**Assignment**

**BIT 205**

**Object-Oriented Programming in C++**



**Student ID: E9001344**

Table of Contents

[Source Code 1](#_Toc90854481)

[main.cpp 1](#_Toc90854482)

[CTIS.h 14](#_Toc90854483)

[User.cpp 18](#_Toc90854484)

[CovidTest.cpp 20](#_Toc90854485)

[TestKit.cpp 23](#_Toc90854486)

[TestCentre.cpp 26](#_Toc90854487)

[CentreOfficer.cpp 30](#_Toc90854488)

[Patient.cpp 33](#_Toc90854489)

[CTIS.cpp 37](#_Toc90854490)

[Test Data CTIS 46](#_Toc90854491)

[Register Test Centre 1 46](#_Toc90854492)

[Register Test Centre Manager 1 46](#_Toc90854493)

[Test Centre Manager Login 1 48](#_Toc90854494)

[Register Test Centre 1 50](#_Toc90854495)

[Register Test Centre 2 52](#_Toc90854496)

[Register Test Centre Manger 2 52](#_Toc90854497)

[Test Centre Manager Login 2 54](#_Toc90854498)

[Register Test Centre 2 58](#_Toc90854499)

[Record Tester 1 60](#_Toc90854500)

[Test Centre Manager Login 1 60](#_Toc90854501)

[Record Tester 1 62](#_Toc90854502)

[Record Tester 2 64](#_Toc90854503)

[Test Centre Manager Login 2 64](#_Toc90854504)

[Record Tester 2 66](#_Toc90854505)

[Manage Kit Stock 69](#_Toc90854506)

[Test Centre Manager Login 1 69](#_Toc90854507)

[Record New Stock 1 71](#_Toc90854508)

[Record New Stock 2 73](#_Toc90854509)

[Test Centre Manager Login 2 75](#_Toc90854510)

[Record New Stock 3 77](#_Toc90854511)

[Update Stock 79](#_Toc90854512)

[Record New Test 81](#_Toc90854513)

[Tester Login 1 81](#_Toc90854514)

[Record New Test 1 83](#_Toc90854515)

[Record New Test 2 86](#_Toc90854516)

[Record New Test 3 90](#_Toc90854517)

[Tester Login 2 93](#_Toc90854518)

[Record New Test 4 96](#_Toc90854519)

[Update Result 99](#_Toc90854520)

[Tester Login 1 99](#_Toc90854521)

[Update Result 1 101](#_Toc90854522)

[Tester Login 2 103](#_Toc90854523)

[Update Result 2 104](#_Toc90854524)

[Update Result 3 105](#_Toc90854525)

[View Test History 107](#_Toc90854526)

[Login Patient 1 107](#_Toc90854527)

[View Test History 1 108](#_Toc90854528)

[Login Patient 2 109](#_Toc90854529)

[View Test History 2 110](#_Toc90854530)

[Login Patient 3 111](#_Toc90854531)

[View Test History 3 112](#_Toc90854532)

[Generate Test Report 113](#_Toc90854533)

[Test Centre Manager Login 1 113](#_Toc90854534)

[View Test History Report 1 115](#_Toc90854535)

[Test Centre Manager Login 2 116](#_Toc90854536)

[View Test History Report 2 118](#_Toc90854537)

[Tester Login 1 119](#_Toc90854538)

[View Test History Report 3 120](#_Toc90854539)

[Tester Login 2 121](#_Toc90854540)

[View Test History Report 4 122](#_Toc90854541)

# Source Code

## main.cpp

*/\*\*  
 \* The CTIS Console Program implements  
 \* the Covid Testing Information System application which consists of various menus:  
 \* Add new user, manage kit stock, manage test records, and view test history.  
 \* Student ID: E1900344  
 \* Date: 16 December 2021  
 \* C++ version: C++14  
 \* IDE : CLion  
 \* The program works well without errors or interruptions.  
 \*/*#include <iostream>  
  
#include "CTIS.h"  
  
#include <vector>  
  
using namespace std;  
  
*// Function prototype*char mainMenu();  
*// Menu & login for Admin*void adminLoginMenu();  
char adminMenu();  
void registerTestCentreManager(char &position);  
void registerTestCentre();  
void manageKitStock();  
  
*// Menu & login for Tester*char testerMenu();  
void testerLoginMenu();  
void recordNewTest();  
void updateTestResult();  
  
*// Menu & login for Patient*void patientLoginMenu();  
char patientMenu();  
  
*// Supporting method*string generateCentreID();  
string generateKitID();  
string generateTestID();  
string date();  
  
*// Global variable*int iCentreName;  
int cID = 0;  
int kID = 0;  
int tID = 0;  
  
*// Global object*CTIS CTISSystem;  
CentreOfficer \*testerLogin;  
CentreOfficer \*centreManagerLogin;  
TestCentre centreKitStock;  
  
  
*/\*\*  
 \* main method used to  
 \* provide various menu for User:  
 \* Register Test Centre Manager,  
 \* Test Centre Manager Login,  
 \* Tester Login,  
 \* and Patient Login.  
 \*/*int main() {  
 char option;  
 char tempPosition;  
 do {  
 option = mainMenu();  
 switch (option) {  
 case '1':  
 tempPosition = 'C';  
 cout << "\t\tCreate your Test Centre Manager account\n" << endl;  
 registerTestCentreManager(tempPosition);  
 cout << "\nRegister success! A Test Centre Manager account is created." << endl;  
 break;  
 case '2':  
 adminLoginMenu();  
 break;  
 case '3':  
 testerLoginMenu();  
 break;  
 case '4':  
 patientLoginMenu();  
 break;  
 case 'e':  
 case 'E':  
 cout << "\t\tApplication closed" << endl;  
 break;  
 default:  
 cout << "Invalid choice! Try again" << endl;  
 }  
 }while (option != 'E' && option != 'e');  
 return 0;  
}  
  
