**FUNCTION userContinue()**

PROMPT user to continue

WAIT for user input

**END FUNCTION**

**FUNCTION displayTreatmentLength()**

IF condition exists THEN

DISPLAY treatment length

IF no treatment available THEN

DISPLAY No Treatment Available

ELSE IF treatment is permanent THEN

DISPLAY  Forever

ELSE

CALCULATE years and months from treatment length

DISPLAY years if applicable

DISPLAY months if applicable

ELSE

DISPLAY  Treatment Length: N/A

**END FUNCTION**

**FUNCTION calculateCost()**

CREATE an empty list of costs

IF daily cost is not zero THEN

CALCULATE monthly cost from daily cost

CALCULATE yearly cost from daily cost

CALCULATE weekly cost from daily cost

ADD daily, weekly, monthly, and yearly costs to the list

ELSE

ADD zero values for daily, weekly, monthly, and yearly costs to the list

RETURN the list of costs

**END FUNCTION**

**FUNCTION findConditionByID(conditionID)**

SEARCH through conditions for matching ID

IF found, RETURN the condition

IF not found, RETURN nullptr

**END FUNCTION**

**FUNCTION addCondition(conditionID)**

FIND condition by ID

IF condition exists, ADD it to patient conditions

**END FUNCTION**

**FUNCTION removeCondition(conditionID)**

FOR each condition in patient conditions DO

IF condition ID matches, REMOVE the condition

END FOR

**END FUNCTION**

**FUNCTION hasCondition(conditionID)**

CHECK if any condition matches the given ID in patient conditions

IF found, RETURN true

ELSE, RETURN false

**END FUNCTION**

**FUNCTION doesChemotherapy()**

CHECK if any condition ID is 3 or 4 in patient conditions

IF found, RETURN true

ELSE, RETURN false

**END FUNCTION**

**FUNCTION hashPassword(password)**

COMPUTE the SHA256 hash of the password

CONVERT the hash to a hexadecimal string

RETURN the hexadecimal string

**END FUNCTION**

**FUNCTION displayUserDetails(user)**

CLEAR the console

DISPLAY basic user information (role, username, age, doctor, nurse)

IF user has conditions THEN

INITIALIZE total cost variables

FOR each condition in the user's conditions DO

DISPLAY condition details

ADD the costs for this condition to the totals

END FOR

DISPLAY total accumulated costs

END IF

**END FUNCTION**

**FUNCTION displayConditions()**

CLEAR the console

FOR each condition in the list of conditions DO

DISPLAY condition ID and name

DISPLAY treatment length for the condition

END FOR

WAIT for the user to continue

**END FUNCTION**

**FUNCTION updateUsersCSV()**

OPEN the CSV file for writing

IF the file cannot be opened THEN

DISPLAY an error message and exit the function

WRITE the header to the CSV file

FOR each user in the users list DO

WRITE user details (ID, username, password, age, role, doctor name, nurse name)

WRITE each condition ID for the user in quotes

END FOR

IF any error occurs during writing THEN

DISPLAY an error message

**END FUNCTION**

**FUNCTION isValidPassword(password)**

INITIALIZE flags for lowercase, uppercase, digit, and special character

FOR each character in the password DO

CHECK if the character is lowercase, uppercase, digit, or special

UPDATE corresponding flags

END FOR

RETURN true if all conditions (lowercase, uppercase, digit, special character) are met

ELSE, return false

**END FUNCTION**

**FUNCTION hasWhitespace(str)**

CHECK if any character in the string is a whitespace

RETURN true if whitespace is found, else return false

**END FUNCTION**

**FUNCTION containsSpecialCharacters(username)**

FOR each character in the username DO

IF the character is not alphanumeric and not an underscore

RETURN true

END FOR

RETURN false

**END FUNCTION**

**FUNCTION accessPatientsDetails(doctorView)**

CLEAR the console

SET done flag to false

WHILE done is false DO

IF doctorView is true THEN

DISPLAY list of patients assigned to the logged doctor

END IF

PROMPT user to enter patient ID or 0 to cancel

GET user input for patient ID

IF patient ID is not 0 THEN

CALL changePatientsDetails function with the given patient ID

ELSE

SET done flag to true (exit loop)

END WHILE

**END FUNCTION**

**FUNCTION calculateAverageAges()**

INITIALIZE variables for average smoking age, smoker count, average cancer age, cancer count, and patients with both conditions

FOR each patient in users DO

IF the patient is a  Patient  THEN

INITIALIZE flags for smoker and cancer conditions

FOR each condition in the patient's conditions DO

GET the condition ID

IF condition ID is in the smoking range THEN

ADD patient's age to average smoking age

INCREMENT smoker count

SET smoker flag to true

END IF

IF condition ID is in the cancer range THEN

ADD patient's age to average cancer age

INCREMENT cancer count

SET cancer flag to true

END IF

IF patient has both smoking and cancer conditions THEN

INCREMENT the count of patients with both conditions

END IF

END FOR

END IF

END FOR

DISPLAY average smoking age (if smokers exist)

