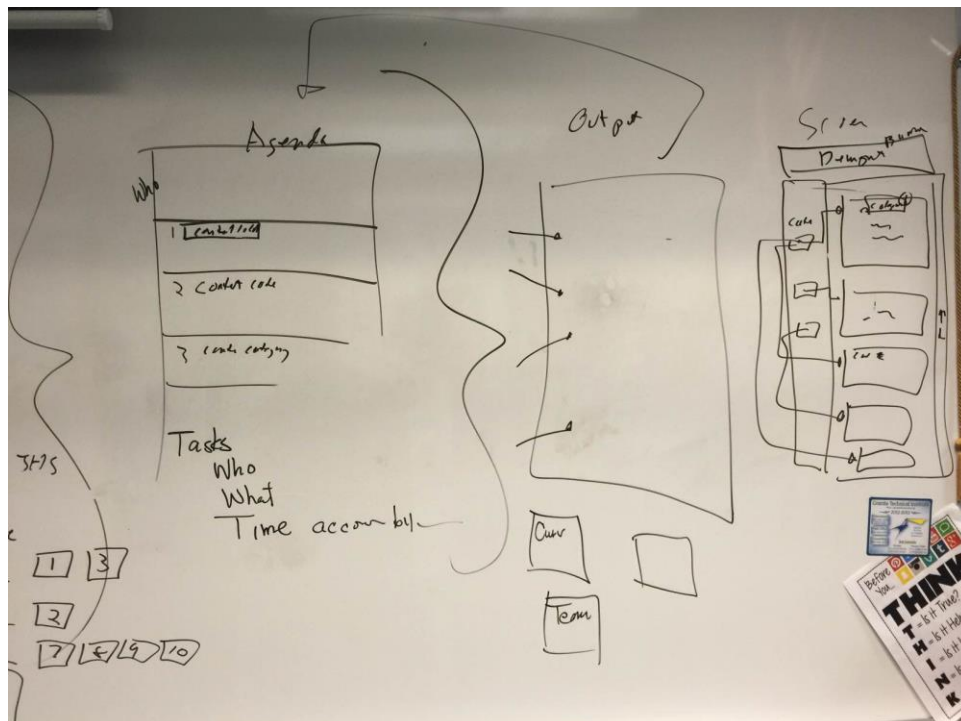
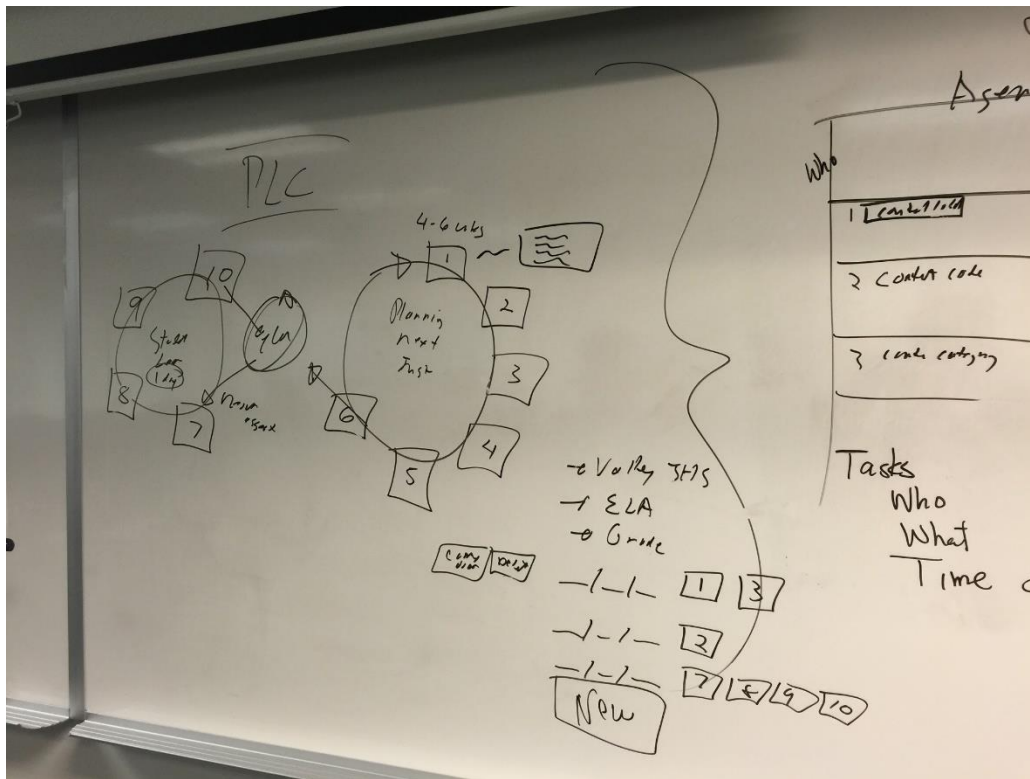


For help with coding we mostly stuck to the tried and true stackoverflow.com (a website which provides help for specific code problems in many languages).

We also used the standard C# documentation at docs.microsoft.com

[Codeproject.com](https://codeproject.com) was used to understand the basics of libraries in C# and the basics of custom controls, along with some bug fixes.

Photos from a presentation with administrators and a teacher





EDUCATION PROBLEM

In the everyday life of a teacher at the Granite School District, meetings are important. Some go one a week, some go more. Carbon, software written in C#, adds a feature that these meetings never had before: Efficient documentation.

WHAT IT DOES

Carbon is a simple program in concept, but hard to implement. The project contains several key features important to the function of the program. It can document meetings efficiently; with different groups of teachers. Carbon also has posts to allow on topic discussion of specific topics with document attachment. Alongside these functions, Carbon also gives users different permissions, to help administrators manage teachers or other users that could create problems if left unmanaged.

MISSION

With our project, teachers can more effectively communicate with other teachers and share learning material, which allows them to more effectively teach their students.

OBJECTIVES

- Record meetings with all information and documents that need to be saved
- Fast, efficient, and clean interface
- Simple to use, yet still powerful
- Searching features with sorting to allow quick locating of what a user needs.

