SY306 FINAL PROJECT

“CyberMonke”

MIDN 2/C DYLAN LARKIN

MIDN 2/C CHLOE VONDERLINDEN

MIDN 2/C ADRIAN SCHALK

MIDN 2/C DANIELLE CHOI

REQUIREMENTS NOT MET (IF ANY):

N/A

**Team Involvement:**

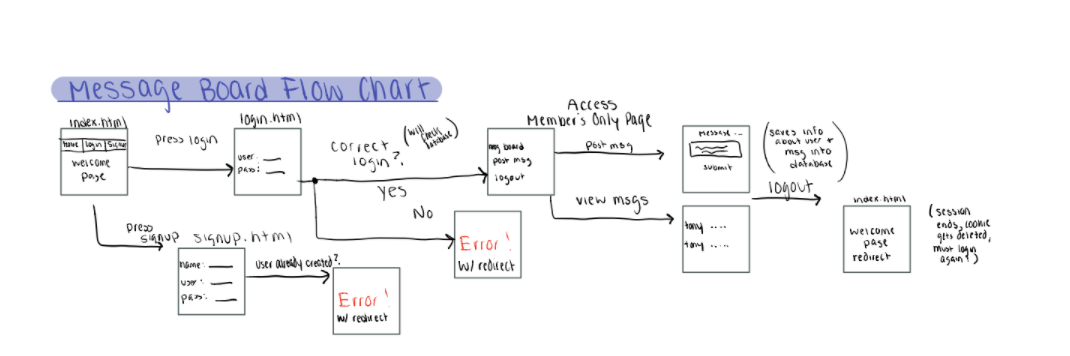
* Dylan Larkin: Wrote the Python, Javascript, and SQL code in order to allow all of the pages to communicate with one another and also tested the code.
* Chloe VonderLinden: Wrote the HTML code and CSS for the public pages, tested all of the code and wrote the project report
* Adrian Schalk: Wrote the HTML code for the member’s only pages
* Danielle Choi: Will implement the javascript regular expressions when we work on the second portion of the project

**Users:**

* Normal:
  + Username: dylan
  + Password: dylan
* Admin:
  + Username: tony
  + Password: tony

**Files Included in the Project**:

* checkLogin.py: connects to database to verify login information
* postMessage.py: stores name, message information, and date/time in database
* readMessageTable.py: posts the messages stored in the database to the html table on the message board
* saveUser.py: saves sign-up information into database so that it can be accessed later when they log in
* 1.jpeg: image file used for giving a nice look to each webpage
* error.html: whenever the user encounters an error for logging in or creating an account that already exists, they will be sent to this page
* index.html: home page with buttons to login/sign-up
* login.html: form to log in to an existing account
* MembersOnlyMessageBoard.html: after logging in, a user can access the message board and read the messages
* MembersOnlyPOST.html: after logging in, a user can post messages to the message board
* signup.html: if the user does not have an account, they will need to create a new, unique account
* Userconfirmation.html: confirmation page to give to a new user after creating a new account successfully. There is a button for them to log in with their new information
* Userexists.html: if the account that the user is trying to create already exists, they will be directed to this error page prompting them to try again
* Proj1.js: javascript file created to control the cookies on the webpage. Cookies can be set, cleared, and deleted from the path. This file also dynamically creates a message board.
* projectMAIN.sql: creates a database that is used throughout the website in order to store login information and messages posted on the message board

**Technical Report Flow Chart**

**Explanation of Flow Chart**

When someone accesses the index.html site, they are instructed to login to their account or to create a new account if they are not a member already. There are three links in the navigation bar to include home, a login page, and a page to create a new account. All of these pages were created with HTML and CSS. If someone is not a member, they will create a new account and that information will be stored in the sql table so that the website will remember their login information. By using python, the information retrieved from the form, the server will communicate with the SQL database called “projectdb”. They can then login with their username and password which will be stored as a cookie on the website. The way that the cookies were created in python is extremely useful. For example, there is a checkLogin function which if there is an incorrect login submitted, the SQL query will return “none” before redirecting the user to an error.html page. However if a correct login is submitted from the form, three cookies will be available on every page on the website (LoggedIn, User, UserType). When the user’s session expires, the cookie will then be deleted and they will need to login again when they access the website. If they select the “remember me” button on the login page, their information should be stored so that they will have a faster login experience in the future. After someone inputs a correct login combination, they will have access to the member’s only html website. Here the cookie is still stored so that they can switch between the various pages without having to log back in. The member’s only pages allow the user to post messages and view the messages on the message board which are dynamically created through the SQL “Messages” table. When someone posts a message, the python script retrieves the information from the form and saves it into the sql database. In the database, the name of the user, message, and date/time will all be logged. When the user is done with the message board, they can log out. When the user logs out, their session expires which deletes all cookies and redirects them to the index.html welcome page. They must then log back in with the correct username and password.