*/\*\*  
 \* mainMenu support method used to  
 \* show various menu for User:  
 \* Register Test Centre Manager,  
 \* Test Centre Manager Login,  
 \* Tester Login,  
 \* and Patient Login.  
 \*/*char mainMenu() {  
 char option = 0;  
 cout << "\n\n\t\tCovid Testing Information System (CTIS) Menu\n" << endl;  
 cout << "1. Register test centre manager" << endl;  
 cout << "2. Test centre manager login" << endl;  
 cout << "3. Tester login" << endl;  
 cout << "4. Patient login" << endl;  
 cout << "Press (E/e) to exit" << endl;  
 cout << "\nMenu of choices: ";  
 cin >> option;  
 cout << endl;  
 return option;  
}  
  
*/\*\*  
 \* patientLoginMenu support method used to  
 \* provide menu for Patient.  
 \*/*void patientLoginMenu() {  
 cout << "\t\tUse your Patient account\n" << endl;  
 string usrname, passw;  
 cout << "Enter username : ";  
 cin >> usrname;  
 cout << "Enter password : ";  
 cin >> passw;  
 *// Find Patient object based on username and password* Patient \*patientLogin = CTISSystem.patientLoginValidation(usrname,passw);  
 if(patientLogin == nullptr)  
 cout << "\nCan't find your Patient account" << endl;  
 *// Patient found* else {  
 cout << "\nWelcome, " << patientLogin->getName() << endl;  
 char option;  
 do {  
 option = patientMenu();  
 switch (option) {  
 case '1':  
 cout << "\t\tTest history" << endl;  
 patientLogin->viewTestHistory();  
 break;  
 case 'e':  
 case 'E':  
 cout << "Back to main menu" << endl;  
 break;  
 default:  
 cout << "Invalid choice! Try again" << endl;  
 }  
 }while (option != 'E' && option != 'e');  
 }  
}  
  
*/\*\*  
 \* patientMenu support method used to  
 \* show menu for Patient.  
 \* @return a char of selected menu.  
 \*/*char patientMenu() {  
 char choice = 0;  
 cout << "\n\t\tPatient Menu\n" << endl;  
 cout << "1. View test history" << endl;  
 cout << "Press (E/e) to log out" << endl;  
 cout << "\nMenu of choices: " << flush;  
 cin >> choice;  
 cout << endl;  
 return choice;  
}  
  
*/\*\*  
 \* recordNewTest support method used to  
 \* allow Tester administer test for Patient  
 \*/*void recordNewTest() {  
 Patient tempPatient;  
 cout << "\t\tRegister Patient\n" << endl;  
 int iPatientType;  
 string usrname, passw, name, symptoms;  
 *// Input patient information* cout << "Enter username : ";  
 cin >> usrname;  
 cout << "Enter password : ";  
 cin >> passw;  
 cout << "Enter name : ";  
 getline(cin >> ws, name);  
 cout << "\n\t\tPatient type\n" << endl;  
 tempPatient.showAllPatientType();  
 cout << "\nSelect patient type: ";  
 cin >> iPatientType;  
 cout << "Enter symptoms : ";  
 getline(cin >> ws, symptoms);  
  
 *// Get patient type based on selected index* string patientType = tempPatient.getSelectedPatientType()[iPatientType - 1];  
 *// Create Patient object* Patient newPatient(usrname, passw, name, patientType, symptoms);  
  
 *// Create CovidTest object* CovidTest newCovidTest;  
 newCovidTest.setTestDate(date());  
 newCovidTest.setTestId(generateTestID());  
 newCovidTest.setStatus("pending");  
  
 *// Add if Patient object is not in vector* if(CTISSystem.findPatient(usrname)) {  
 CTISSystem.addPatient(newPatient);  
 }  
  
 *// Get Patient object in vector* Patient \*patientTest;  
 patientTest = CTISSystem.getPatient(usrname);  
  
  
 cout << "\n\t\tTest Centre\n" << endl;  
 CTISSystem.showAllTestCentres();  
 cout << "Select centre name: ";  
 cin >> iCentreName;  
  
 *// Get Test Centre object based on the selected index* TestCentre \*centreCovidTest;  
 centreCovidTest = &CTISSystem.getTestCentre()[iCentreName-1];  
  
 TestKit \*testKitCovidTest;  
 TestKit \*testKitCovidTest1;  
 CentreOfficer \*centreManagerTest;  
  
 *// Centre Manager must record test kit first* if(centreCovidTest->getTestKits().empty())  
 cout << "\nCentre Manager haven't recorded the test kit" << endl;  
 else {  
 string kitID;  
 cout << "\n\t\tAvailable Test Kit" << endl;  
 centreCovidTest->showAvailableTestKit();  
 cout << "\nEnter kid ID: ";  
 cin >> kitID;  
 while(!centreCovidTest->kitIDValidation(kitID)) {  
 cout << "\nInvalid kit ID! Try again" << endl;  
 cout << "Enter kid ID: ";  
 cin >> kitID;  
 }  
 *// Find Test Kit object* centreManagerTest = CTISSystem.getCentreManager(centreCovidTest->getCentreName());  
 testKitCovidTest = CTIS::getTestKitByManager(\*centreManagerTest, kitID);  
 testKitCovidTest1 = CTISSystem.getTestKit(kitID);  
 }  
  