VALUE

Our project aims to help teachers procure meetings that outline key concepts for teaching. Since knowledge is one of the most valuable things a person can learn, giving easy access to it is important. Our project eliminates the need for paper, and sorts information effectively making collaborating with other teachers easier and faster.

Carbon Work Log

12/4/2017 – Planned Work : N/A (This section goes over the last week)

Completed Work : To start, we formed a team. First, we thought of what we were going to build software wise. We came to a concept of a program designed for meetings. The idea came from someone at our institution recommending that a program like this should be designed because it would benefit her in the workplace.

Discrepancies : N/A, (Goes over discrepancies between Planned Work and Completed Work)

12/11/2017 – Planned Work : Design Layout (1 hour, Stefan and Qianlang), Build Backend (6 hours, Stefan), Design basic UI, (3 hours, Qianlang)

Completed Work : At the end of the week, we had the basic layout of the program on paper and had started to code the framework. We began to design the backend, where all the information was stored. Also, we started to design the GUI.

Discrepancies : Didn't finish the backend this week, will continue next week.

12/18/2017 – Planned Work : Finish Backend (2 hours, Stefan), Build Drag and Drop (2 hours, Qianlang)

Completed Work : In the last week, we successfully completed the core part of the backend. The GUI was also starting to look good as aesthetics were applied and we integrated the drag and drop functionality, which took longer than expected because of various difficulties (bugs).

Discrepancies : The actual drag and drop functionality took roughly 8 hours because of various "features" of c#.

12/25/2017 – Planned Work : Build Basic UI Layout - Lists of our groups and such (1 hour, Qianlang), Integrate UI and Backend (2 hours, Stefan and Qianlang)

Completed Work : The GUI was now integrated with the backend, allowing interaction between the two parts. Basic GUI functionality was added for showing three columns and allowing all of them to be dragged and dropped.

Discrepancies : None

1/1/2018 – Planned Work : Design UI Data Fields (30 mins, Stefan and Qianlang)

Completed Work : This week was a lot of designing of the next step of the GUI, the actual display of information. No real programming was done, except a bug fix or two.

Discrepancies : None

1/8/2018 – Planned Work : Design first UI Data Field – Post (4 hours, Stefan and Qianlang), Background Music (30 mins, Stefan and Qianlang)

Completed Work : A big week for the project, we finished the Post Panel, which displayed information for each post that was made. We also added background music. For fun.

Discrepancies : Music took roughly 5 minutes, although the picking of the song took about an hour.

1/15/2018 – Planned Work : Editing Functionality – Post Panel (2 hours, Stefan and Qianlang), Other Data Fields (4 hours, Stefan and Qianlang)

Completed Work : Lots of bug fixes for the Post Panel were added, as well as editing functionality. The counterparts for meetings and groups was also added (Meeting Panel and Group Panel).

Discrepancies : Fixed more bugs than we expected – this is a trend.

1/22/2018 – Planned Work : Polish what we've made so far and add the ability to remove data members by UI (2 hours, Stefan and Qianlang)

Completed Work : Bug fixes. Deletion of posts / groups / meetings was added.

Discrepancies : Trend – More bugs.

1/29/2018 – Planned Work : Add more features to backend (4 hours, Stefan)

Completed Work : More backend work, realizing we needed more functionality.

Discrepancies : None

2/5/2018 – Planned Work : Add post/meeting/group button (2 hours, Qianlang)

Completed Work : Add post button was added.

Discrepancies : Only did the post button because the actual task took 6 hours due to bugs. Future buttons should be easier however.

2/12/2018 – Planned Work : Adding a much needed Save function (2 hours, Stefan) Add the rest of the UI buttons (New Meeting and New Group) (1 hour, Qianlang)

Completed Work : Another big week for the program. We instated a Save feature which allowed the changes made – Although it is with XML which will need to be replaced later. Also, a new meeting button was added.

Discrepancies : None

2/19/2018 – Planned Work : A slight oversight but we realize we also need a load feature, (2 hours, Stefan)

Completed Work : Load feature was added, which made the Save feature useful. Lots and lots of bugfixes as well.

Discrepancies : Trend – Bugs.

2/26/2018 – Planned Work : Added user-oriented help labels (2 hours, Stefan and Qianlang)

Completed Work : More bug fixes that needed ironing out. Added helpful labels that make the program easier to use for someone who didn't design it.

Discrepancies : Trend, None.

3/5/2018 – Planned Work : Upgrade User-friendliness (4 hours, Stefan and Qianlang)

Completed Work : User friendliness was added with an included help page and even more labels for new users.

Discrepancies : None

3/12/2018 – Planned Work : Finish and polish documentation (10 hours, Stefan), Options for music and others (1 hour, Qianlang)

Completed Work : Documentation was spruced up to prepare for the competition, and some final bug fixes were ironed out. A text file with options was added to add optional music.

Discrepancies : Documentation took longer than expected, music options also took longer to figure out, so for the competition we opted out and just created different versions with and without music.

3/19/2018 – Planned Work : Do nothing productive (1 week, Stefan and Qianlang)

Completed Work : After our first-place victory at the Utah State TSA Conference, we decided to take a break for a week.

Discrepancies : None, relaxing went as planned.

3/26/2018 – Planned Work : Replace our XML with a database for improved, everything (15 hours, Stefan and Qianlang)

Completed Work : We added functionality of a database to be able to seamlessly load and save in the background without user input.

Discrepancies : Probably took over 30 hours, needed to find the correct database and way to store data, lots of debate and problems.

4/2/2018 – Planned Work : Add user accounts and functionality (5 hours, Qianlang)

Completed Work : Added user functionality.

Discrepancies : None

4/9/2018 – Planned Work : Make a login screen (2 hours, Qianlang and Stefan)

Completed Work : Added a login screen so an actual user can log in, instead of using pure code.

Discrepancies : Login page had problems with database that improved both of them when fixed.

4/16/2018 – Planned Work : Make a signup page (4 hours, Qianlang)

Completed Work : Sign up page for new users.

Discrepancies : None

4/23/2018 – Planned Work : Add a menu bar for various functions (3 hours, Qianlang), Adds users to groups (2 hours, Stefan and Qianlang)

Completed Work : Added a menu bar on the left and groups can have users as part of their group. However, menu bar so far does nothing but change the database in debug mode.

