Technical Style Guide

# #INCLUDES

<iostream> //cin, cout

<string> //strings

<vector> //vectors

<fstream> //reading, writing to file

<cstdio> //eof

# Containers

// Will store all the booking information

struct **BookingInformation** - string location;  
 - int date, d, e;

// Will store all the users login information and reference the booking information

struct **User** - string username, password;  
 - int permission;  
 - **BookingInformation** booking;

// Will store all the stock information

struct **Stock** - string bloodGroup;  
 - int available;

// Will store all the requests status and reference the user that made the request, and the stock information.

struct **Request** - **User** user;  
 - **Stock** stock;  
 - int status;

vector<**User**> **users** // vector of User struct

vector<**Stock**> **totals** // vector of User struct

vector<**Request**> **requests** // vector of User struct

User\* **loggedinUser**; // LoggedinUser struct pointer

# Global Variables

string **filepathUsersLogin** // location of the usersLogin.txt file

string **filepathBookings** // location of the bookingAppointment.txt file

string **filepathStock** // location of the stock.txt file

string **filepathRequest** //location of the requests.txt file

string **message** // Pass messages to display to the user

string **line** // For reading each line of a file

int **d** // Identifies if the user has an appointment

int **e** //Identifies if the user has passed an eligibility test

bool **toApprove** // Set to true if there are stock requests for the admin to check

# Functions

bool **login()**  
// Returns true if the username and password entered match. Returns false after 3 wrong attempts. Stores users information in User vector/struct.

*Local variables – login()*

string username //variable for users username input

string password //variable for users password input

size\_t i //variable to reference which user is being checked against

void **loginRedirection ()**  
// Depending on the logged in users permission level: 0 - Admin, 1 - Hospital, 2 - Donor; Redirects the user to pages they are allowed to access, or boots user out of system if they don’t have permission

*Local variables – loginRedirection()*

int tries //variable to count number of attempts user has logging in

void adminMenu();

// Alerts Admin if there is any requests to review. Allows Admin to view stock and stock requests. Allows Admin to view, edit and delete appointments. Allows Admin to view, edit and delete users

*Local variables – adminMenu()*

char input //variable for user to input choice in menu

void getAllLogins();

// Gets all the logins from the filepathUsersLogin. Sets them all to be a users[] vector

*Local variables – getAllLogins()*

string temp //variable store permission value out of file before stoi()

User **temp\_user** //Initializing new User struct called temp\_user

void viewAllLogins();

// Allows an Admin to View all logins in the system. Admin can enter a username to edit the login

*Local variables – viewAllLogins()*

string input //variable for user to input choice in menu

void editUser(string user);

// Gets username from input parameter. Finds the vector struct with that username and allows user to change password or delete the user. Rewrites file with all users

*Local variables – editUser()*

string tempBlood //variable to store value user enters as bloodGroup to check

char choice //variable for user to input choice in menu

bool checkIfUsernameExists(string tempName);

// Checks if the username passed already exists in the usersLogin file. Returns true if the username already exists, Returns false if the username doesn't exist.

*Local variables – checkIfUsernameExists()*

bool **ans** //variable to store true if the username exists, false if it doesn’t

bool checkIfBloodGroupExists(string tempBlood);

*// Checks if the bloodgroup passed exists in the stock file or matches na. Returns true if the bloodgroup already exists or matches na, Returns false if the bloodgroup doesn't exist.*

*Local variables – checkIfBloodGroupExists()*

bool **ans** //variable to store true if the username exists, false if it doesn’t

void getAllBookings();

// Gets all the bookings from the filepathBookings. Checks if the name in the booking matches a users name. If it matches, adds booking details to users[] vector

*Local variables – getAllBookings()*

string temp //variable store date value out of file before stoi()

string tempName //variable store name value out of file to compare against existing users

void viewAllBookings();

// Allows an Admin to View all bookings in the system. Admin can then add new bookings, or edit bookings

*Local variables – viewAllBookings()*

string input //variable for user to input choice in menu

void hospitalMenu();

// Allows hospital user to view and request stock

*Local variables – hospitalMenu()*

char input //variable for user to input choice in menu

void getAllStock();

// Gets all the stock from the filepathStock. Sets them all to be a totals[] vector

*Local variables – getAllStock()*

string temp //variable store number of available stock value out of file before stoi()

void viewAllStock();

// Displays all stock in the system. Directs hospitals to requestStock. Directs admin to viewAllRequests

void requestStock();

// Allows a hospital to enter a new request for stock in the system

*Local variables – requestStock()*

string input //variable for user to input choice in menu

int amount //variable for user to input amount of stock requesting

void viewAllRequests();

// Allows an Admin to View all requests in the system. Admin can then edit the status of any request

*Local variables – viewAllRequests()*

string input //variable for user to input choice in menu

void getAllRequests();

// Gets all the requests from the filepathRequests. Sets them all to be a requests[] vector

*Local variables – getAllRequests()*

string temp //variable store number of available stock value out of file before stoi()

Request **temp\_request** //Initializing new Request struct called temp\_request

void rego();

//rego code is the switch case for the main menu

*Local variables – rego()*

int selection //variable for user to input

void hospital();

//registerhospital within admin perimeters

*Local variables – hospital()*

ofstream **file** //variable to reference the file

string **tempName** //variable to store inputted username

string **tempPass** //variable to store inputted password

void donor();

//File Handling for Donor registration part

*Local variables – donor()*

ofstream **file** //variable to reference the file

string **tempName** //variable to store inputted username

string **tempPass** //variable to store inputted password

string **tempBlood** //variable to store inputted bloodGroup

void welcome();

//Donors main menu with switch case

*Local variables – welcome()*

int input//variable for user to choose from switch case

void eligibilityTest();

//donor can choose to do this test, however they cannot book without doing this first

*Local variables – eligibilityTest()*

Char **a, b & op** //variables to help users with answering questions and choosing their next steps

String **array**//variable to help loop through questions

void bookAppt();

//users then can use this option if they’re eligible

*Local variables – bookAppt()*

char b, c & op //variables to assist in decision making, or if user hasn’t done eligibility test first

ofstream **file** //stores users booking info

void viewAppt();

//users can always view their appt

*Local variables – viewAppt()*

Char **a, b & op** //variables in case user hasn’t booked an appt or done eligibility test

void editAppt();

//users can edit their appt

*Local variables – editAppt()*

ofstream **file** //users new info gets stored in and replaces previous

void deletefile(string user);

//users can cancel their appt

*Local variables – deletefile()*

char //variable to ask user if they want to cancel

ofstream **file** //removes booking info