Blue Team Games

"Security. It matters."

Taylor Smith (Software Architect)
Seth Little (Editor and Game Design Architect)
Vinish Kumar (Client/Instructor Contact)
Iftekhar Uddin (Project Manager)
Dan Barker (Client, UNC Finance and Operations)

Last update: April 23, 2020

GDD Template Written by: Benjamin "HeadClot" Stanley

Overview

Theme / Setting / Genre

- Collection of Cyber Security Based Games with an emphasis on Social Competitiveness

Core Gameplay Mechanics Brief

- Platformer, Tower Defense, Message Board, Jeopardy

Targeted Platforms

- Web Browser on 1400+ devices of UNC employees (PCs, Laptops, and iPads)

Target Users

- The target user is a UNC employee (with an onyen) who works for the Finance and Operations department at UNC. These users are primarily English-speaking, very competitive, and often interchanging. In addition, they are not the technology generation and include slightly older people. In the words of our client, these employees are "normal adults who don't think about security." A main priority will be to discourage link clicking among these users.

Server

- Windows 2019 server with IIS on it with a MS SQL 2019 server to be accessed via Citrix (Virtual Labs). Shibboleth will be set up for onyen authentication using SSO, Single Sign On authentication (More info in the Development Section)

Project Scope

- One Semester
    - It will cost server space.
    - It will take one semester to build the base modular system with a game or two
- Roles of the 4 Team Members
    - Vinish Kumar
        - Client/Instructor Contact
    - Taylor Smith
        - Software Architect

- Responsible for bringing all the pieces together
- Seth Little
  - Editor
    - Responsible for making sure everything's together
  - Game Design Architect
    - Aims the team toward the desired visual design
- Iftekhar Uddin
  - Project Manager
    - Tracks everyone's progress

Influences

- Doodle Jump
- Mobile Game
- Simple Game with a simple mechanics
  - Click and Tilt

- Jeopardy
- Television Show
- The design of a social game with a quiz element allows the players to learn things while also having healthy competition

- Tower Defense
- It's a simple game type that allows for research, optimization, and learning of the character types.

The Elevator Pitch

A social browser game where UNC users can compete in mini games based on

Computer Security in order to improve software practices.

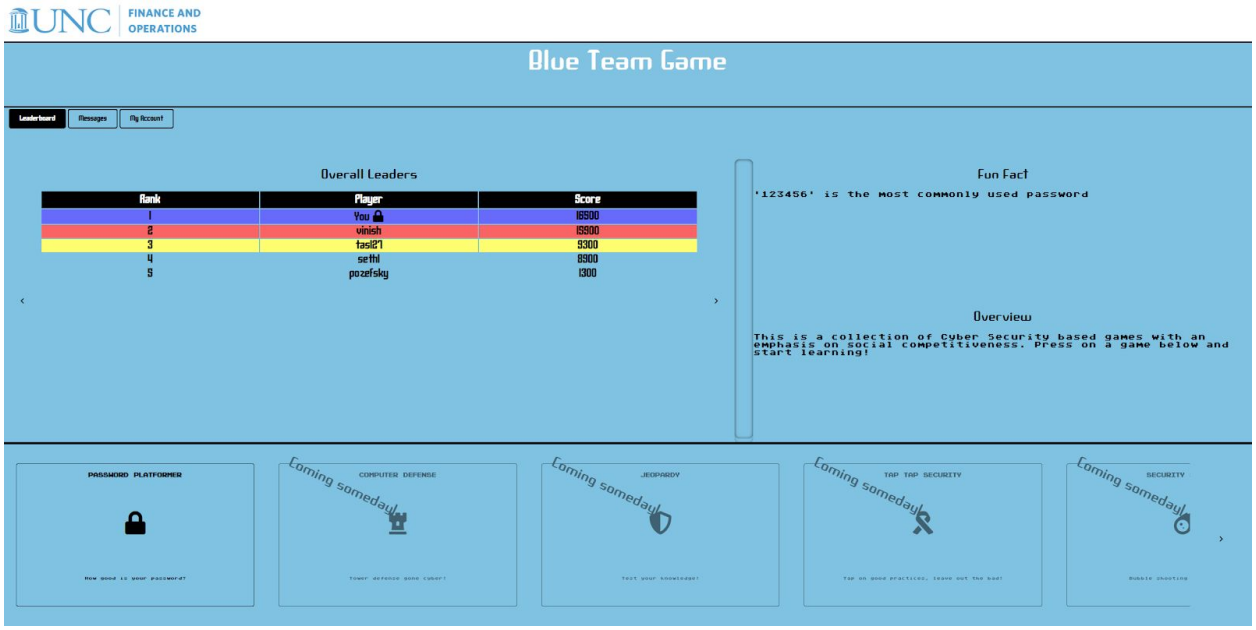
Project Description :

The client envisions a game that can be played by his various clients, including UNC employees with departments from Facilities to Groundskeeping, with sensitive information to improve their habits online. A UNC onyen is used to sign into the game and to link scores to an account. There are differences in knowledge among the future users of the game across departments and even within a department (e.g. a manager might have more experience online than someone in the field). One particular topic that was emphasized by the client was not trusting the internet and not clicking the links because they often install malware (spam identification). The client would like to discourage software installed games and prioritize non flash browser games. The game(s) should preferably be compatible with tablets since there will be about 500 tablet-using faculty in the coming years. Finally, the client would also like us to prioritize competition, as this aspect would drive usage and therefore learning, and modularity, since there are many topics to teach and the game should be able to be handed to the next developers for further improvement at the end of the semester. To allow for ease of transfer, we will include an API to describe how future developers can easily add games into this design, and update the badge icons as needed.

Our most important priority is to get the gamification of this project correct, since we want to keep our users to stay engaged enough to learn the

information. As Dan said, “Make it pretty, make it interesting, make it businesslike.”

Home Screen



Current Image of the Home Screen

We envisioned the top two-thirds of the screen to hold the leaderboard (which is vital due to the social aspect helping to “edutain” our users), informational text, the message board, and access to your account, which displays your badges and various rankings. The leaderboard will show overall rankings as well as rankings for individual games when the arrows next to the leaderboard are clicked (see below). In the leaderboard, the top 3 overall scorers will have a blue, red, and yellow highlight respectively to emphasize the achievement. By default, all leaderboards show the top 10 scores, but

users can click a See More button to load more scores if they are available. The overall leaderboard currently displays all badges held by each player (currently defined by if they are in the top 15% of scores for a given game), but future iterations will allow the player to choose which badge(s) they would like to display beside their name. Badges serve as an indicator both for breadth (has the user learned a lot about various topics by playing different games?) and depth (has the user learned a lot about a topic by engaging deeply with a particular game?). The lower third is dedicated to world or game selection, laid out in the form of a carousel with a few games on the screen (imagine a Google search-esque related items display). Games which are not complete yet will be grayed out with text along the lines of "Coming Soon"; this serves to show the users that there will be more information coming while abstracting the initial size of the overall game. Each game will be displayed with a fun, informative name (e.g. Go Phish!) and a cool icon; these come from the Font Awesome Icons (https://fontawesome.com/) currently, but future developers could develop new icons (functionality which is allowed by the modularity of our design). Clicking on the game will replace the leaderboard section with an informational section which will give the user instructions on how to play the game and a light introduction to the topic taught in the game (max 3

sentences). Along with this text, there will be buttons to play the game or the user can select a tab to go back to the leaderboard.