Discrepancies : None

4/30/2018 – Planned Work : Improve Visuals and Replace Scrollbar (6 hours, Stefan and Qianlang)

Completed Work : Improved visuals greatly by removing lots of seams, also created a custom scrollbar that fits the color scheme.

Discrepancies : Scrollbar took much longer and required a lot more steps than we initially planned.

5/7/2018 – Planned Work : Encrypt passwords in database (1 hour, Stefan), Meetings show members (1 hour, Qianlang), Sorting (1 hour, Stefan and Qianlang)

Completed Work : Added a layer of security with user password encryption inside the database.

Meetings now show members. You can now sort posts/meetings/groups by chronological order, alphanumeric, and for groups, level of permissions.

Discrepancies : Everything took less time than we expected.

5/14/2018 – Planned Work : Reorganize UI (5 hours, Qianlang)

Completed Work : Reorganized the UI to make it more intuitive for new users. With bugfixes.

Discrepancies : None

5/21/2018 – Planned Work : Add code functionality to UI (2 hours, Stefan and Qianlang)

Completed Work : Groups can now add users without the use of code. Meetings now have users and can add them as well.

Discrepancies : None

5/28/2018 – Planned Work : Add code functionality to UI (2 hours, Stefan and Qianlang), Add tag functionality (4 hours, Stefan and Qianlang)

Completed Work : Users can now be removed from both meetings and groups. Basic tag functionality added.

Discrepancies : None

6/4/2018 – Planned Work : Add more features to our text editor (2 hours, Qianlang), Add parent tags to meetings (1 hour, Stefan).

Completed Work : Added new text entering functions like bolding and italics. Meetings now have parent tags that all posts in that meeting must have.

Discrepancies : We ran into some problems with Rich Text Format when it comes to storing into our database, so the text editor took longer.

6/11/2018 – Planned Work : Add search functionality (4 hours, Qianlang)

Completed Work : Added search functionality and fixed many bugs.

Discrepancies : None

6/18/2018 – Planned Work : Prepare all documentation (10 hours, Stefan), Polish and Bugfixes (Qianlang)

Completed Work : Updated help page for all the new functions of the program, preparing for nationals. (Documentation)

Discrepancies : None (so far)

Requirements

Purpose

The purpose of our program is to provide an easy to use meeting recording functionality. It would allow a user to record meetings as they happen and go through that information logically at a later point. With document attachment, a meeting member can easily add relative data with an excel spreadsheet, or a presentation they made, etc. The software must also be user friendly and simple because the target audience is teachers and administrators who may be smart but aren't too tech savvy. At the same time, it will be a powerful tool for those who can use it to its full potential.

Design

- Should be easy for new users to understand and use the program.
- Sleek design that looks modern, yet still serious.
- Easy reordering of posts/meetings/groups so a user can keep the important things on top.
- Create new data members with simple fields to make creation easy.
- Interactive tutorial to show users the functionality of the program.
- Profile pictures, so you can identify people at a glance, and to customize the workspace.
- Darkening of already selected options, to eliminate confusion with unopened posts/meetings.

Functionality

- To save and load to an efficient local database (future updates will make the database centralized).
- Search field to find specific posts/meetings/groups quickly.
- Creation of new posts/meetings/groups.
- User permissions to prevent unwanted tampering – Owner > Admin > Regular User.
- The ability to attach files to posts to add relevant documents.
- Sorting posts/meetings/groups.

High-Level Software Design

Purpose:

This program was designed to record meetings, and our design reflects the basic functions necessary. The user who starts the program can create or delete posts, meetings, and groups. All this is stored in memory (for faster access) but is simultaneously reflected into the local database. The user also can change the order of any of the items by dragging and dropping groups, meetings or posts, which changes the order for the specific user only. When the user attaches a file, Carbon transfers the inputted file into the local database for later opening. The search function keeps to the memory to improve speeds as well. All of these functions allow a user to record a meeting very efficiently.

Who	What	Where
Teacher	Create Post	Memory
	Create Meeting	Memory
	Create Group	Memory
	Delete Post	Memory
	Delete Meeting	Memory
Administrator	Delete Group	Memory
	Drag and Drop	Memory
	Search	Memory
	Login	Disk
User	Attach File	Disk
	Modify Tags	Memory
	Save (Automatic)	Disk
	Load (Automatic)	From Disk

Program Testing

If we were to estimate time spent making the program, only 20% of the time was spent coding features, UI, etc., while the other 80% hours were spent testing and debugging. Therefore, good debugging practices were important and useful.

We designed a method called `tra.ce()` which plays a main role in our debugging strategy. It can trace any object we want to see the details of. We can pass strings, arrays, even objects to this method. It then processes the object we send and outputs relevant information.

```

/// <summary>
/// This is not a class. This is a method (Whaaaaat??).
/// </summary>
public class tra
{
    /// <summary>
    /// An advanced or maybe the world-best trace method.
    /// It simply writes everything you pass as args into the output-panel.
    /// </summary>
    static public string ce(params object[] args)

```

In this example, we are going to test our `LoadData()` function, and we want to check if we loaded the group data properly. After we wrote the code for the loading function, we used the `tra.ce()` method to see the data that we loaded:

```

static public List<Group> LoadData(string url)
{
    List<Group> groups = new List<Group>();

    if (!File.Exists(url))
        throw new FileNotFoundException("Where in the world is that location!!");

    Group currentGroup = new Group("");
    Meeting currentMeeting = null;
    Post currentPost = null;

    groups.Add(currentGroup);

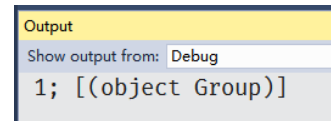
    XmlReader reader = XmlReader.Create(url);
    string currentNode = "";

    while (reader.Read())
    {
        tra.ce(groups.Count, groups);
        reader.Close();
    }

    return groups;
}

```

If we run the program, the `tra.ce()` function outputs that there is only 1 group loaded. But we know that the sample data we inputted has 3 different groups. This means we have a flaw in the loading function.