 *// Patient has not taken test* if(patientTest->getCovidTests().empty()) {  
 patientTest->addCovidTest(newCovidTest);  
 testerLogin->addCovidTest(newCovidTest);  
 testKitCovidTest->addCovidTest(newCovidTest);  
 *// Set available stock of Test Kit* testKitCovidTest->setAvailableStock(testKitCovidTest->getAvailableStock()-1);  
 testKitCovidTest1->setAvailableStock(testKitCovidTest1->getAvailableStock()-1);  
 cout << "\nRegister and record success! A new test is created for " << patientTest->getName() << endl;  
 }  
 else {  
 cout << "\nPatient has taken test"<< endl;  
 cout << "Only patient types and symptoms are updated" << endl;  
 patientTest->setPatientType(tempPatient.getSelectedPatientType()[iPatientType - 1]);  
 patientTest->setSymptoms(symptoms);  
  
 patientTest->addCovidTest(newCovidTest);  
 testerLogin->addCovidTest(newCovidTest);  
 testKitCovidTest->addCovidTest(newCovidTest);  
  
 *// Set available stock of Test Kit* testKitCovidTest->setAvailableStock(testKitCovidTest->getAvailableStock()-1);  
 testKitCovidTest1->setAvailableStock(testKitCovidTest1->getAvailableStock()-1);  
 cout << "\nRecord success! A new test is created for " << patientTest->getName() << endl;  
 }  
}  
  
*/\*\*  
 \* updateTestResult support method used to  
 \* allow Tester to update information of the test.  
 \*/*void updateTestResult() {  
 string testID;  
 string result;  
 if(CTISSystem.getNumOfPendingCovidTest()) {  
 CTISSystem.showAllCovidTestID();  
 cout << "Enter test ID: ";  
 cin >> testID;  
 while(!CTISSystem.covidTestIDValidation(testID)) {  
 cout << "\nInvalid test ID! Select another" << endl;  
 cout << "Enter test ID: ";  
 cin >> testID;  
 }  
 *// Show Test details* cout << "\n\t\tCovid test details\n";  
 cout << \*CTISSystem.getCovidTestOfTester(testID) << endl;  
  
 *// Show additional details* Patient \*patientInfo;  
 patientInfo = CTISSystem.getPatientByCovidTest(testID);  
 cout << "\n\t\tPatient details\n" << endl;  
 cout << "Patient name: " << patientInfo->getName() << endl;  
 cout << "Patient type: " << patientInfo->getPatientType() << endl;  
 cout << "Patient symptoms: " << patientInfo->getSymptoms() << endl;  
 cout << "\n\t\tTest Centre details\n" << endl;  
 cout << CTISSystem.getTestCentreByCovidTest(testID)->getCentreName() << endl;  
 cout << "\n\t\tTest Kit details\n" << endl;  
 cout << CTISSystem.getTestKitByCovidTest(testID)->getTestName() << endl;  
  
 cout << "\n\t\tResult test\n" << endl;  
 cout << "1. Positive" << endl;  
 cout << "2. Negative" << endl;  
 cout << "\nSelect result: ";  
 cin >> result;  
 if(result == "1")  
 result = "Positive";  
 else if(result == "2")  
 result = "Negative";  
  
 *// Update information* CTISSystem.getPatientCovidTest(testID)->setStatus("completed");  
 CTISSystem.getPatientCovidTest(testID)->setResult(result);  
 CTISSystem.getPatientCovidTest(testID)->setResultDate(date());  
  
 CTISSystem.getCovidTestOfTester(testID)->setStatus("completed");  
 CTISSystem.getCovidTestOfTester(testID)->setResult(result);  
 CTISSystem.getCovidTestOfTester(testID)->setResultDate(date());  
  
 CTISSystem.getCovidTet(testID)->setStatus("completed");  
 CTISSystem.getCovidTet(testID)->setResult(result);  
 CTISSystem.getCovidTet(testID)->setResultDate(date());  
 cout << "\nThe test result is updated" << endl;  
 }  
 else  
 cout << "No test results to update" << endl;  
  
}  
  
*/\*\*  
 \* testerLoginMenu support method used to  
 \* provide various menu for tester.  
 \*/*void testerLoginMenu() {  
 cout << "\t\tUse your Tester account\n" << endl;  
 string usrname, passw;  
 cout << "Enter username: ";  
 cin >> usrname;  
 cout << "Enter password: ";  
 cin >> passw;  
 *// Find Tester object based on username and password* testerLogin = CTISSystem.testerLoginValidation(usrname, passw);  
 if(testerLogin == nullptr)  
 cout << "\nCan't find your Tester account" << endl;  
 *// Tester found* else {  
 cout << "\nWelcome, " << testerLogin->getName() << endl;  
 char choice;  
 do {  
 choice = testerMenu();  
 switch (choice) {  
 case '1':  
 recordNewTest();  
 break;  
 case '2':  
 updateTestResult();  
 break;  
 case'3':  
 cout << "\t\tTest history report" << endl;  
 testerLogin->viewTestHistory();  
 break;  
 case 'e':  
 case 'E':  
 cout << "Back to main menu" << endl;  
 break;  
 default:  
 cout << "Invalid choice! Try again" << endl;  
 }  
 }while (choice != 'E' && choice != 'e');  
 }  
}  
  
*/\*\*  
 \* testerMenu support method used to  
 \* show various menu for tester.  
 \* @return a char of selected menu.  
 \*/*char testerMenu() {  
 char choice = 0;  
 cout << "\n\t\tTester Menu\n" << endl;  
 cout << "1. Record new test" << endl;  
 cout << "2. Update test result" << endl;  
 cout << "3. View test history report" << endl;  
 cout << "Press (E/e) to log out" << endl;  
 cout << "\nMenu of choices: " << flush;  
 cin >> choice;  
 cout << endl;  
 return choice;  
}  
  