-Badges

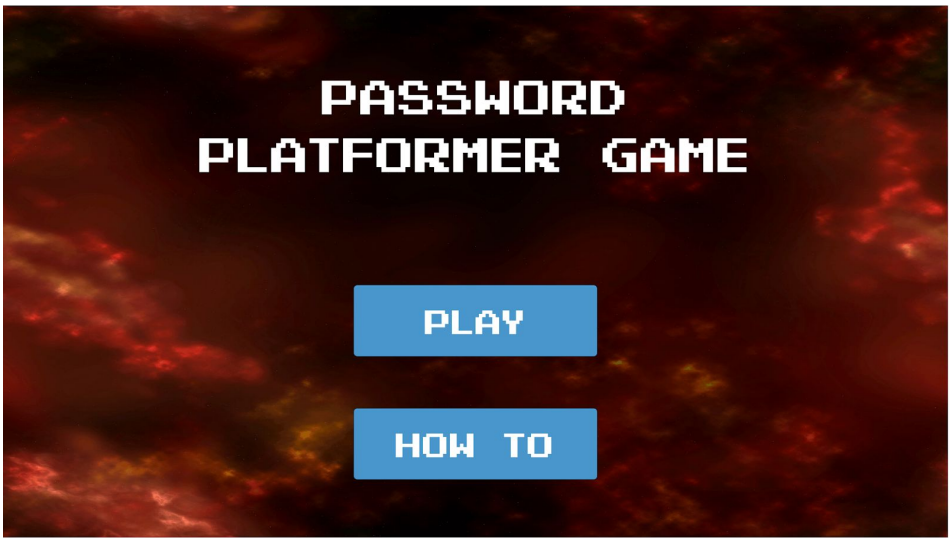
The badges currently use Font Awesome icons to represent the games and their badges. A badge icon for say the password platformer will be the same as the icon for the game itself, in this case, a lock. Badges are given to the top 15% of scores in the game; users can look at the individual game leaderboard to determine the top scores overall in the game. Whatever badge design is used, they need to be made ADA compliant and be recognizable for people with colorblindness (perhaps by putting a 1, 2, 3, etc. inside).

-Fun facts

To allow for several opportunities to learn, there is a fun facts area on the main page that displays different fun and informational facts about computer security every 30 seconds. This will be pulled in from a .txt file (located at /static/funFacts/funFacts.txt) that Dan will be able to edit and make easy changes to from the server. The file contents are pulled into the index.js file using the fetch() command.

Games

- Password Platformer

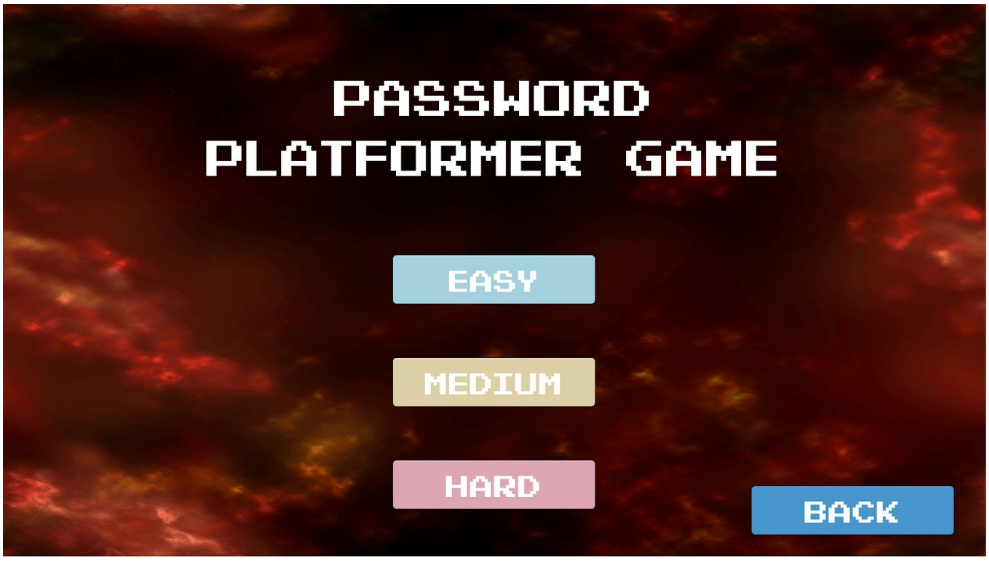


We chose the space theme to really engage users. Both the background and the tile map were downloaded for free from the Unity Asset Store. Here is the beginning page with the Play and How To choices.

Overview

For the first mini-game, we made a platformer game based around the idea of being able to identify strong and weak passwords. We planned it to be a game similar to Super Mario (though the game is also Roguelike with procedurally generated platforms - the only limitation is the timer) where the player moves across a world from left to right and jumps on platforms that have passwords written on them. If the player jumps on a platform with a strong password, they will be rewarded with 100 points. Additionally, a 2x and then 4x multiplier is added as the player continues to get passwords correct (2x for 5 correct choices in a row and 4x for 10 choices in a row). If the player

jumps on a platform with a weak password, the player loses any streak and points according to their chosen difficulty level. If the player falls off the map, he/she is immediately respawned at the most recent middle ground. Players will get points based on how many passwords they get correct before the time runs out (currently 90 seconds).



This is the difficulty choice screen, with easy, medium, hard, and a back button to the beginning page.

This game will have three difficulty modes. The game uses the ZXCVBN password model, which was recommended by our client and will be used to test the strength of passwords in the upcoming one-year Onyen password system. This system judges passwords on a 0-to-4 scale where 0 is incredibly easy to crack and 4 is insanely difficult. Good passwords, according to our model, score a 4; meanwhile, bad passwords score a 2 or below. The overall space of good