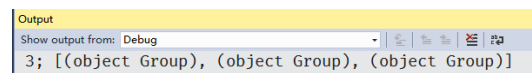
Output
Show output from: Debug
1; [(object Group)]

To start solving the problem, we read the code again carefully and found that we only created one group object. To solve the problem, we created new group objects as the reader reads, and added them into the list.

```
while (reader.Read())
{
    switch (reader.NodeType)
    {
        case XmlNodeType.Element:
            currentNode = reader.Name;

            switch (reader.Name)
            {
                case "group":
                    currentGroup = new Group(reader.GetAttribute("name"));
                    groups.Add(currentGroup);
                    break;
            }
        }
    }
}
```

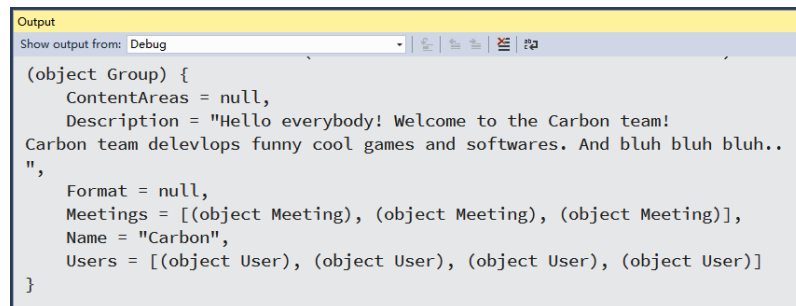
After we changed the code, we ran the program again and saw 3 groups in the list outputted by the `tra.ce()` function, showing that we fixed the problem.



Output
Show output from: Debug
3; [(object Group), (object Group), (object Group)]

To make sure that those groups contain the same data as the data in the file, we use the `tra.ce()` function on the first group to see its details.

```
}
}
tra.ce(groups[0]);
reader.Close();
```



Output
Show output from: Debug
(object Group) {
 ContentAreas = null,
 Description = "Hello everybody! Welcome to the Carbon team!
Carbon team delevops funny cool games and softwares. And bluh bluh bluh..
",
 Format = null,
 Meetings = [(object Meeting), (object Meeting), (object Meeting)],
 Name = "Carbon",
 Users = [(object User), (object User), (object User), (object User)]
}

Comparing the output to the test file tell us that we imported everything correctly. Problem solved.

Now we are going to test our SaveData() function. (Code hidden in foreach)

```
static public void SaveData(List<Group> groups, string url)
{
    AddXMLText($"<?xml version=\"1.0\" encoding=\"utf-8\" ?>");
    AddXMLText($"<data>");

    foreach (Group g in groups) {

        AddXMLText("</data>");

        File.WriteAllText(url, ""); // delete previous records
        File.WriteAllLines(url, xmlFile);

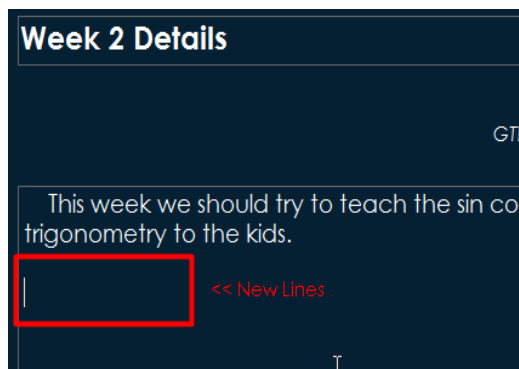
        xmlFile.Clear();
    }
}
```

After we loaded the sample data file, we just saved it to another file without making any changes, so we can see if it would be the same as the original file.

Before

After

As we can see in the picture, it's not quite the same, because our formatting has some issues. These problems will make a difference when we load the file again, which means the users will see their saved data change. We need to avoid that.



After rereading the code carefully, we find that, when we were loading the data, we changed the “\n” (newlines) in the files to Environment.NewLine (because Environment.NewLine does better in user interfaces), and we didn’t change the Environment.NewLine to “\n”’s back when we saved it. Again we use the tra.ce() method to find out if our theory is correct.

```
static public void SaveData(List<Group> groups, string url)
{
    AddXMLText($"<?xml version='1.0' encoding='utf-8' ?>");
    AddXMLText($"<data>");
    tra.ce(groups[0]);
    foreach (Group g in groups) {...}
    AddXMLText("</data>");
    File.WriteAllText(url, ""); // delete previous records
    File.WriteAllLines(url, xmlFile);
    xmlFile.Clear();
}
```

```
Output
Show output from: Debug
(object Group) {
  ContentAreas = null,
  Description = "Hello everybody! Welcome to the Carbon team!
Carbon team develops funny cool games and softwares. And bluh bluh bluh..
",
  Format = null,
  Meetings = [(object Meeting), (object Meeting), (object Meeting)],
  Name = "Carbon",
  Users = [(object User), (object User), (object User), (object User)]
}
```

As expected, there is an unnecessary new-line after the description of the group named Carbon. To fix the problem, we edited our existing formatting function to replace all the Environment.NewLine with “\n”s. Additionally, to avoid unnecessary empty spaces after the strings, we remove the backspace-characters from the string.

```
static protected string FormatXMLText(string text, string mode)
{
    // when loading, we need to replace \n's with Environment.NewLine,
    // when saving, replace Environment.NewLine with \n's

    if (mode == "load")
        text = text.Substring(1).Replace("\t", "").Replace("\b", "").Replace("\n", n.l);
    else if (mode == "save")
        text = text.Replace("\t", "").Replace("\b", "").Replace(n.l, "\n");

    // remove annoying empty char at the end of the string
    if (text[text.Length - 1] == '\b' || text[text.Length - 1] == '\n' ) // bingo!
        text = text.Substring(0, text.Length - 1);

    tra.ce(mode, text);

    return text;
}
```

Now we can test the new advanced-formatted string via tra.ce(). We can see that it removes the unnecessary new-line at the end of the string but keeps the other newlines where they’re supposed to be.

```

Output
Show output from: Debug
"Hello everybody! Welcome to the Carbon team!"
Carbon team develops funny cool games and softwares. And bluh bluh bluh.."
"The blogs published by the world's best programming group Carbon."
" The professional programming team Carbon(tm) is now programming their new cl
" Form1 is making a pretty Carbon Logo! Let's hope we can see it soon!"