*/\*\*  
 \* adminLoginMenu support method used to  
 \* provide various menu for admin.  
 \*/*void adminLoginMenu() {  
 cout << "\t\tUse your Test Centre Manager account\n" << endl;  
 string usrname, passw;  
 cout << "Enter username : ";  
 cin >> usrname;  
 cout << "Enter password : ";  
 cin >> passw;  
 *// Find Centre Manager object based on username and password* centreManagerLogin = CTISSystem.adminLoginValidation(usrname, passw);  
 if(centreManagerLogin == nullptr)  
 cout << "\nCan't find your Test Centre Manager account" << endl;  
 *// Centre Manager found* else {  
 cout << "\nWelcome, " << centreManagerLogin->getName() << endl;  
 char option;  
 char tempPosition;  
 do {  
 option = adminMenu();  
 switch (option) {  
 case '1':  
 registerTestCentre();  
 break;  
 case '2':  
 tempPosition = 'T';  
 cout << "\t\tCreate your Tester account\n" << endl;  
 registerTestCentreManager(tempPosition);  
 cout << "\nRegister success! A Tester account is created." << endl;  
 break;  
 case '3':  
 manageKitStock();  
 break;  
 case '4':  
 cout << "\t\tTest history report" << endl;  
 centreManagerLogin->viewTestHistory();  
 break;  
 case 'e':  
 case 'E':  
 cout << "Back to main menu" << endl;  
 break;  
 default:  
 cout << "Invalid choice! Try again" << endl;  
 }  
 }while (option != 'E' && option != 'e');  
 }  
}  
  
*/\*\*  
 \* adminLoginMenu support method used to  
 \* show various menu for admin.  
 \* @return a char of selected menu.  
 \*/*char adminMenu() {  
 char choice = 0;  
 cout << "\n\t\tAdmin Menu\n" << endl;  
 cout << "1. Register test centre" << endl;  
 cout << "2. Register tester" << endl;  
 cout << "3. Manage test kit stock" << endl;  
 cout << "4. View test history report" << endl;  
 cout << "Press (E/e) to log out" << endl;  
 cout << "\nMenu of choices: " << flush;  
 cin >> choice;  
 cout << endl;  
 return choice;  
}  
  
*/\*\*  
 \* manageKitStock support method used to  
 \* allow manage kit stock of test kits.  
 \*/*void manageKitStock() {  
  
 cout << "\t\tManage Test Kit Stock\n" << endl;  
 *// Test Center must be saved to Center Manager first* if(!centreManagerLogin->getTestCentres().empty()) {  
 int choice;  
 cout << "1. Record new stock" << endl;  
 cout << "2. Update stock" << endl;  
 cout << "\nMenu of choices: ";  
 cin >> choice;  
 if(choice == 1) {  
 int iTestKit = 0, stock;  
 string testKit\_name;  
 *// Show new stock of Test Kits* if(centreKitStock.getNumOfNewStocks() > 0) {  
 cout << endl;  
 centreKitStock.showNewStocks();  
 cout << "\nSelect test kit : ";  
 cin >> iTestKit;  
 cout << "Enter stock : ";  
 cin >> stock;  
 *// Get test kit name by selected user* testKit\_name = centreKitStock.getNewStock()[iTestKit-1].getTestName();  
 *// Create test kit object* TestKit testKit(testKit\_name, generateKitID(), stock);  
 *// Set new stock status to ID* centreKitStock.getNewStock()[iTestKit-1].setKitId(testKit.getKitId());  
 centreKitStock.getNewStock()[iTestKit-1].setAvailableStock(testKit.getAvailableStock());  
 *// Record to vector TestKit in CTIS and TestCentre* centreManagerLogin->getTestCentre()->addTestKit(testKit);  
 string centreID = centreManagerLogin->getTestCentre()->getCentreId();  
 CTISSystem.getTestCentreByID(centreID)->addTestKit(testKit);  
 cout << "\nThe new stock received and recorded to the Test Centre" << endl;  
 }  
 else  
 cout << "\nAll new test kit stock has been received" << endl;  
 }  
 else if (choice == 2){  
 string kit\_id;  
 int updateStock;  
 *// Centre Manager must record stock Test Kit first* if(!centreManagerLogin->getTestCentre()->getTestKits().empty()) {  
 cout << "\n\t\tUpdate Kit Stock" << endl;  
 *// Show all Test Kit* centreManagerLogin->getTestCentre()->showAllTestKit();  
 cout << "\nEnter kit ID: ";  
 cin >> kit\_id;  
  
 *// Validation kit ID entered based on vector in Test Centre* while(centreManagerLogin->getTestCentre()->findTestKit(kit\_id) == nullptr) {  
 cout << "\nInvalid kit ID! Try another" << endl;  
 cout << "Enter kit ID: ";  
 cin >> kit\_id;  
 }  
 *// Get TestKit object based on kit ID* TestKit \*getTestKit = centreManagerLogin->getTestCentre()->findTestKit(kit\_id);  
  
 cout << "Update stock: ";  
 cin >> updateStock;  
  
 *// Update stock of TestKit object in CTIS and TestCentre vector* getTestKit->setAvailableStock(updateStock);  
 CTISSystem.getTestKit(kit\_id)->setAvailableStock(updateStock);  
 string temp\_testName = getTestKit->getTestName();  
 cout << "\nThe stock of " << temp\_testName << " is updated to " << getTestKit->getAvailableStock() << endl;  
 }  
 else  
 cout << "\nYou haven't recorded the stock kit" << endl;  
 }  
 }  
 else  
 cout << "\nYou haven't recorded the test centre" << endl;  
}  
  