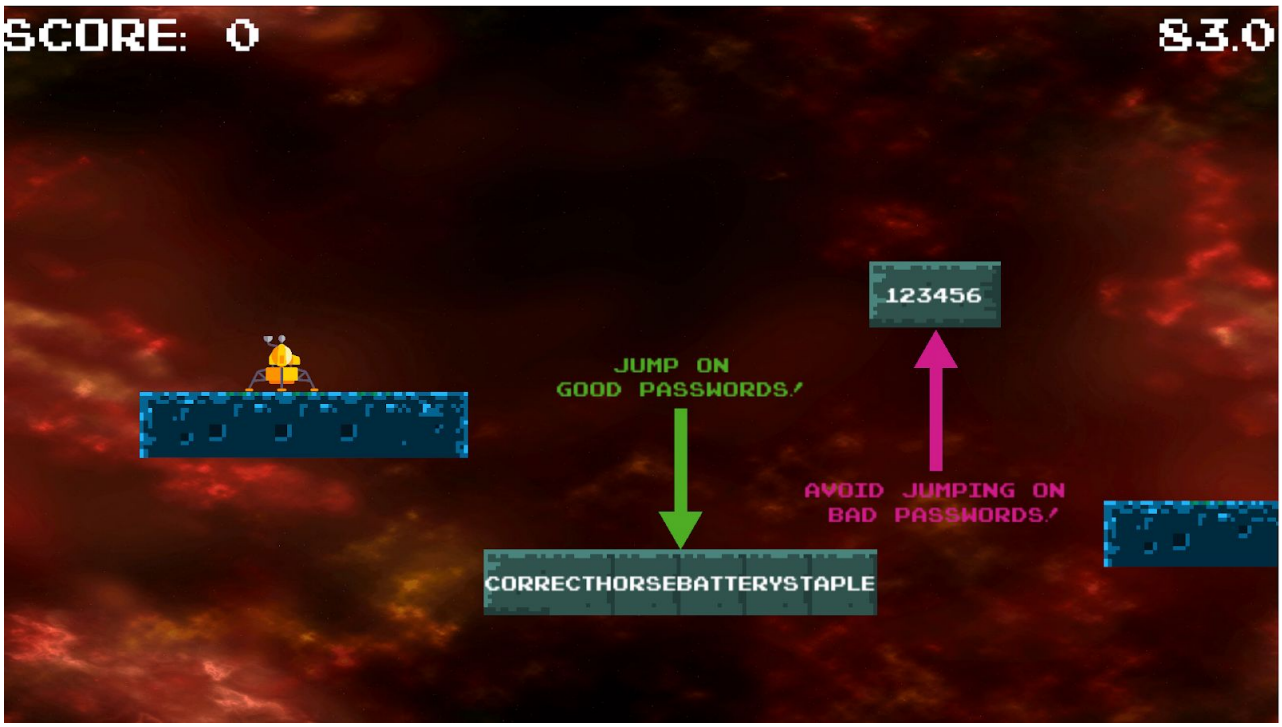
passwords is made up of passwords made up of three to four English words, obscured by converting them partially to l33t-speak (internally, the password generation script uses the same l33t map as the ZXCVBN password model rather than a much broader list) and ensuring that they receive a 4 from ZXCVBN. Different difficulty levels of passwords are made based both on the differential length between good and bad passwords and the makeup of the bad passwords. In the easy game mode, bad passwords are either a male name, female name, surname, or a common password pulled from the ZXCVBN internal files. Good passwords in this mode consist of the entirety of currently generated good passwords. In the medium game mode, bad passwords are a male name plus a surname (or vice versa), a female name plus a surname (or vice versa), anywhere from two to four English words, or any of the above converted partially to l33t speak. Good passwords in the medium mode are those good passwords which are within two standard deviations of length of the average medium bad password. Finally, the bad passwords in the hard mode are made up of all the medium passwords, those passwords reversed, those passwords with punctuation at the end, those passwords with numbers at the end, or any of the above partially converted to l33t speak. Good passwords in the hard mode are those good passwords which are within one standard deviation of length of the average hard bad password. The balance in these three game

difficulties has not been tested yet on a player who doesn't know of the formation algorithm.

The game uses procedural generation to generate one password platform on the upper half of the screen and another on the lower half of the screen. On easy and medium difficulties, one platform contains a good password and the other contains a bad password. On the hard difficulty, however, they could both be good or both be bad.

The passwords turn green when landed on, and a plus amount of points (200 in this case, since it is the medium game mode) shows up at the top of the screen and falls. The white text is the amount of time it takes to crack this password, which comes from the ZXCVBN model above.

Gameplay



The first two platforms contain the instructions, and are always the same; the player is instructed to jump on the good passwords and avoid the bad passwords. The game begins with a 90 second timer, although the image above only shows 83.0 seconds remaining.

The main gameplay will be a 90 second timer and a user playing a platformer. The player will be able to move backwards, but that will only waste time. There will be landing platforms in between the bad and good passwords. When a user lands on a good password, they are awarded a certain number of points based on their chosen difficulty (100 for easy, 200 for medium, and 400 for hard, potentially multiplied by streaks), and when they land on a bad password, they lose points (with negative scores allowed). The player can use

the up arrow to jump, the down arrow to accelerate downward, and the left and right arrows to move side to side.

Mechanics

There will be gravity and colliders on the player, and air movement will be allowed. The down arrow key allows for quicker descent. We followed a tutorial online for guidance in the Mechanics of the movement of the player. The videos are below in the Development Section.

Game World

The game world will consist of platforms as well as a background, and these will be sprites freely downloaded from the Unity Asset Store. Specifically, these packages are “Grotto Escape Pack” by Ansimuz for the tiles, and “Dynamic Space Background Lite” by DinV Studio for the space background.

Character

The sprite itself is a character that was freely downloaded and is a spaceship to fit in with the space theme. The hope for this game, although NOT implemented yet, is to have several different sprites for the character that the user could choose between to play this game with.

User Interface

The user interface is simple: the screen to display the game, along with the left and right arrow keys for movement, the up key or space bar for jumping, and the down arrow key for quick descent. These inputs will be managed in the Unity Settings (Edit > Project Settings > Input Manager > Vertical, Horizontal, Jump, etc.). For the iPad users, we are thinking that there will be a joystick to move the character left and right, as well as a button to press for the character to go down, although these features are NOT yet added into the game. In the top left corner is the players current score, and in the top right corner is the timer. We have also implemented the ability for the user to press the "R" key on their keyboard to restart their timer and move them to the beginning with a reset score or the "Esc" key to go back to the Select Difficulty screen (though this does conflict with the Unity full screen mode keys).
Other screens in the game include the start screen, the instruction screen, the screen for selecting the difficulty, and the game over screen where the score is displayed and the user has the option to either play again or quit (which reloads the main page of our website).

Player Feedback

The player feedback is given by the actions after landing on either a good or a bad password. If the player lands on a good password, the password text color will turn a shade of green and the score on the player's screen will be increased. In addition, on the bad password platforms, when the user lands on it, the password text will turn a shade of red (remembering to keep ADA compliance for color-blindness), and the score will reflect the change. Feedback is also given from the ZXCVBN model based on how long it would take to crack the password. This inclusion serves as an immediate feedback on how good or bad a password is and allows the user to learn the makeup of each category.

- Message Board

The message board is accessible from the home/main page. In order to increase knowledge about spam and clicking links that could install viruses/malware, we wanted some type of aspect of messaging. With our client's focus on the social aspect of the game, we decided on a message board. This message board could use a filter package and could also have a report feature that will send messages to an admin email. The instructional portion of this aspect of the game is that there are spam messages created by Dan that are sent saying "click this link" or "download this" and the player gets

points and a badge if they correctly flag that message as spam and do not click the link. Additionally, our goal was to have several different “dummy” users, set up beforehand, to send these messages to the employees. Clicking the link will send the user to a page saying “you clicked on an untrusted link” and warn them about the dangers of clicking unknown links and downloading unknown files. Another feature we have not implemented would be players using the message board to share new badges or have some friendly trash talk to get them more excited about the games. At current functionality, the Message Board cannot be posted to by a regular user. Only SQL SERVER commands to the database server can add messages.

Currently, the message board is set up to give the user 100 points when they successfully click the report button or lose 100 points if they click the reply button or a link within the message.

The challenging aspect for this part of the game is making the message board engaging, thereby encouraging people to read and think about what they receive. One potential idea is giving reward points for various actions. For instance, a player could earn points by successfully entering in the website to check their onyen expiration date, although this has also not been added yet.

- Computer Defense (Not implemented yet)

This game is similar to tower defense games in which enemies are going down a path and if enough cross a threshold, the player loses. In order to win, the player must put “towers” on the sides of the path to defend your computer. The user will see a starting screen of a very small number of slow moving enemies coming down the path. Hovering over the enemy sprites will reveal information about their type of attack method.

An example enemy is as follows: “The coffeehouse hacker hacks into public routers like in a coffee shop and can see everything you type and post!”