```

Also, to format the tabs so that it is more readable in the data file during testing, we programmed the function to add certain numbers of tabs before each line. Since we remove all the tabs when the user loads data, the user cannot tell.

```

static protected void AddXMLText(string toAdd, int tabs = 0)
{
    string tabString = "".PadRight(tabs, '\t');
    xmlFile.Add(tabString + toAdd.Replace("\n", "\n" + tabString));
}

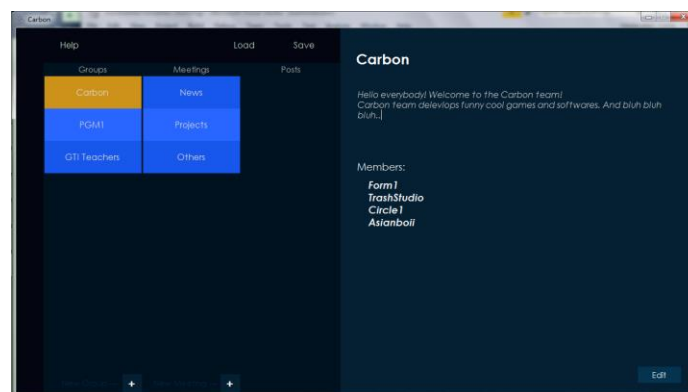
```

```

SampleData.xml
1 <?xml version="1.0" encoding="utf-8" ?>
2 <data>
3   <group name="Carbon">
4     Hello everybody! Welcome to the Carbon team!
5     Carbon team develops funny cool games and softwares. And bluh bluh bluh..
6     <user name="Form1" origin="Coleman"/>
7     <user name="TrashStudio" origin="Stefan"/>
8     <user name="Circle1" origin="Marlon"/>
9     <user name="Asianboii" origin="Qianlang"/>
10    <meeting name="News">
11      The blogs published by the world's best programming group Carbon.
12      <post name="New Carbon Client UI" date="17-12-20">
13        The professional programming team Carbon(tm) is now programming their ne
14      </post>
15      <post name="Carbon Logo by Form1" date="18-1-4">
16        Form1 is making a pretty Carbon Logo! Let's hope we can see it soon!
17      </post>
18      <post name="TS and Circle1 plays diep in class!!!" date="18-1-6">
19        At the end of the term, TrashStudio and Circle1 played diep.io on most o
20      </post>
21      <post name="We've all gotta 45 in Diepio!" date="18-1-10">
22        Every Carbon team member has got to 45th level in diep.io! Go Carbon!
23      </post>

```

Now, the saved data looks much more readable. One more time, we load the same saved data again, just to make sure it's loaded correctly. Problem solved.



Feature Testing

Feature: Show Password

Action: Tap and hold the button.

Expected Result: Shows your password while the button is hold down and hides back your password when the button is released.

Actual Result: Exactly as expected.

Feature: Check Username Existence

Action: Click.

Expected Result: Checks if the username signing up already exists in the database and warns you to choose another username if it already exists.

Actual Result: Exactly as expected.

Feature: Help Button

Action: Click.

Expected Result: Help window shows up and displays the help content relevant to what the user was viewing before the help button was clicked.

Actual Result: Exactly as expected.

Feature: Search Groups, Meetings, and Posts

Action: Type a keyword to search for groups, meetings, or posts.

Expected Result: A search result window with the groups, meetings, and posts that has content containing the keyword.

Actual Result: Exactly as expected.

Feature: List Menu Buttons

Action: Shows the menu of a list only when the mouse hovers it.

Expected Result: Displays the menu buttons on the list of groups when the mouse hovers the list of groups and hides them when the mouse leaves the list of groups.

Actual Result: Exactly as expected.

Feature: Drag-and-droppable Lists

Action: Rearrange the order of the items in a list by using the mouse to drag and drop them.

Expected Result: While the mouse is pressed, the list-item directly below the mouse follows the mouse cursor but stays in the list, until the mouse is released.

Actual Result: Exactly as expected.

Feature: Sort List-Items

Action: Sort the items in a list by a specific parameter (name, date, etc.)

Expected Result: The items in a list are sorted alphabetically from A to Z when “Sort Items by Name” option is chosen.

Actual Result: Exactly as expected.

Feature: Create Groups, Meetings, and Posts

Action: Hover the mouse onto the list of group/meeting/post and choose “Create a group/meeting/post”.

Expected Result: Shows up a panel that contains a bunch of text-boxes where users can enter the information and add members for the new group they’re creating. When users are done and the “Create” button is pressed, creates a new record of group/meeting/post in the database.

Actual Result: Exactly as expected.

Feature: Edit Groups, Meetings, or Posts

Action: Click on “Edit” button.

Expected Result: Every text-box in the panel becomes editable (only if the user has that permission) to let the users edit the content of them. When the users are done and the “Save” button is clicked, updates the record of that group/meeting/post in the database.

Actual Result: Exactly as expected.

Feature: Format Post Content

Action: Create a new post or edit an existing post.

Expected Result: Shows an email-editor-like menu where users can format their post content however they like. Makes the selected text bolded, italic, or underlined, or adjusts the size or color of the selected text, as the users choose the relevant options.

Actual Result: Exactly as expected.

Feature: Add Members

Action: Click on “Add Members” button in the group/meeting panel.

Expected Result: Pops up a window with a text-box where users will search for new members, shows results as the users type in their keywords. When the users are done, adds those chosen new members to the record of the group/meeting in the database.

Actual Result: Exactly as expected.

Feature: Remove Members

Action: Click on the “Edit” button, hover the mouse onto an icon of a group/meeting member in the meeting list, then click on the “X” on the top right.

Expected Result: When the users are editing the information of a group or meeting, shows a “Remove” button as the mouse hovers the icon of each group member. When the button is clicked, asks to make sure if the users want to remove them, then remove the member from the record of the group/meeting in the database.

Actual Result: Exactly as expected.

Feature: User Basic Information Panel

Action: Click on an icon of a member of a group or meeting.