DISPLAY average cancer age (if cancer patients exist)

DISPLAY number of patients with both smoking and cancer conditions

**END FUNCTION**

**FUNCTION calculateCostForEveryCondition()**

FOR each condition in conditions DO

DISPLAY condition name and treatment details

CALL and display treatment length for the condition

CALL calculateCost for the condition and store the result in costs

DISPLAY daily, weekly, monthly, and yearly costs for the condition

END FOR

**END FUNCTION**

**FUNCTION displayStatistics()**

CLEAR the console

CALL calculateAverageAges to display average ages

DISPLAY an empty line

CALL userContinue to prompt user to continue

**END FUNCTION**

**FUNCTION changePatientsDetails(doctorView, ID)**

CLEAR the console

FOR each user in users DO

IF user matches the ID and doctor has access or doctor view is not required THEN

CLEAR the console

DISPLAY patient details

DISPLAY options to edit conditions, add new condition, or exit

GET user input for selected option

SWITCH selected option:

CASE 1: Edit current conditions

IF patient has no conditions THEN

DISPLAY  Patient is healthy

ELSE

DISPLAY existing conditions

GET input for condition to edit

IF condition is cured THEN

REMOVE condition from patient

UPDATE user details in CSV

DISPLAY success message

ELSE

DISPLAY possible new conditions

GET input for new condition

REMOVE old condition, add new condition to patient

UPDATE user details in CSV

DISPLAY success message

END IF

CASE 2: Add new condition

IF patient is undergoing chemotherapy THEN

DISPLAY  Cannot combine Chemotherapy with other treatments

ELSE

DISPLAY possible new conditions

GET input for new condition

IF patient already has condition THEN

DISPLAY  Patient already has this condition

ELSE

ADD the new condition to the patient

UPDATE user details in CSV

DISPLAY success message

END IF

CASE 3: Exit

CLEAR the console

EXIT loop

DEFAULT: Display  Invalid Option

END SWITCH

EXIT loop after processing selected patient

END FOR

**END FUNCTION**

**FUNCTION importUsers()**

OPEN USERDETAILS file for reading

IF file cannot be opened THEN

DISPLAY error message

RETURN false

END IF

CLEAR the existing user list

DECLARE variables for ID, username, password, age, role, doctorName, nurseName, and condition

WHILE there are more rows in the CSV DO

READ each field in the row

IF the ID is  ID  THEN

SKIP this row (header)

END IF

REMOVE surrounding quotes from the condition field if they exist

SPLIT the condition field by commas and convert each piece to an integer, adding them to the conditions list

CONVERT ID and Age from strings to integers, skipping invalid rows

ADD the new user to the user list

END WHILE

CLOSE the CSV file

RETURN true

**END FUNCTION**

**FUNCTION registerNewUser(newUserRole)**

CLEAR the input buffer

DEFINE variables for new user details (name, password, age)

SET isValidUsername to false

WHILE the username is invalid DO

ASK for the new user's name

IF the username is too short, contains special characters, or has whitespace THEN

DISPLAY  Username is invalid

CONTINUE to next iteration

END IF

ASSUME the username is unique

FOR each existing user in the user list DO

IF the new username matches an existing one THEN

DISPLAY  Username is already in use

SET isValidUsername to false

BREAK the loop

END IF

END FOR

END WHILE

SET isValidPasswordBool to false

WHILE the password is invalid DO

ASK for the new user's password

IF the password is too short, contains whitespace, or is invalid THEN

DISPLAY  Password is invalid

CONTINUE to next iteration

END IF

SET isValidPasswordBool to true

HASH the password

END WHILE

ASK for the new user's age

DEFINE variables for doctor name, nurse name, and user ID

SET default condition to  Healthy  (condition ID = 0)

ADD the new user to the user list

DISPLAY  User Successfully Registered

UPDATE the user details in the CSV file

CALL userContinue() to proceed

**END FUNCTION**

**FUNCTION getConditions()**

OPEN the CONDITIONS file

IF the file cannot be opened THEN

DISPLAY error message and return false

END IF

DEFINE variables for reading file data (ID, condition, treatment, frequency, cost, treatmentLength)

WHILE there are still lines in the file DO

READ each field from the CSV into the respective variables

IF the ID is  ID  THEN

SKIP this row (header row)

END IF

CONVERT ID to integer (intID)

SET treatmentLength to 999 if it is  N/A , 0 if it is  0 , otherwise convert to integer