**SQL Script**

CREATE DATABASE IF NOT EXISTS `projectdb` /\*!40100 DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci \*/ /\*!80016 DEFAULT ENCRYPTION='N' \*/;

USE `projectdb`;

-- MySQL dump 10.13 Distrib 8.0.22, for Linux (x86\_64)

--

-- Host: 127.0.0.1 Database: projectdb

-- ------------------------------------------------------

-- Server version 8.0.23-0ubuntu0.20.04.1

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!50503 SET NAMES utf8 \*/;

/\*!40103 SET @OLD\_TIME\_ZONE=@@TIME\_ZONE \*/;

/\*!40103 SET TIME\_ZONE='+00:00' \*/;

/\*!40014 SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0 \*/;

/\*!40014 SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0 \*/;

/\*!40101 SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='NO\_AUTO\_VALUE\_ON\_ZERO' \*/;

/\*!40111 SET @OLD\_SQL\_NOTES=@@SQL\_NOTES, SQL\_NOTES=0 \*/;

--

-- Table structure for table `Messages`

--

DROP TABLE IF EXISTS `Messages`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `Messages` (

`Username` varchar(45) NOT NULL,

`Message` varchar(100) NOT NULL,

`Timestamp` datetime DEFAULT CURRENT\_TIMESTAMP

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `Messages`

--

LOCK TABLES `Messages` WRITE;

/\*!40000 ALTER TABLE `Messages` DISABLE KEYS \*/;

INSERT INTO `Messages` VALUES ('tony','test','2021-04-28 17:58:24'),('tony','hello','2021-04-28 19:11:24');

/\*!40000 ALTER TABLE `Messages` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Table structure for table `Users`

--

DROP TABLE IF EXISTS `Users`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!50503 SET character\_set\_client = utf8mb4 \*/;

CREATE TABLE `Users` (

`Username` varchar(45) NOT NULL,

`Pass` varchar(45) NOT NULL,

`UserRole` varchar(7) DEFAULT 'Regular',

`Name` varchar(45) NOT NULL,

PRIMARY KEY (`Username`),

UNIQUE KEY `Username\_UNIQUE` (`Username`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_0900\_ai\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

--

-- Dumping data for table `Users`

--

LOCK TABLES `Users` WRITE;

/\*!40000 ALTER TABLE `Users` DISABLE KEYS \*/;

INSERT INTO `Users` VALUES ('dylan','dylan','Regular','dylan'),('greg','greg','Regular','greg'),('test','test','Regular','test'),('tony','tony','Regular','tony');

/\*!40000 ALTER TABLE `Users` ENABLE KEYS \*/;

UNLOCK TABLES;

--

-- Dumping routines for database 'projectdb'

--

/\*!40103 SET TIME\_ZONE=@OLD\_TIME\_ZONE \*/;

/\*!40101 SET SQL\_MODE=@OLD\_SQL\_MODE \*/;

/\*!40014 SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS \*/;

/\*!40014 SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS \*/;

/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;

/\*!40111 SET SQL\_NOTES=@OLD\_SQL\_NOTES \*/;

-- Dump completed on 2021-04-28 19:15:07

**Part 2**

**Team Involvement:**

* Dylan Larkin: Worked on writing the code to store passwords securely. Helped with error checking too.
* Chloe VonderLinden: Worked on writing the code to securely store passwords in the database. Also updated the signup files so that a user must create their password following the rules listed on the website. Limited the input size on the message board. Error checked the code. Also updated the project report.
* Adrian Schalk: completed the “secret/CSRF” token approach/captcha to prevent cross-site request forgery attacks
* Danielle Choi: Came up with the javascript function for the REGEX for making a password for a new account.

In order to prevent any attacks on the database, we found a way to securely store the passwords in the database. In order to do this we used SHA256 and a salt in order to create random passwords displayed in the database. By doing this, if someone were to obtain access to our database, they would not be able to hack into anyone’s account. Additionally, we limited the input size on the message board so that no one could execute malicious code on our website. We also have included defenses against HTML/Javascript injection attacks as well as preventing clickjacking. We also implemented a CAPTCHA in order to prevent any cross-site request forgery attacks. When someone loads the page, the CAPTCHA creates a random number that the user must enter in order to confirm that they are human.