Expected Result: When the icon of a member in the member list of a group or meeting is clicked, pops up a window that shows the name, username, status, email address, and phone number of that member. (This information was provided by the member when they signed up to the Carbon platform.)

Actual Result: Exactly as expected.

Feature: Manage Group Permission Settings

Action: Click on the “Advanced Settings” button in the group panel.

Expected Result: When “Advanced Settings” button is clicked, a window pops up with the permission settings of the group. Regular group members can only view those settings, but the owner of the group may click on the “Edit” button to edit the permission settings.

Actual Result: Exactly as expected.

Feature: Add/Remove Tags for Meetings or Posts

Action: Click on the “Add Tag” button to add a tag or click on the “X” while mouse is hovered a tag to remove a tag.

Expected Result: When the “Add tag” button is clicked, a new tag is created and editable so that the users can type the name of the new tag and press enter to save it to the meeting or post. When the mouse hovers an added tag, a little “Remove” button shows up, which by clicking will remove a tag from a meeting or post.

Actual Result: Exactly as expected.

Feature: Add/Remove Attachments for Posts

Action: Click on the “Add Attachment” button to attach a file or click on the “Remove” button while mouse is hovered an attachment to remove an attachment.

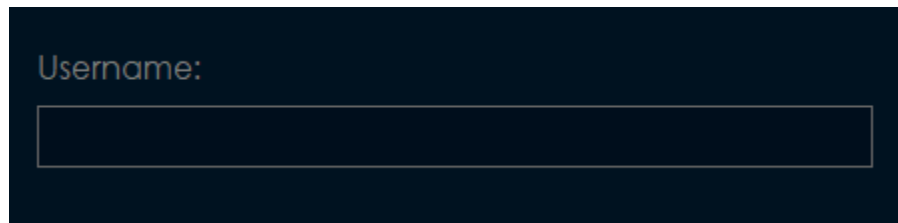
Expected Result: When the “Add Attachment button is clicked, an “Attach Files to Post” window pops up. When the users are done with selecting the files to attach to the post, links the files to the post and the record of it in the database, then shows the list of all attached files of the post. When the mouse hovers an attachment, a “Remove” button shows up, which by clicking will remove an attachment from a post.

Actual Result: Exactly as expected.

End User Documentation / Instructions

Login

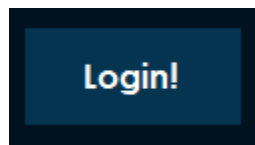
To login, enter your username in the username field,

A dark blue rectangular box containing the label "Username:" in a light gray font. Below the label is a white rectangular input field with a thin gray border.

Then enter your password in the password field

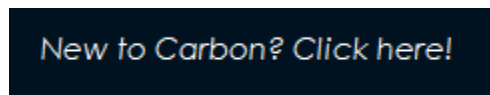
A dark blue rectangular box containing the label "Password:" in a light gray font. To the right of the label is a small, light blue button with the text "Show Password" in a darker blue font. Below the label and button is a white rectangular input field with a thin gray border.

And then press login at the bottom right

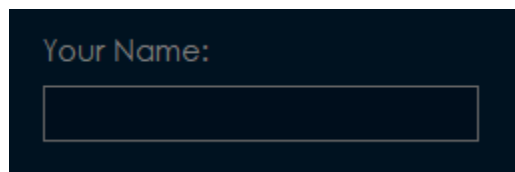
A dark blue rectangular button with the text "Login!" in a white font.

Sign-up

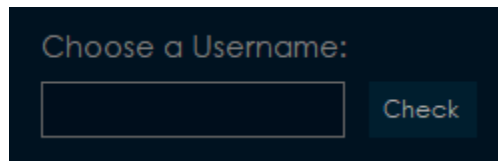
If you do not have a user account press - New to Carbon? Click here!

A dark blue rectangular button with the text "New to Carbon? Click here!" in a light gray font.

Enter your information including your name,

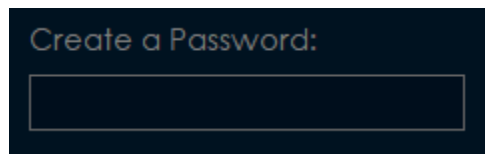
A dark blue rectangular box containing the label "Your Name:" in a light gray font. Below the label is a white rectangular input field with a thin gray border.

A unique username,

A dark-themed form with the label "Choose a Username:" in a light blue font. Below the label is a white rectangular input field. To the right of the input field is a blue button with the word "Check" in white text.

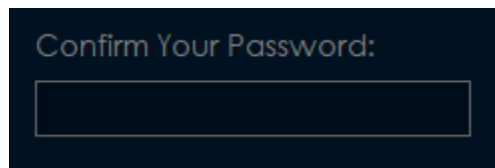
Choose a Username:

And a password at least 8 characters in length.

A dark-themed form with the label "Create a Password:" in a light blue font. Below the label is a white rectangular input field.

Create a Password:

And once again...

A dark-themed form with the label "Confirm Your Password:" in a light blue font. Below the label is a white rectangular input field.

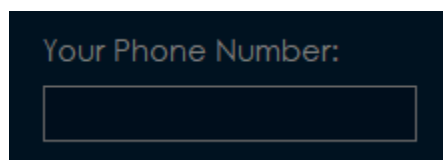
Confirm Your Password:

You may optionally add your email,

A dark-themed form with the label "Your Email:" in a light blue font. Below the label is a white rectangular input field.

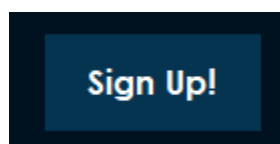
Your Email:

And your phone number.

A dark-themed form with the label "Your Phone Number:" in a light blue font. Below the label is a white rectangular input field.

Your Phone Number:

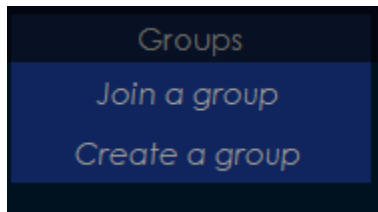
When you're finished, press Sign Up.

A dark-themed button with the text "Sign Up!" in white, bold font.

Sign Up!