To defend against these enemies, the player must place various towers, each of which is suited to particular types of enemies.

One example of a tower is a VPN tower that encrypts your data which will kill the “Coffee House Hacker’s” ability to sniff your data.

Each of the towers has informational text revealing their usage. As the game goes on, the choices of towers will be limited, and the player will have to be smarter about selecting which towers to use in order to most efficiently complete the level. A generic list has been created with the help of our client about types of attacks and their respective defenses.

The goal of this game is to teach a wide range of security practices through the flavor text of the towers and enemies which inform the player of

how to defend themselves. Over time the knowledge would ideally become second nature as players attempt to top the leaderboard. We personally recommend this be the next game implemented after improving the message board.

- Jeopardy (Not implemented Yet)

The client mentioned a trivia-style game where the question set could be changed easily. This enables users to test what they have learned in other games and also enables someone who cannot develop more games or levels to teach more information in a game-like fashion. Upon entry into this game, the categories will be randomly selected from a larger pool of categories and questions. A twist on Jeopardy that increases immersion into the game was discussed with doorways that have the answers and wrong answers produce bad things like snakes, etc and the right answer takes you to the next room. A further future improvement might be including multiplayer (either real time or competing against “ghost players” of real players) to improve the social competitiveness of the overall game.

Content Additions

The hope is for future groups, whether that be Serious Games groups or others, to have the ability to easily implement more games in this design. We built a plug and play type platform that can host several different Unity WebGL games or games built with other methods (e.g. native JavaScript/HTML5 games). If Unity begins to pull support for WebGL based games, creators might need to create other types of games. We want to prevent users from downloading a game file and playing it, as that would be against what we are teaching with clicking on/downloading untrusted links. Future groups will need to build a database for scores (or other uses related to the game), create games, add them to the modular JSON file, and access our server in order to add more games.

Schedule

- **Sprint 1**
  - 2/12/2020
    - Github Created
    - Unity installed
    - Game Design Document Created
    - Servers Discussed and Process Started
- **Sprint 2**
  - 2/26/2020

- Develop Core System
- Move to a game page

- Sprint 3

- 3/18/2020
    - Server setup
    - Begin setting up the Database
    - Fun Facts
    - Game 1 Finished (Platformer)

- Sprint 4

- 4/8/2020
    - Complete the Database set up
    - Begin research on Msg Board and Tower Defense Game
    - Finish up Game 1 (platformer)
        - New Sprite
        - polish the scoring system
        - polish the easy/medium/hard difficulties
        - polish the password generation system

- Sprint 5

- 4/22/2020
    - Finish the Msg Board - get as far as we can
    - Polish off the Platformer game
        - Quit button
        - Edit instructions/overview
    - Leaderboard
    - Account page working

Team website

https://blueteam.web.unc.edu/

- This site contains the Journal for the website, which was updated after every meeting with the meeting's notes. This journal is the place that contains all the different decisions made for this project throughout the semester.
- This website also includes
    - Team member information
    - Client information
    - Scheduled meetings
    - Design Overview

Data Collection and Database Design

MS SQL 2019

- Set up for us and hosted on the server (in the deployment section)
- Tables
    - Game table for each Game
        - Onyen, high score, total times played (current best practice is to call these fields Onyen, Score, and TimesPlayed for standardization across tables)

- "Overall_Leaderboard" table is a sort of user table
    - Onyen, ~~username~~, total high score, list of badges
        - We wanted usernames in this table, but we did not get to it
    - Currently the table only has Onyen and Total High Score
        - We're calculating who gets a badge on the JS end (more down below)
- MessageBoardMessages table is to hold all the messages
    - Timestamped
    - Messages take a standard form (more down below)

- Important note* on the profile page, the client likes usernames, but would like the actual name (pulled from the onyen server) to be displayed in a profile page view or even on a hover of the username
- The MSSQL management studio allows for administrator privileges to be assigned. This is what will be used rather than an administrator table that determines access.
- Badges are currently calculated at runtime by computing the top 15% of each game's scores, but future groups should look into alternate ways of storing badges as the number of games and users grows.

- Don't need to worry about performance as much

Schemas

On the server, the **Microsoft SQL Server Tools 2018** are very helpful especially **Microsoft SQL Server Management Studio 18** (both found from the Windows menu). In SQL SERVER, The FO-COMPSEWEB is the server, and the Databases/BlueTeamTables/Tables is where these tables lie.

Table 1

dbo.MessageBoardMessages	
Onyen	varchar(50) not null
Title	varchar(50) not null
Message	varchar(max) not null
Timestamp	Primary Key, datetime, not null

Important Note*: This is where the messages that Dan will use for the MessageBoard Scores are located.

For the MessageBoard “game”, it is required to do two things for every message,

1. Include two buttons in the message, the “Report” button with the correctButton class tag, and the “Reply” button with the incorrectButton class tag; these buttons will allow the points to be recorded.

2. In the updateMessagesScore.php file, the total number of responses must be incremented for every new message, this is required to prevent users from spamming the buttons and getting a lot of points.

An Example message entry is given below:

Onyen	Title	Message	Timestamp
dlbarker	Free 100 Points!	Click here to receive your 100 points! <button class="correctButton">Report</button><button class="incorrectButton">Reply</button>	2020-04-14 14:33:06.717

Table 2

dbo.MessageBoardScores	
Onyen	Primary Key, varchar(50) not null
Score	Int, not null
AttacksCaught	Int, not null

Table 3

dbo.PasswordPlatformerScores	
Onyen	Primary Key, varchar(50) not null

Score	Int, not null
TimesPlayed	Int, not null

The two above tables are the type of table a user creating another game would add.

Deployment

Game Website

- games.fo.unc.edu
    - Must be accessed while on campus or using a VPN (instructions in the google doc link below)

Server

- Information on connecting to the remote server is from Dan, and listed below

Please look in Virtual Lab and see if you have “F&O Web FO-CompSeWeb” as an available connection.

- https://virtuallab.unc.edu

That should let you RDP to the Web Server. The server is running Windows with IIS and PHP installed.

- http://fo-compseweb.ad.unc.edu/
- http://fo-compseweb.ad.unc.edu/php.php

The space is ours to set up and configure as you want. Dan worked with Gary Wilhelm to get a SSO login system created and connected for use. Dan also caught up with Alan Forrest and got the SQL server installed for that server. Currently, we deploy files to the website through the use of a shared folder. This is a folder which can be accessed on your machine and on the server.

For help setting up the shared folder of the server,

- https://docs.google.com/document/d/1Khp2Ed8MuPUmg2PxEfBLuT5_JX8NevWTOlPjK122f5E/edit

On the server, the D:/intepub/games.fo.unc.edu folder contains the important files for the website. This is the location where the files to be displayed on the website should go. This folder contains a few extra files that were already in the folder, including the aspnet_client folder, the danart folder, php file, and the web.config file. The main file locations used in this project are the shared folder, FO Comp 585 Security folder in the Y: Drive, and the games.fo.unc.edu folder.

To deploy files to the server, first move them to the shared folder. Then, following the instructions above, login to the server using Virtual Lab. Note that access to the server requires 2-Step verification to be set up. Once you are on the server, you can move files from the shared folder to the games.fo.unc.edu folder above. You may sometimes get strange errors

regarding permissions. To circumvent these, first drag files or folders onto the server's desktop, then into the games.fo.unc.edu folder.