*/\*\*  
 \* registerTestCentre support method used to  
 \* allow Test Centre Managers / Admins register test centre  
 \* and recorded the Test Centre to centre managers.  
 \*/*void registerTestCentre() {  
 cout << "\t\tRegister test centre\n" << endl;  
 *// Show approved Test Centres and Centre Managers has not recorded Test Centre.* if(centreManagerLogin->getTestCentres().empty()) {  
 if(CTISSystem.getNumOfApprovedTestCentres() > 0) {  
 *// Show Test Centres* CTISSystem.showApprovedTestCentres();  
 cout << "\nSelect centre name: ";  
 cin >> iCentreName;  
 *// Validation select centre name* while(iCentreName >=3) {  
 cout << "\nInvalid centre name! select another" << endl;  
 cout << "Select centre name: ";  
 cin >> iCentreName;  
 }  
 *// Find centre name by index* string centreName = CTISSystem.getTestCentre()[iCentreName-1].getCentreName();  
 *// Create object Test Centre* TestCentre testCentre(generateCentreID(), centreName);  
 *// Set Test Center approved status in CTIS to ID* CTISSystem.getTestCentre()[iCentreName-1].setCentreId(testCentre.getCentreId());  
 *// Record Test Centre to Centre Manager based on Centre Manager account login* centreManagerLogin->addTestCentre(testCentre);  
 cout << "\nThe test centre is created and recorded for the centre manager" << endl;  
 }  
 else  
 cout << "Sorry, no test center to be approved" << endl;  
 }  
 else  
 cout << "Admin is only responsible for one test centre!" << endl;  
}  
  
*/\*\*  
 \* registerTestCentreManager support method used to  
 \* allow register Test Centre Manager to CTIS System.  
 \* @param tempPosition reference to differentiate between Centre Manager or Tester.  
 \*/*void registerTestCentreManager(char &tempPosition) {  
 *// Centre officer dynamic object* CentreOfficer \*centreOfficer = nullptr;  
 string usrname, passw, name, post;  
 cout << "Enter username : ";  
 cin >> usrname;  
 *// Admin username cannot be the same* if(tempPosition == 'C') {  
 while(CTISSystem.findCentreManager(usrname)) {  
 cout << "\nThat admin username is taken. Try another." << endl;  
 *// input new username* cout << "Enter new username : ";  
 cin >> usrname;  
 }  
 }  
 *// Tester username cannot be the same* else if(tempPosition == 'T') {  
 while(CTISSystem.findTester(usrname)) {  
 cout << "\nThat tester username is taken. Try another." << endl;  
 *// input new username* cout << "Enter new username : ";  
 cin >> usrname;  
 }  
 }  
  
 cout << "Enter password : ";  
 cin >> passw;  
  
 cout << "Enter name : ";  
 getline(cin >> ws, name);  
  
 if(tempPosition == 'C') {  
 *// Set position to Centre Manager* post = "Centre manager";  
 *// Create object Centre Officer* centreOfficer = new CentreOfficer(usrname, passw, name, post);  
 *// Add to the vector* CTISSystem.addCentreManager(\*centreOfficer);  
 }  
 else if(tempPosition == 'T') {  
 *// Set position to Tester* post = "Tester";  
 *// Create object Centre Officer* centreOfficer = new CentreOfficer(usrname, passw, name, post);  
 *// Add to the vector Centre Officer in CTIS* CTISSystem.addTester(\*centreOfficer);  
 }  
 *// Deallocate memory space* delete centreOfficer;  
}  
  
*/\*\*  
 \* Determine the length of centreID.  
 \* @return a string of centreID with one/two positive int or zero.  
 \*/*string generateCentreID() {  
 cID++;  
 string centre\_id = to\_string(cID).length() == 1 ?("C00" + to\_string(cID))  
 : to\_string(cID).length() == 2 ? ("C0" + to\_string(cID))  
 : to\_string(cID);  
 return centre\_id;  
}  
  
*/\*\*  
 \* Determine the length of kitID.  
 \* @return a string of kitID with one/two positive int or zero.  
 \*/*string generateKitID() {  
 kID++;  
 string kit\_ID = to\_string(kID).length() == 1 ?("K00" + to\_string(kID))  
 : to\_string(kID).length() == 2 ? ("K0" + to\_string(kID))  
 : to\_string(kID);  
 return kit\_ID;  
}  
  
*/\*\*  
 \* Determine the length of testID.  
 \* @return a string of testID with one/two positive int or zero.  
 \*/*string generateTestID() {  
 tID++;  
 string id = to\_string(tID).length() == 1 ?("T00" + to\_string(tID))  
 : to\_string(tID).length() == 2 ? ("T0" + to\_string(tID))  
 : to\_string(kID);  
 return id;  
}  
  
*/\*\*  
 \* Generate current time using The strftime() function.  
 \* @return a string of date and time from calendar.  
 \*/*string date() {  
 *// get time based on current time system* time\_t localTime;  
 tm \* currTime;  
 char dt[100];  
 time(&localTime);  
 currTime = localtime(&localTime);  
 *// convert to string* strftime(dt, 50, "%F %R", currTime);  
 return dt;  
}

## CTIS.h

#ifndef **E1900344\_CTIS\_H**#define **E1900344\_CTIS\_H**#include <vector>  
#include <ostream>  
  
using namespace std;  
  
class User {  
public:  
 User();  
 User(string usrname, string passw, string name);  
 ~User();  
 const string &getUsername() const;  
 void setUsername(const string &newUsername);  
 const string &getPassword() const;  
 void setPassword(const string &newPassword);  
 const string &getName() const;  
 void setName(const string &newName);  
 friend ostream &operator<<(ostream &os, const User &user);  
 virtual void viewTestHistory() = 0 ;  
private:  
 string username;  
 string password;  
 string name;  
};  
  
class CovidTest {  
private:  
 string testID;  
 string testDate;  
 string result;  
 string resultDate;  
 string status;  
  
public:  
 CovidTest();  
 CovidTest(string testId, string testDate, string result, string resultDate,  
 string status);  
 ~CovidTest();  
 const string &getTestId() const;  
 void setTestId(const string &testId);  
 const string &getTestDate() const;  
 void setTestDate(const string &testDate);  
 const string &getResult() const;  
 void setResult(const string &result);  
 const string &getResultDate() const;  
 void setResultDate(const string &resultDate);  
 const string &getStatus() const;  
 void setStatus(const string &status);  
 friend ostream &operator<<(ostream &os, const CovidTest &test);  
};  
  