Group

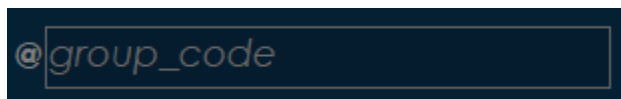
To create a group, hover your mouse over the column where the groups go (the left one) and press the button labeled Create a Group.



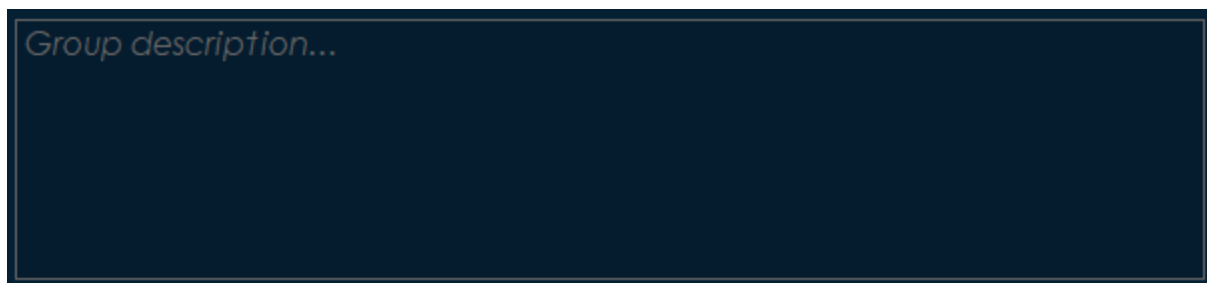
On the right, enter your Group name in the top box labeled Group Name,

A dark blue text input box with a light blue border. The text 'Group Name' is written in a light blue font inside the box.

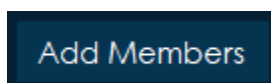
Next, enter a unique group code in the box labeled "group_code,"

A dark blue text input box with a light blue border. The text '@group_code' is written in a light blue font inside the box.

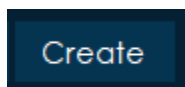
Then add a description of your group in the big Group description box

A large dark blue text input box with a light blue border. The text 'Group description...' is written in a light blue font inside the box.

Add whatever users you need with the add members button.

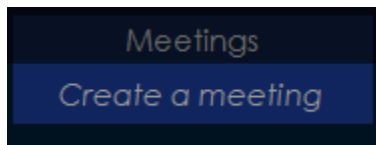
A dark blue button with a light blue border. The text 'Add Members' is written in a light blue font inside the button.

When you're finished, press Create at the bottom right to make your group.

A dark blue button with a light blue border. The text 'Create' is written in a light blue font inside the button.

Meeting

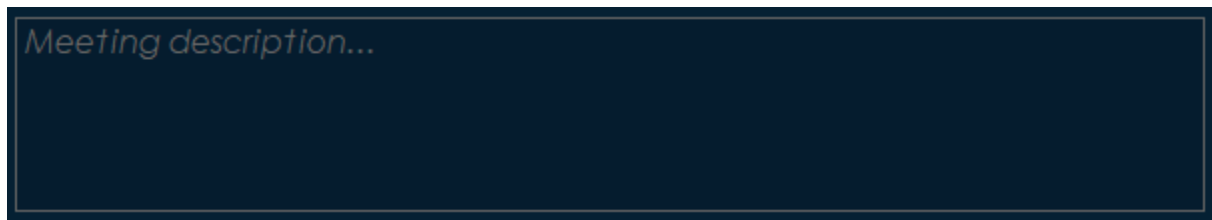
To create a meeting, hover your mouse over the column where the meetings go (the middle one) and press the button labeled Create a Meeting.



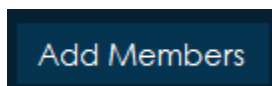
On the right, enter your Meeting name in the top box labeled Meeting Name.

A dark blue rectangular input field with a thin white border. The text "Meeting Name" is written in a light blue, sans-serif font at the top left of the field.

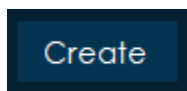
Then add a description of your meeting in the big Meeting description box.

A large, dark blue rectangular input field with a thin white border. The text "Meeting description..." is written in a light blue, sans-serif font at the top left of the field.

Add whatever users you need with the add members button.

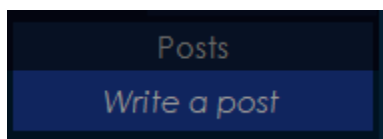


When you're finished, press Create at the bottom right to make your meeting.



Post

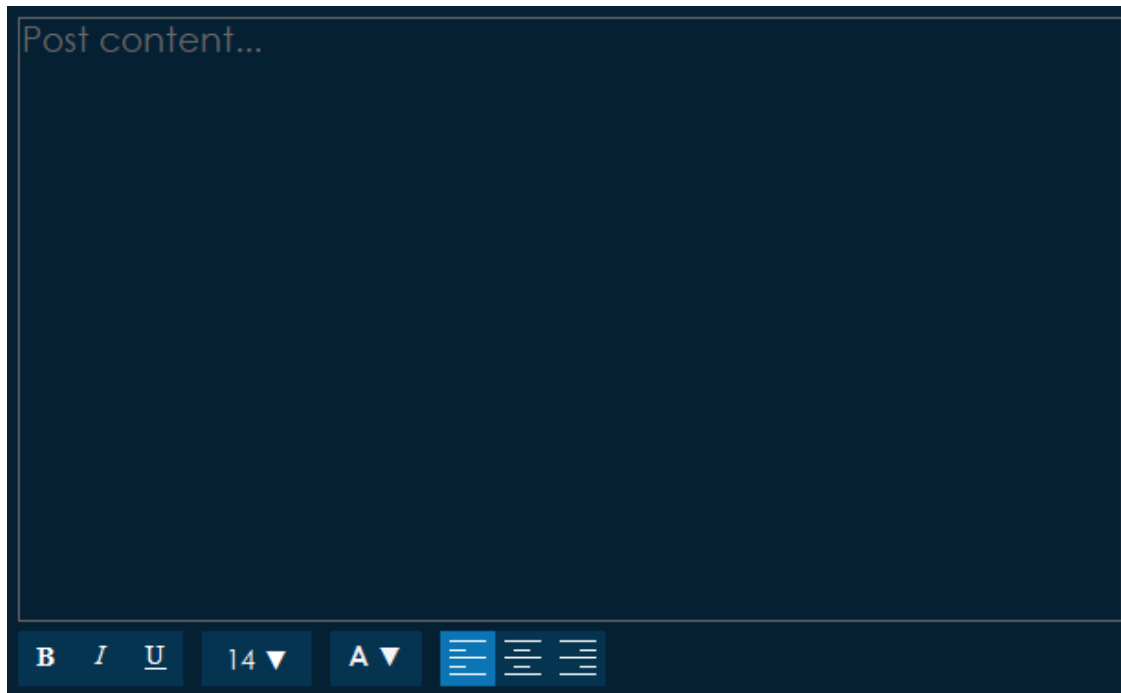
To create a post, hover your mouse over the column where the posts go (the right one) and press the button labeled Write a Post.