Development Documentation

Code - [GitHub Repo]

File system

QUICK START
To familiarize yourself with the code, look first at the **index.html** file in the **BlueTeam/** directory. There are two main JavaScript files that go along with this html page. These can be found in **BlueTeam/static/js** and are **index.js** and **messages.js**. To modify the Password Platformer game, open up PasswordPlatformGame (open the PasswordPlatformerGame folder not the BlueTeam folder!) in Unity and edit away! The next step in our design is to work on completing the message board by adding functionality for users to use it to chat with one another and brag about high scores!

- Where to find the js files to edit
    - **messages.js** contains code that sends AJAX calls to **getMessages.php** and **updateMessagsScore.php**. This would be the place to add calls for users to send their own messages to the database
- Where to add messages
    - Look in the "development" section that explains how tables work.
    - Getting familiar with accessing and adding to the table. (php side)
        - Read the **README.md** in the **sqlconnect/games** folder to learn about the php code and how to get started.
        - Create a php file to insert new strings following our standard into the table with a timestamp so that you can later pull from that table with the already existing php and js code provided to build the messageboard page.

        - Or to add messages you can right click the messages table and “edit top 200 lines” to add more lines manually.
- Cleaning up the report and reply button landing pages. (standard web dev)
    - **message.html** - for the loss of 100 points
    - **goodMessage.html** - for the gain of 100 points

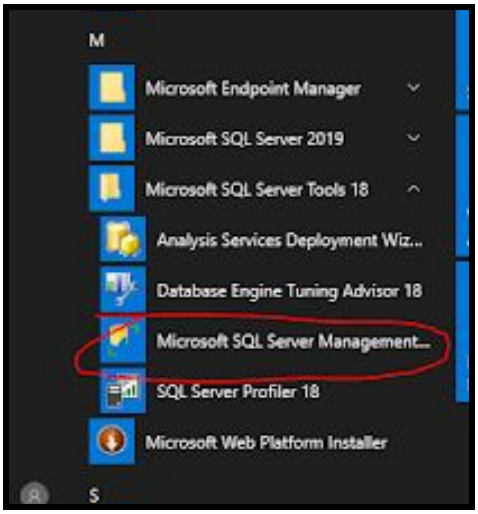
The next thing to do is to develop a second game (maybe Computer Defense)

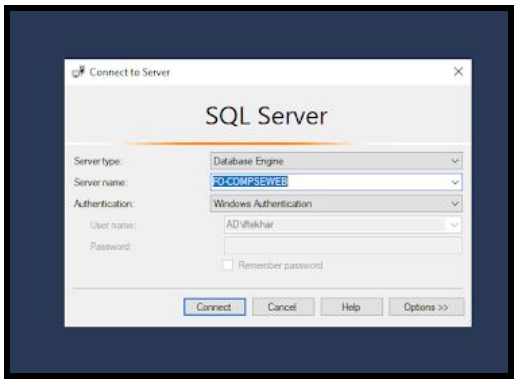
- The **README.md** in the php folder will tell you what you need to add for a standard new game: it shouldn’t be difficult.

The following is an image of the file outline of the repository, and what each file means/helps to do. The code also contains several comments to help the developers.

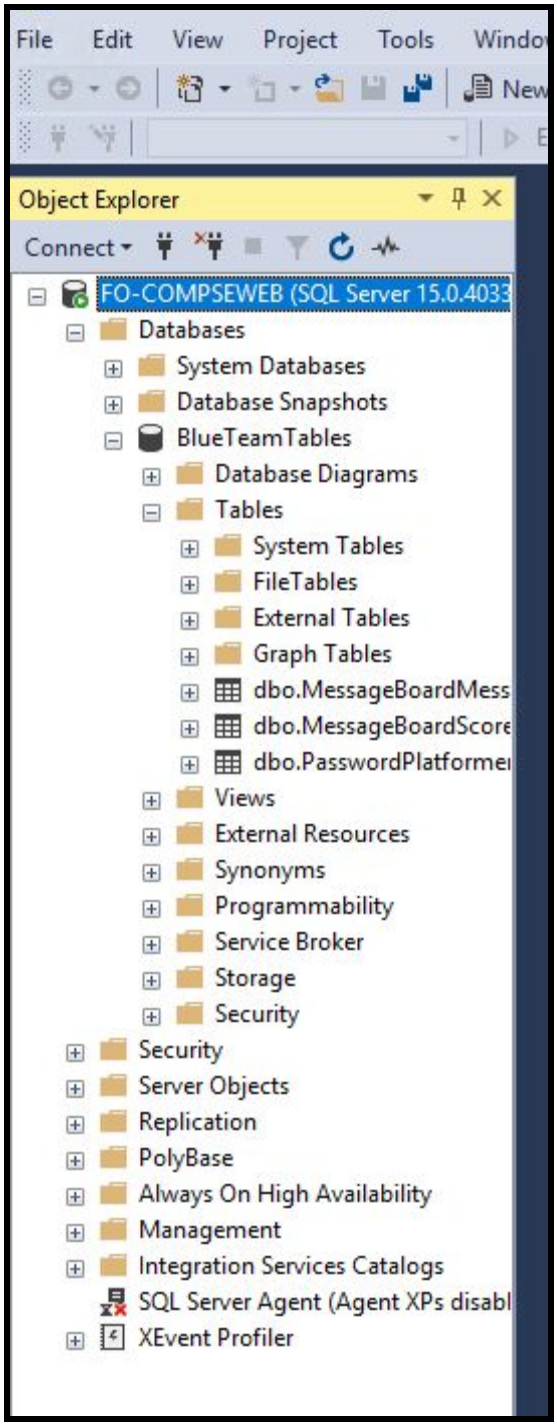
IMPORTANT (PHP and MSSQL issues)

- Before making any SQL queries, make sure to search “SQL SERVER.” This is an incredibly important distinction if you don’t want to get tripped up. Error logs for PHP files are also frustrating. More about that below. To ensure that the SQL server queries aren’t what’s causing an issue, go into Microsoft SQL Server Management Studio 18 on the virtual lab and create a query. Once the query does exactly what you expect, you can insert that into your php code with the appropriate variables and some minor modifications (for example, queries on the MSSQL Management Studio often require the format of [Database].[dbo].[TableName] whereas the SQL queries in the PHP code require only dbo.TableName).
- Server Management

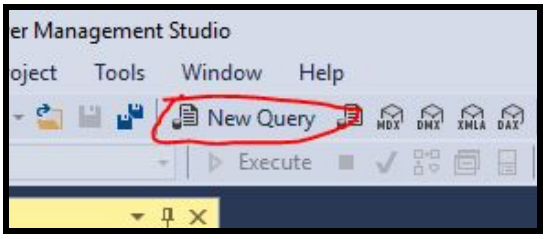




Once you open Microsoft SQL Server Management Tools and click connect you'll find the tables by following the trees down below.