class TestKit{  
private:  
 string testName;  
 string kitID;  
 int availableStock;  
 vector<CovidTest> covidTests;  
  
public:  
 TestKit();  
 TestKit(string testName, string kitId, int availableStock);  
 ~TestKit();  
 const string &getTestName() const;  
 void setTestName(const string &testName);  
 const string &getKitId() const;  
 void setKitId(const string &kitId);  
 int getAvailableStock() const;  
 void setAvailableStock(int availableStock);  
 friend ostream &operator<<(ostream &os, const TestKit &kit);  
 void addCovidTest(CovidTest &covidTest);  
 CovidTest \*getCovidTest();  
 vector<CovidTest> getCovidTests() const;  
};  
  
class TestCentre {  
private:  
 string centreId;  
 string centreName;  
 vector<TestKit> newStocks;  
 vector<TestKit> testKits;  
 vector<TestKit>::iterator t;  
public:  
 TestCentre();  
 TestCentre(string centreId, string centreName);  
 ~TestCentre();  
 const string &getCentreId() const;  
 void setCentreId(const string &centreId);  
 const string &getCentreName() const;  
 void setCentreName(const string &centreName);  
 friend ostream &operator<<(ostream &os, const TestCentre &centre);  
 void addNewStock(TestKit &testKit);  
 void showNewStocks();  
 int getNumOfNewStocks();  
 TestKit \*getNewStock();  
 void addTestKit(TestKit &testKit);  
 vector<TestKit> getTestKits() const;  
 void showAllTestKit();  
 TestKit \*getTestKit();  
 TestKit \*findTestKit(const string& kitID);  
 bool kitIDValidation(const string& kitID);  
 void showAvailableTestKit();  
  
};  
  
class CentreOfficer : public User{  
private:  
 string position;  
 vector<TestCentre> testCentres;  
 vector<CovidTest> covidTests;  
 vector<CovidTest>::iterator c;  
 vector<TestCentre>::iterator t;  
  
public:  
 CentreOfficer();  
 CentreOfficer(const string &usrname, const string &pass, const string &name, string position);  
 ~CentreOfficer();  
 const string &getPosition() const;  
 void setPosition(const string &position);  
 friend ostream &operator<<(ostream &os, const CentreOfficer &officer);  
 void viewTestHistory() override;  
 void addTestCentre(TestCentre testCentre);  
 vector<TestCentre> getTestCentres() const;  
 TestCentre \*getTestCentre();  
 void addCovidTest(CovidTest &covidTest);  
 vector<CovidTest> getCovidTests() const;  
 CovidTest \*getCovidTest();  
};  
  
class Patient : public User{  
private:  
 string patientType;  
 string symptoms;  
 vector<string> patientTypes;  
 vector<CovidTest> covidTests;  
 vector<CovidTest>::iterator c;  
  
public:  
 Patient();  
 Patient(const string &usrname, const string &pass, const string &name, string patientType,  
 string symptoms);  
 ~Patient();  
 const string &getPatientType() const;  
 void setPatientType(const string &patientType);  
 const string &getSymptoms() const;  
 void setSymptoms(const string &symptoms);  
 friend ostream &operator<<(ostream &os, const Patient &patient);  
 void viewTestHistory() override;  
 void addPatientType(string patientType);  
 void showAllPatientType();  
 string \*getSelectedPatientType();  
 void addCovidTest(CovidTest &covidTest);  
 CovidTest \*getCovidTest();  
 vector<CovidTest> getCovidTests() const;  
};  
  
class CTIS {  
private:  
 vector<CentreOfficer> CTISCentreManagers;  
 vector<CentreOfficer> CTISTesters;  
 vector<TestCentre> CTISTestCentres;  
 vector<Patient> CTISPatients;  
  
public:  
 CTIS();  
 ~CTIS();  
 void addCentreManager(CentreOfficer &centreManager);  
 void addTestCentre(TestCentre &testCentre);  
 void addTester(CentreOfficer &tester);  
 void addPatient(Patient &patient);  
 bool findCentreManager(const string& username);  
 bool findTester(const string& username);  
 bool findPatient(const string& username);  
 TestCentre \*getTestCentreByID(const string& centreID);  
 TestCentre \*getTestCentre();  
 Patient \*getPatient(const string& username);  
 TestKit \*getTestKit(const string& kitID);  
 CentreOfficer \*getCentreManager(const string& centreName);  
 static TestKit \*getTestKitByManager(CentreOfficer &centreOfficer, const string& kitID);  
 CovidTest \*getCovidTestOfTester(const string& testID);  
 Patient \*getPatientByCovidTest(const string& testID);  
 TestCentre \*getTestCentreByCovidTest(const string& testID);  
 TestKit \*getTestKitByCovidTest(const string& testID);  
 CovidTest \*getPatientCovidTest(const string& testID);  
 CovidTest \*getCovidTet(const string& testID);  
 CentreOfficer \*adminLoginValidation(const string& usrname, const string& pass);  
 Patient \*patientLoginValidation(const string& username, const string& password);  
 CentreOfficer \*testerLoginValidation(const string& username, const string& password);  
 int getNumOfApprovedTestCentres();  
 void showApprovedTestCentres();  
 void showAllTestCentres();  
 void showAllCovidTestID();  
 bool getNumOfPendingCovidTest();  
 bool covidTestIDValidation(const string& testID);  
};  
#endif *//E1900344\_CTIS\_H*

## User.cpp

*/\*\*  
 \* User is abstract class of Patient and CentreOfficer,  
 \* defines a simple object type that represents a User  
 \* Student ID: E1900344  
 \* Date: 16 December 2021  
 \* C++ version: C++14  
 \* IDE : CLion  
 \*/*#include <utility>  
#include "CTIS.h"  
  