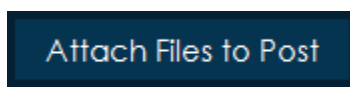
On the right, enter your Post name in the top box labeled Post Title.

A dark blue rectangular box with a thin white border. Inside the box, the text "Post Title" is written in a light blue, sans-serif font.

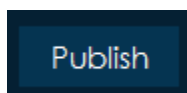
Then add your post content in the big Post content box -- Feel free to use the formatting tools at the bottom.

A large dark blue rectangular box with a thin white border. Inside the box, the text "Post content..." is written in a light blue, sans-serif font. At the bottom of the box, there is a row of formatting tools: a bold icon (B), an italic icon (I), an underline icon (U), a font size dropdown showing "14" with a downward arrow, a text color dropdown showing "A" with a downward arrow, and three alignment icons (left, center, right).

If there are any relevant files to add, such as an excel spreadsheet, press the Attach Files to Post button at the bottom to add the files.

A dark blue rectangular button with a thin white border. Inside the button, the text "Attach Files to Post" is written in a light blue, sans-serif font.

Whenever you're done, press Publish at the bottom right to finish your post (Don't worry, you can edit it later by pressing edit)

A dark blue rectangular button with a thin white border. Inside the button, the text "Publish" is written in a light blue, sans-serif font.

Tags

Tags are an important part of Carbon, because they are general identifiers on your posts and meetings to identify what kind of content is in the meeting/post. To add a tag on a meeting or a post, first open a meeting or a post so you see the contents on the right. Next, press an empty tag under the name of the post or meeting.



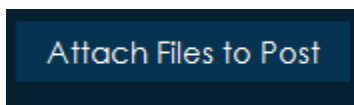
Fill in the tag with whatever it will be called, and press enter.

Quick - tip: If a meeting has a tag, every post under the meeting will also share that tag, and unless you remove it from the meeting, it cannot be removed from a post.

Quick-tip #2 - in the search bar, if you want only posts and meetings that contain certain tag, just write "\"Tag: [tag name]\"" and it will show all meetings and posts with that tag.

Attachments

To add attachments to a post, simply go to a post you would like to add the attachments to and press the button at the bottom called Attach Files to Post.



If you want to modify the already attached posts, press the view attachments button

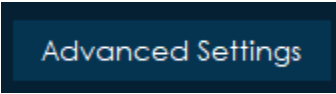


You can double click an attachment to open it. Press the edit button at the bottom right to remove an attachment.


 Edit

Permissions

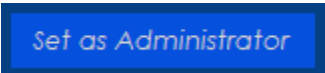
Permissions work like this: Owner > Administrator > Regular User. An owner or administrator of a group can create meetings, while a regular user can. Any user however, can make a post. An owner or administrator can promote normal users to an administrator by selecting a group and going to advanced settings at the bottom right.


 Advanced Settings

Here, you can also modify lots of permissions, such as letting more than just admins do a specific task or restricting a task to just the owner.

Invite Members <i>Who can add new members to this group.</i>	<u>Administrators</u>
Remove Members <i>Who can remove members from this group.</i>	<u>Administrators</u>
Create Meetings <i>Who can create meetings in this group.</i>	<u>Everyone</u>
Delete Meetings <i>Who can delete the meetings they're in.</i>	<u>Owner</u>

To make a user a group administrator, click their picture/name in the group user menu and press set as administrator.


 Set as Administrator

Team Self Evaluation

What we expected and what we got

When we started the project, we didn't really know what we were getting into. We were excited and were ready to jump in headfirst, without realizing how big the hole is. We knew what we wanted to build, but we didn't know how much work really entailed. The idea was a meeting management program, and it fused into a very versatile program made for any type of nested storage (Our specialized classes for posts/meetings/groups in this case). At the end, we got a program that had many of the key features we were looking for, but not all. Over the three months since our state TSA competition, we have worked harder than we did before, because we knew we would have competition.

Troubles in between

At the beginning we had a problem where we didn't really plan out our project before we started. Ignoring the general software development practices was a bad idea and after a week of struggling we went back to the drawing board. A memorable (yet devastating) moment during our process was one of our team members deleting two weeks' worth of work. This demotivated the team for a good few weeks until we got back in track and built it much more efficiently than before.

A few big arguments challenged our team as friends and as developers as we were unable to decide on some key decisions about our program, such as the design of our panels. Every time we had an argument like this, we worked through it by compromising with each other to come to a product that we all liked even more than our original ideas separately.

Future plans

Our plans listed below were from a few months ago, and the ones that are struck out are goals that we have completed. It is presented like this to show the progress we had, from just a few months ago at our state TSA competition.

- Server functionality where data isn't stored locally.
- ~~Users and groups which give people permissions and identities (and responsibility)~~
- ~~Scrolling in lists because the current maximum is 7~~
- ~~A layout that is dynamic which allows for window resizing~~
- ~~Add more media/attachments options for posts~~
- ~~Personalization~~
- Login page/Splash screen
- ~~UI Sorting~~
- Printouts of meetings
- Provide history of posts including who edited what
- Communication between clients

References

The entire program was designed, created, and coded **by ourselves** without borrowing code from websites or any source. The only reference we have is the optional background music that is royalty free.

Music from Doctor Vox – Frontier