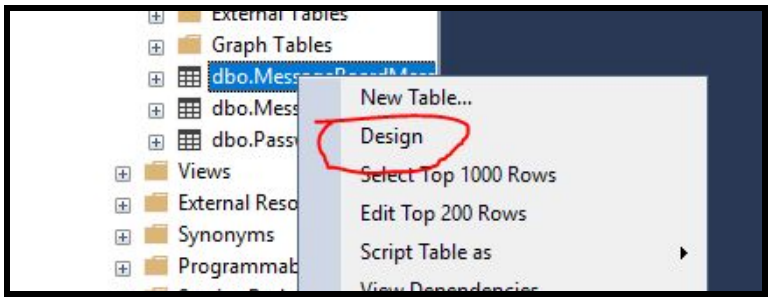


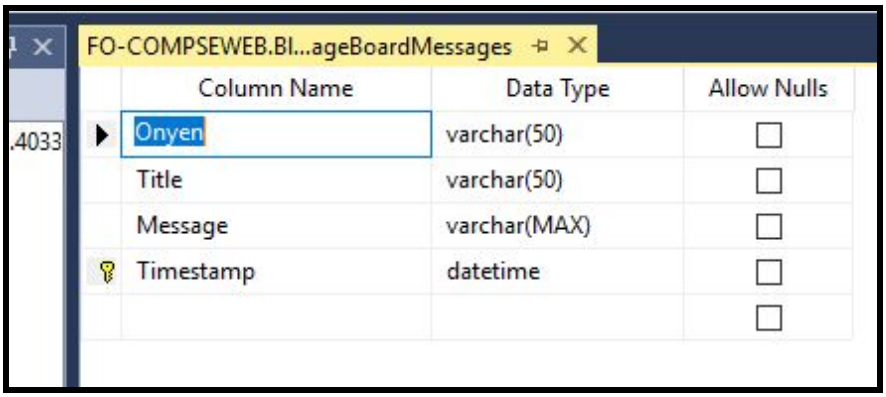
At the top left you'll see a button called "new query" that you can use to test SQL SERVER code.



To modify the table, right click and click design.

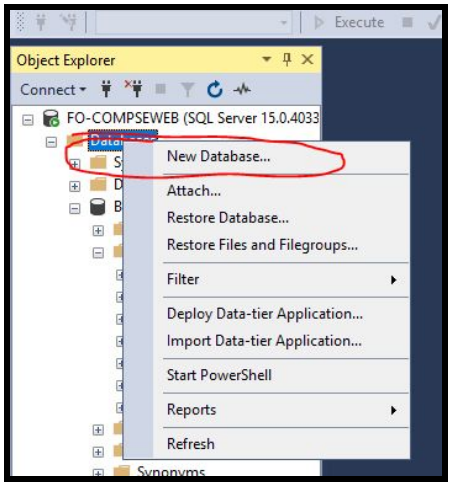
- NOTE. Modifying Tables requires special permissions. Not recommended but if you really need to change something, remember to change the permissions back afterwards.



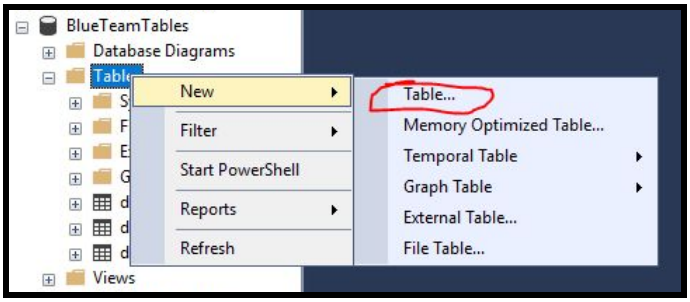


Ctrl + s once done

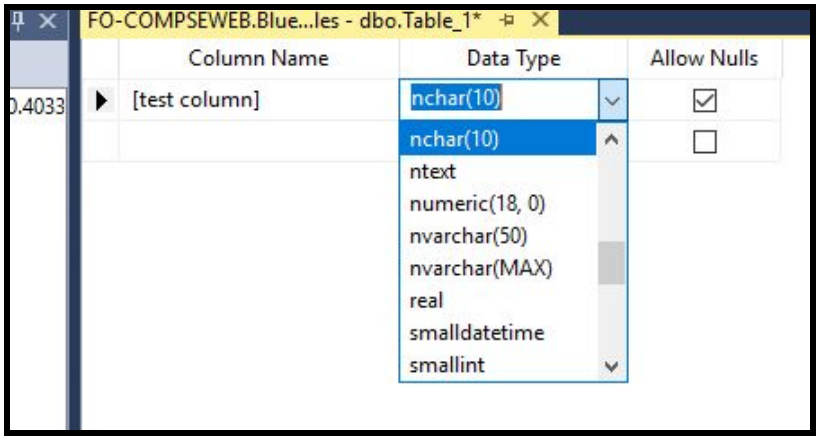
To create a new database, right click “databases” and “new database.” Self explanatory settings after that.



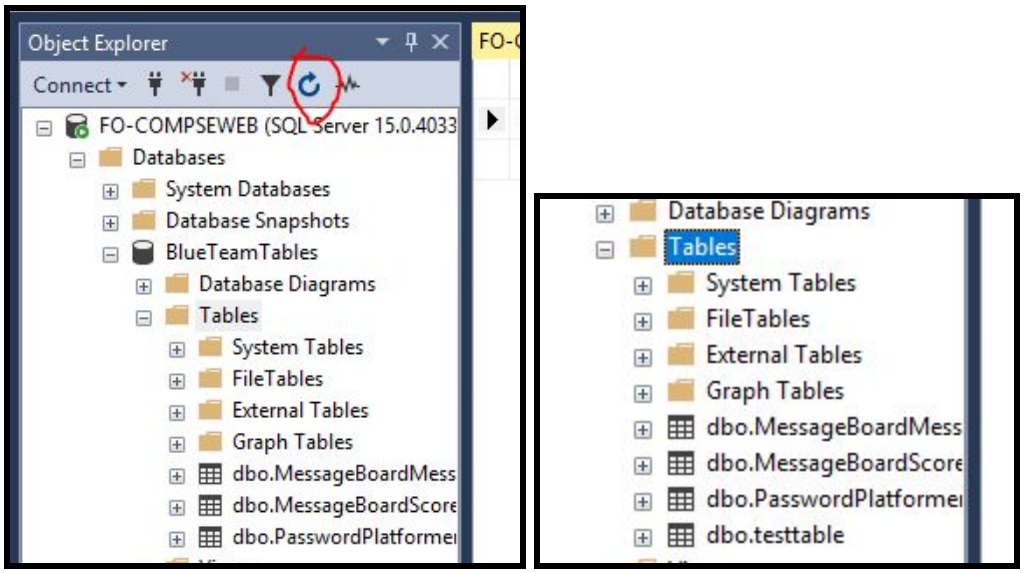
To create a new table:



“Allow nulls” just allow for no value to be had when a column is added. I don’t see a situation where you allow nulls but if you don’t allow nulls, this may mean you need to hardcode default values for new columns. For example, TimesPlayed in the platformer game is created by adding a default 1 if it’s a new user in the table. Otherwise, the request will be rejected.