*/\*\*  
 \* Default constructor without arguments,  
 \* to create User objects and initialize data member to default values.  
 \*/*User::User() : username("unknown"), password("unknown"), name("unknown"){}  
  
*/\*\*  
 \* Constructor with 3 arguments,  
 \* to set the data member  
 \* to the value passed in the arguments.  
 \* @param usrname value passed in the argument.  
 \* @param passw value passed in the argument.  
 \* @param name value passed in the argument.  
 \*/*User::User(string usrname, string passw, string name)  
: username(std::move(usrname)), password(std::move(passw)), name(std::move(name)) {}  
  
*/\*\*  
 \* An user defined destructor  
 \*/*User::~User() {}  
  
*/\*\*  
 \* The getUsername getter method is used to  
 \* obtain the username value  
 \* @return a string value of username  
 \*/*const string &User::getUsername() const {  
 return username;  
}  
  
*/\*\*  
 \* The setUsername setter method is used to  
 \* set the value of username data member.  
 \* @param newUsername value to set the username data member.  
 \*/*void User::setUsername(const string &newUsername) {  
 User::username = newUsername;  
}  
  
*/\*\*  
 \* The getPassword getter method is used to  
 \* obtain the password value  
 \* @return a string value of password  
 \*/*const string &User::getPassword() const {  
 return password;  
}  
  
*/\*\*  
 \* The setPassword setter method is used to  
 \* set the value of password data member.  
 \* @param newPassword value to set the password data member.  
 \*/*void User::setPassword(const string &newPassword) {  
 User::password = newPassword;  
}  
  
*/\*\*  
 \* The getName getter method is used to  
 \* obtain the name value  
 \* @return a string value of name  
 \*/*const string &User::getName() const {  
 return name;  
}  
  
*/\*\*  
 \* The setName setter method is used to  
 \* set the value of name data member.  
 \* @param newName value to set the name data member.  
 \*/*void User::setName(const string &newName) {  
 User::name = newName;  
}  
  
*/\*\*  
 \* The output stream method is used to return  
 \* the detail User object information.  
 \* @param os the original ostream object  
 \* @param user access the private data member within a User object.  
 \* @return detail User object information.  
 \*/*ostream &operator<<(ostream &os, const User &user) {  
 os << "Username: " << user.username  
 << "\nPassword: " << user.password  
 << "\nName: " << user.name ;  
 return os;  
}

## CovidTest.cpp

*/\*\*  
 \* CovidTest class defines a simple object type that represents a CovidTest.  
 \* Student ID: E1900344  
 \* Date: 16 December 2021  
 \* C++ version: C++14  
 \* IDE : CLion  
 \*/*#include <utility>  
#include "CTIS.h"  
  
  
*/\*\*  
 \* Default constructor without arguments,  
 \* to create Covid Test objects and initialize data member to default values.  
 \*/*CovidTest::CovidTest() : testID("unknown"), testDate("unknown"), result("unknown"), resultDate("unknown"),  
status("unknown") {}  
  
 */\*\*  
 \* Constructor with 5 arguments,  
 \* to set the data member  
 \* to the value passed in the arguments  
 \* @param testId value passed in the argument.  
 \* @param testDate value passed in the argument.  
 \* @param result value passed in the argument.  
 \* @param resultDate value passed in the argument.  
 \* @param status value passed in the argument.  
 \*/*CovidTest::CovidTest(string testId, string testDate, string result, string resultDate,  
 string status) : testID(std::move(testId)), testDate(std::move(testDate)),  
 result(std::move(result)), resultDate(std::move(resultDate)),  
 status(std::move(status)) {}  
  
*/\*\*  
 \* An user defined destructor  
 \*/*CovidTest::~CovidTest() {}  
  
*/\*\*  
 \* The getTestId getter method is used to  
 \* obtain the testID value  
 \* @return a string value of testID  
 \*/*const string &CovidTest::getTestId() const {  
 return testID;  
}  
  
*/\*\*  
 \* The setTestId setter method is used to  
 \* set the value of testID data member.  
 \* @param testID value to set the testID data member.  
 \*/*void CovidTest::setTestId(const string &testId) {  
 testID = testId;  
}  
  
*/\*\*  
 \* The getTestDate getter method is used to  
 \* obtain the testDate value  
 \* @return a string value of testDate  
 \*/*const string &CovidTest::getTestDate() const {  
 return testDate;  
}  
  
*/\*\*  
 \* The setTestDate setter method is used to  
 \* set the value of testDate data member.  
 \* @param testDate value to set the testDate data member.  
 \*/*void CovidTest::setTestDate(const string &testDate) {  
 CovidTest::testDate = testDate;  
}  
  
*/\*\*  
 \* The getResult getter method is used to  
 \* obtain the result value  
 \* @return a string value of result  
 \*/*const string &CovidTest::getResult() const {  
 return result;  
}  
  
*/\*\*  
 \* The setResult setter method is used to  
 \* set the value of result data member.  
 \* @param result value to set the result data member.  
 \*/*void CovidTest::setResult(const string &result) {  
 CovidTest::result = result;  
}  
  
*/\*\*  
 \* The getResultDate getter method is used to  
 \* obtain the resultDate value  
 \* @return a string value of resultDate  
 \*/*const string &CovidTest::getResultDate() const {  
 return resultDate;  
}  
  
*/\*\*  
 \* The setResultDate setter method is used to  
 \* set the value of resultDate data member.  
 \* @param resultDate value to set the resultDate data member.  
 \*/*void CovidTest::setResultDate(const string &resultDate) {  
 CovidTest::resultDate = resultDate;  
}  
  
*/\*\*  
 \* The getStatus getter method is used to  
 \* obtain the status value  
 \* @return a string value of status  
 \*/*const string &CovidTest::getStatus() const {  
 return status;  
}  
  