CONVERT cost to double (doubleCost)

CREATE a new Condition object with the parsed data

ADD the new Condition object to the conditions list

END WHILE

RETURN true

**END FUNCTION**

**FUNCTION readLoginDetails(inputtedUsername, inputtedPassword)**

FOR each user in users DO

IF the username matches AND the hashed password matches THEN

SET loggedUser to the current user

RETURN true

END IF

END FOR

RETURN false

**END FUNCTION**

**FUNCTION assignPatientToDoctor**

CLEAR console screen

DISPLAY message  Patients that currently do not have a doctor assigned to them

FOR each user in users

IF user is a Patient and does not have a doctor

DISPLAY user ID and user name

END IF

END FOR

PROMPT the user to enter a Patient ID (or 0 to cancel)

READ patientID input

IF patientID is not 0

FOR each user in users

IF user matches the patientID

Assign loggedUser as the doctor's name for the patient

UPDATE user details in CSV

DISPLAY success message  Patient Successfully Assigned

BREAK from loop

END IF

END FOR

END IF

CALL userContinue to wait for user input

**END FUNCTION**

**FUNCTION displayMenu**

CLEAR console screen

DISPLAY welcome message with loggedUser's username

GET loggedUser's role

SET menuOption to 0

SET logout flag to false

WHILE logout is false

CLEAR console screen

DISPLAY menu options for the user

PROMPT user to select an option

READ menuOption input

SWITCH menuOption

CASE 1:

DISPLAY loggedUser details

CALL userContinue to wait for user input

BREAK

CASE 2:

IF loggedUser is a Doctor

CALL accessPatientsDetails with doctorView set to true

ELSE

CALL showAccessDenied with message  Only Doctors can access this.

END IF

BREAK

CASE 3:

IF loggedUser is a Pharmacist

CALL accessPatientsDetails with doctorView set to false

ELSE

CALL showAccessDenied with message  Only Pharmacists can access this.

END IF

BREAK

CASE 4:

CALL displayStatistics

BREAK

CASE 5:

IF loggedUser is a Doctor, Nurse, or Pharmacist

DISPLAY options for registering a new role (Doctor or Nurse)

READ newRole input

SWITCH newRole

CASE 1:

CALL registerNewUser with  Doctor

BREAK

CASE 2:

CALL registerNewUser with  Nurse

BREAK

CASE 3:

BREAK

DEFAULT:

CALL showAccessDenied with message  Invalid Role

END SWITCH

ELSE

CALL showAccessDenied with message  Only Doctors, Nurses, or Pharmacists can register a new Doctor/Nurse.

END IF

BREAK

CASE 6:

IF loggedUser is a Pharmacist

CALL registerNewUser with  Pharmacist

ELSE

CALL showAccessDenied with message  Only Pharmacists can register new Pharmacists.

END IF

BREAK

CASE 7:

IF loggedUser is a Doctor

CALL assignPatientToDoctor

ELSE

CALL showAccessDenied with message  Only Doctors can assign patients.

END IF

BREAK

CASE 8:

SET logout to true

BREAK

DEFAULT:

CALL showAccessDenied with message  Invalid option. Please try again.

BREAK

END SWITCH

END WHILE

**END FUNCTION**

**FUNCTION displayLogin**

SET validLogin to false

WHILE validLogin is false

CLEAR console screen

DECLARE variables username and password to store user input

DISPLAY prompt  Enter Username:

READ username input

DISPLAY prompt  Enter Password:

READ password input

CALL readLoginDetails with username and password

IF readLoginDetails returns true

SET validLogin to true

CALL displayMenu to display the user menu

ELSE

SET again to 'n'

DISPLAY prompt  Invalid Login Details. Try Again (y/n):

READ again input

IF again is not 'y'

SET validLogin to true

ELSE

CLEAR the input buffer

END IF

END IF

END WHILE

**END FUNCTION**

**FUNCTION displayHomeMenu**

SET exit to false

DECLARE menuOption as integer

WHILE exit is false

CLEAR console screen

DISPLAY  Welcome to the NHS treatment system!

DISPLAY menu options (Login, Patient Registration, Exit)

IF user input is valid (1, 2, or 3)

SWITCH menuOption

CASE 1:

CALL displayLogin function to initiate login process

BREAK

CASE 2:

CALL registerNewUser with  Patient  as argument to register a new patient

BREAK

CASE 3:

SET exit to true

BREAK

END SWITCH

ELSE

CLEAR input error

DISCARD invalid input to clear the buffer

END IF

END WHILE

**END FUNCTION**

**FUNCTION main**

CALL getConditions to load all available conditions from the conditions file

CALL importUsers to load all users from the users file

CALL displayHomeMenu to start the user interaction

**END FUNCTION**