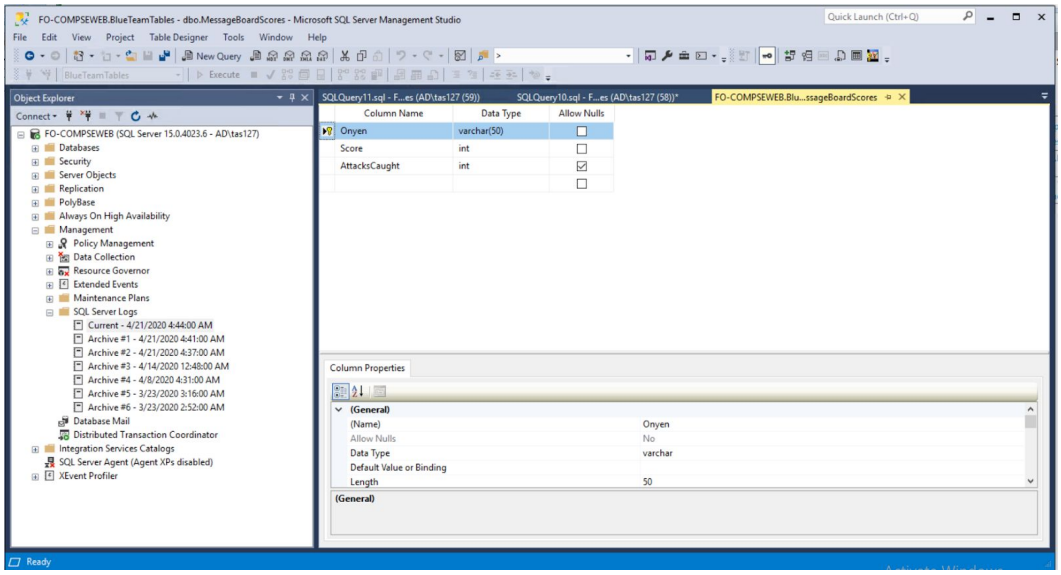


Ctrl + s. It'll ask you to set a table name. BUT you won't see it until you click the refresh button displayed below.



To delete a table, right click the table, delete.

- Debugging
    - There are several methods by which you can debug SQL queries. The most complete way is to locate the log files in MSSQL Management Studio on the server. Once you've logged into the SQL Server (FO-COMPSEWeb), you can find the logs under Management>SQL Server Logs (see the image below for what you should be hoping to see).



- Another debugging option is to add echo statements to your PHP files as breakpoints or to ensure the status of variables. You can then navigate to the PHP file in your browser at https://games.fo.unc.edu/<path-to-php-file>. Your echo statements will then be output on the browser page, and you can see where in the file your code is failing. However, you should know that this method, combined with viewing the logs, will be much faster than random Googling about why MSSQL is not cooperating (for example, maybe you have a tiny syntax error in a SQL query that breaks everything).

BlueTeam/

- **PasswordDictionaries/**
  - **6 .lst files** that will be used to generate passwords
    - These files are taken from the ZXCVBN password internals
- **passwordplatformer/**
    - This is the folder which stores the current build of the Password Platformer game. This build is stored in a separate folder because deploying the entirety of the Unity project to the server is a waste of memory and is prohibitively time-expensive.
    - Either point the Unity build process to this folder or drag the files into this folder when the Build completes.

    - **Build/**
        - **3 .unityweb files**
        - **Build.json**
        - **UnityLoader.js**
    - **TemplateData/**
        - **8 png files**
        - **style.css**
        - **UnityProgress.js**
        - **Favicon.ico** - The icon of the Unity Game
    - **Index.html** - contains the HTML page when the PasswordPlatformer game is clicked on from the homepage.
- **PasswordPlatformerGame/** - This folder contains the Unity files of the Password Platformer game.
    - **Assets/**
        - **CustomScripts/**
            - This folder holds all the scripts which are attached to Unity GameObjects.
            - **gameover/**
                - **GameOver.cs** - This class controls the functionality for game over, including allowing the player to replay the game or quit and takes care of updating their score in the database
            - **startscreen/**
                - **ButtonScript.cs** - This class controls the functionality of the various buttons on the play menu (back or choice of difficulty).
                - **InstructionScript.cs** - This class controls the functionality of the back button on the instruction page.
                - **StartMenu.cs** - This class controls the functionality of the play and instruction buttons on the start screen.
            - **zxcvbn/**
                - We had substantial problems importing the dll file for the C# version of zxcvbn, so this folder holds the code for it, slightly altered to work in

Unity. We worried about this process so you don't have to. In short, though, we cannot use the DLL for the "official" zxcvbn C# port because Unity does not like the embedded text files used in it. Then, the link off that to the official code is broken, so we had to find a different C# version (maybe the same one? Who knows.). After lightly editing this to work with Unity (loading the password files from the Assets>Resources folder through Unity mechanisms), we thought we were off scot-free. Of course, we were wrong. The library had a bug in it which made it produce display times of "Centuries" for bad passwords (the author apparently forgot to implement the reverse password matching). The last update to it was in 2018, and this one bug makes us believe there may be other hidden problems with it. However, after rewriting some of the library, we believe that it is at a point that is workable. It is our hope that you won't ever have to mess with this portion of the game, but if you do, hopefully there's a new, better C# zxcvbn or maybe you can figure out how to call the code of other languages from Unity. Good luck :)

- **BadPlatform.cs** - This class controls what happens when you jump on a bad password (text turns "bad" color, password evaluation falls).
- **BlackHole.cs** - A class which destroys everything with which it comes into contact (used to clean up falling objects).
- **CharacterController2D.cs** - This is a controller file taken from a video (link below in Resources) and adapted to include downward acceleration.
- **Countdown.cs** - A class which governs the timing of the game (currently set to 90 seconds).

- **DifficultyUtility.cs** - This file holds the difficulty enum used by multiple other scripts.
- **Feedback.cs** - This script destroys the Feedback objects once they get below a certain y position.
- **GameController.cs** - A singleton class which serves as a controller for the game, storing such game-wide information as the score and the current user and performing such actions as ending the game.
- **GoodPlatform.cs** - This class controls what happens when you jump on a good password (text turns green, password evaluation falls from screen).
- **InitialTrigger.cs** - This is a script given to the first "normal platform," so the platform generator can know where to start procedurally generating platforms.
- **MaterialController.cs** - This singleton holds the materials and fonts needed by other procedurally generated objects in one place so each individual object does not have to go get them.
- **PasswordGeneration.cs** - A singleton class which takes care of generating passwords. Originally this was done at run time, but now the passwords are loaded from a file, chosen randomly, and passed along to their destination.
- **PlatformGenerator.cs** - A singleton class which serves as a generator for platforms. To perform this duty, it stores the current rightmost point of generation and generates to the right of this. The current method of generation is to generate one password platform on the upper half of the screen and one on the lower half, but this method can be easily changed due to this class's modularity.
- **PlatformTrigger.cs** - A script which generates more platforms when the player collides with its platform.
- **PlayerMovement.cs** - Governs player movement by delegating to the CharacterController2D script.

            - **PortalBottom.cs** - A way for the game to catch the player and teleport them to a particular spot above the screen.
        - **DinV/** - holds the space background assets
        - **Editor/** - holds files relating to the Unity editor
        - **Fonts/** - holds fonts used in the project
        - **Materials/** - holds materials used by various game objects
        - **Passwords/** - holds the text files with passwords of varying difficulties. If you generate more, just drag and drop them here and ensure that they have the same name
        - **Resources/**
        - **Scenes/** - holds the scenes used in the project (see below, currently 5)
        - **Sprites/** - holds the sprites of the player, platforms, and more
    - **Library/**
    - **Packages/**
    - **ProjectSettings/**
- **scripts/**
    - **passwordgeneration/** - This folder contains several different files needed for the password generation that is used in the PasswordPlatformer Game. Run password.py to generate passwords (use the --append argument to add onto existing ones or go without to wipe the slate clean). It also has several text files, including lists of bad passwords or good passwords generated by the password generation process. It also contains the mechanisms to filter profane words from the generation of passwords or from the password lists which are used to generate passwords. Once these password files change, you can drag them into the appropriate folder (Assets/Passwords/) in the Unity project to start using them in the game itself.
- **sqlconnect/**
    - **games/**
        - Note that all PHP files connect to and interact with the database via the well-defined PHP Data Object API. Please view the documentation for further details.

        - **getAccountData.php -** gets the times played and score for each game for a particular user. This data is used in the creation of the Account tab on the homepage.
        - **getMessageLeaderboard.php** - gets the (Onyen, Score) values in descending order of score from the MessageBoardScores table. This file is used to populate the Message Board individual leaderboard
        - **getLeaderboard.php** - gets the overall leaderboard in descending order.
        - **getMessages.php** - This file fetches all the messages in the MessageBoardMessages table
        - **getPasswordLeaderboard.php** - gets the (Onyen, Score) values in descending order of score from the PasswordPlatformerScores table. This file is used to populate the Password Platformer individual leaderboard
        - **SQLConnect.php** - This file connects to the SQL connection, using a Config.php file. This file does not exist in the git repository, but it is on the server. This setup is so that the password is kept more secure.
        - **updateMessagesScore.php** - Connects to the MessageBoardScores table and updates after a link is clicked (increasing or decreasing score based on which link or button is clicked)
        - **updatePlatformerScore.php** - Connects to the PasswordPlatformerScores table and updates after the end screen is reached in the PasswordPlatformerGame
        - **updateOverallLeaderboard.php** - Updates user's score in the overall leaderboard by replacing it with a new cumulative score. Triggered when a player reaches a new high score for a game.
        - **README.md** - Contains all the information one might need to update any php file when adding a new game.
    - **vendor/**
        - **composer/**
            - **6 .php files**
            - **LICENSE**

        - **autoload.php**
            - This is a PHP module which allows us to autoload classes instead of having to use “use” statements in a bunch of places.
            - DO NOT modify these files.
    - **composer.json**
        - Composer is PHP’s package manager (akin to npm or Rust’s cargo). This file is PHP’s version of the package-lock files of npm. Do not modify it unless you know what you are doing.
- **static/**
    - **css/**
        - **index.css** - The main css file for all the html pages listed in the repository
    - **fonts/**
        - **9 different .ttf files** - these files are the downloaded fonts
    - **funFacts/funFacts.txt** - This is the file that Dan will be able to edit on the server to add and remove funFacts with ease, it is fetched by the index.js file.
    - **js/**
        - **Index.js** - This file contains all the javascript functionality of the main page (loading the username, setting up the leaderboard, etc.) and contains the $(document).ready() function in addition to many other helper functions. Thorough comments are included in this file to aid developers in getting started. Any points where values are currently hard coded (for example, when you add a new individual leaderboard) are shown.
        - **messages.js** - This file is used as the Javascript for the index.html page that connects to the database backend and fetches messages from the Message Board
    - **foheader.png** - The image file used as the header on the homepage, taken from the FO website (with permission from Dan)
- **.gitignore** - The .gitignore file contains files that will not be sent to the GitHub repository
- **credits.md** - This credits file gives credit for the sources (for the funFacts and Sprites)

- **games.json** - A JSON file that contains all the information on the current and future games (Coming Soon). This file is fetched to display the carousel at the bottom of the homepage. A file like this defines a modular API where future teams can easily edit and add new games.
- **getAllServer.php**
- **getUser.php** - This file gets the onyen from the server by looking at the PHP $_SERVER array.
- **index.html** - The homepage HTML page that uses files in the static/ folder as scripts and styling.
- **goodMessage.html** - The page that will be brought up when the player clicks on Report and gets a positive number of points in the MessageBoardScore table.
- **messages.html** - The page that will be brought up when the player clicks a bad link, and triggers the loss of points in the MessageBoardScore table
- **README.md** - A file to be used as the markdown file to explain the information in the project on the GitHub

Running locally

First, we must take a (relatively quick) detour on installing Python for the following commands (and for working on the password generation scripts!). If you are on Windows, you can do any of the following:

- https://docs.python.org/3/using/windows.html: Follow the instructions in this webpage to install Python.
- https://docs.anaconda.com/anaconda/install/windows/: Install Python as a part of Anaconda
- https://chocolatey.org/: Install Chocolatey, the Windows package manager (like Linux's apt). Then just run the command "choco install python" on your Command Prompt or Powershell running in administrator mode.
- The Windows Subsystem for Linux: Taylor swears by this method, as it gives you a Linux distro right on your Windows machine, making command-line development a LOT easier. Once you have followed their instructions, you can just open up your Linux box and "sudo apt install python3".

Though Mac comes with Python 2 out of the box, you can follow this link to get Python 3: https://docs.python-guide.org/starting/install3/osx/.
Linux installation of Python is fairly self explanatory: https://docs.python-guide.org/starting/install3/linux/.
To ensure that you have successfully installed Python, open up a command prompt and run the command “python3 --version”.

To locally run this site, run your favorite local web server. A particularly easy one to use is the following command on your command prompt (note that the $ is a stand-in for your prompt string), which needs to be run from the **BlueTeam/** directory:

$ python3 -m http.server

This spins up the built-in Python web server on port 8000 (which, of course, requires you to have Python installed - see above). Another option if you have it installed is to run the built-in PHP web server by running the following:

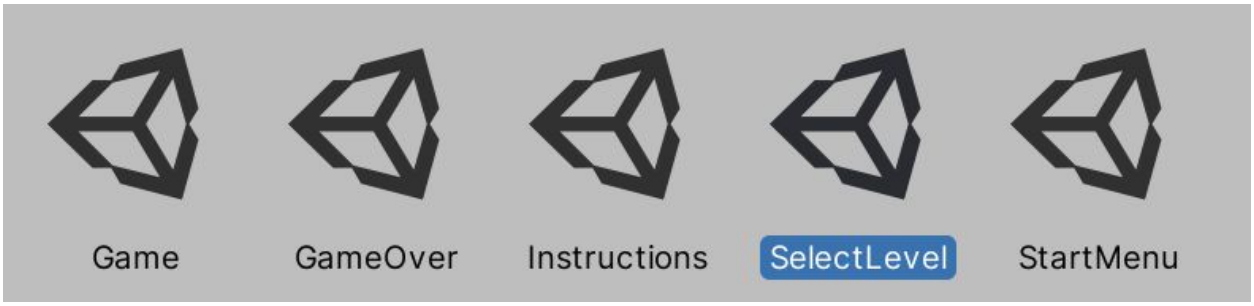
$ php -S localhost:8000

Whichever command you run, you can now access a locally running version of the homepage by navigating to localhost:8000 in your browser.

Game Engine

-Unity (version 2019.3.2f1)

-To get the game up and running in Unity, git clone the repository, open UnityHub and select the PasswordPlatformerGame file (IMPORTANT: not the BlueTeam folder itself, but the subfolder PasswordPlatformerGame). The Scenes are located in the Assets/Scenes folder. There are 5 of them (shown below).



Resources

Several Videos listed below were helpful in the 2D development of this game on Unity, as well as different parts of the design

- Spawning object in Unity [Using Instantiate]
- Unity 5 2D Mobile Game Tutorial - Ep. 5 Platform Spawner
- 2D Movement in Unity (Tutorial)
- How to make a 2D Game in Unity
- SCORE & UI - How to make a Video in Unity (E07)
- Fun Facts Help
- AJAX, PHP, and SQL help
- File that we forked into our project and edited - From Brackeys