*/\*\*  
 \* The setStatus setter method is used to  
 \* set the value of status data member.  
 \* @param status value to set the status data member.  
 \*/*void CovidTest::setStatus(const string &status) {  
 CovidTest::status = status;  
}  
  
*/\*\*  
 \* The output stream method is used to return  
 \* the detail CovidTest object information.  
 \* @param os the original ostream object  
 \* @param test access the private data member within a CovidTest object.  
 \* @return detail CovidTest object information.  
 \*/*ostream &operator<<(ostream &os, const CovidTest &test) {  
 os << "\nTest ID: " << test.testID  
 << "\nTest date: " << test.testDate  
 << "\nResult: " << test.result  
 << "\nResult date: " << test.resultDate  
 << "\nStatus: " << test.status;  
 return os;  
}

## TestKit.cpp

*/\*\*  
 \* TestKit class defines a simple object type that represents a TestKit.  
 \* Student ID: E1900344  
 \* Date: 16 December 2021  
 \* C++ version: C++14  
 \* IDE : CLion  
 \*/*#include "CTIS.h"  
#include <iostream>  
#include <utility>  
  
using namespace std;  
  
*/\*\*  
 \* Default constructor without arguments,  
 \* to create TestKit objects and initialize data member to default values.  
 \*/*TestKit::TestKit() : testName("unknown"), kitID("unknown"), availableStock(0) {}  
  
 */\*\*  
 \* Constructor with 3 arguments,  
 \* to set the data member  
 \* to the value passed in the arguments  
 \* @param testName value passed in the argument.  
 \* @param kitId value passed in the argument.  
 \* @param availableStock value passed in the argument.  
 \*/*TestKit::TestKit(string testName, string kitId, int availableStock)  
: testName(std::move(testName)), kitID(std::move(kitId)), availableStock(availableStock) {}  
  
*/\*\*  
 \* An user defined destructor  
 \*/*TestKit::~TestKit() {}  
  
*/\*\*  
 \* The getTestName getter method is used to  
 \* obtain the testName value  
 \* @return a string value of testName  
 \*/*const string &TestKit::getTestName() const {  
 return testName;  
}  
  
*/\*\*  
 \* The setTestName setter method is used to  
 \* set the value of testName data member.  
 \* @param testName value to set the testName data member.  
 \*/*void TestKit::setTestName(const string &testName) {  
 TestKit::testName = testName;  
}  
  
*/\*\*  
 \* The getKitId getter method is used to  
 \* obtain the kitID value  
 \* @return a string value of kitID  
 \*/*const string &TestKit::getKitId() const {  
 return kitID;  
}  
  
*/\*\*  
 \* The setTestName setter method is used to  
 \* set the value of kitId data member.  
 \* @param kitId value to set the kitId data member.  
 \*/*void TestKit::setKitId(const string &kitId) {  
 kitID = kitId;  
}  
  
*/\*\*  
\* The getAvailableStock getter method is used to  
\* obtain the availableStock value  
\* @return an int value of availableStock  
\*/*int TestKit::getAvailableStock() const {  
 return availableStock;  
}  
  
*/\*\*  
 \* The setAvailableStock setter method is used to  
 \* set the value of availableStock data member.  
 \* @param availableStock value to set the availableStock data member.  
 \*/*void TestKit::setAvailableStock(int availableStock) {  
 TestKit::availableStock = availableStock;  
}  
  
*/\*\*  
 \* The output stream method is used to return  
 \* the detail TestKit object information.  
 \* @param os the original ostream object  
 \* @param kit access the private data member within a TestKit object.  
 \* @return detail TestKit object information.  
 \*/*ostream &operator<<(ostream &os, const TestKit &kit) {  
 os << "\nTest name: " << kit.testName  
 << "\nKit ID: " << kit.kitID  
 << "\nAvailable stock: " << kit.availableStock;  
 return os;  
}  
  
*/\*\*  
 \* The addCovidTest used to append value to container.  
 \* @param covidTest reference covid test object.  
 \*/*void TestKit::addCovidTest(CovidTest &covidTest) {  
 covidTests.push\_back(covidTest);  
}  
  
*/\*\*  
 \* The getCovidTest support method which is used to  
 \* get covidTests object address in container.  
 \* @return covid test object address or nullptr if empty.  
 \*/*CovidTest \*TestKit::getCovidTest() {  
 for(auto & covidTest : covidTests) {  
 return &covidTest;  
 }  
 return nullptr;  
}  
  
*/\*\*  
 \* The getCovidTests support method which is used to  
 \* get size of covidTests container.  
 \* @return address of covidTests container.  
 \*/*vector<CovidTest> TestKit::getCovidTests() const {  
 return covidTests;  
}

## TestCentre.cpp

*/\*\*  
 \* TestCentre class defines a simple object type that represents a TestCentre.  
 \* Student ID: E1900344  
 \* Date: 16 December 2021  
 \* C++ version: C++14  
 \* IDE : CLion  
 \*/*#include "CTIS.h"  
  
#include <iostream>  
#include <utility>  
using namespace std;  
  
*/\*\*  
 \* Default constructor without arguments,  
 \* to create TestCentre objects and initialize data member to default values.  
 \*/*TestCentre::TestCentre() : centreId("unknown"), centreName("unknown") {  
 *// Append new stock of Test Kit by hard-coded* TestKit testKit1("SGTi-flex COVID-19 IgG", "newStock", 0);  
 TestKit testKit2("T-Detect COVID Test", "newStock", 0);  
 TestKit testKit3("SARS-CoV-2 RBD IgG test", "newStock", 0);  
 addNewStock(testKit1);  
 addNewStock(testKit2);  
 addNewStock(testKit3);  
}  
  
*/\*\*  
 \* Constructor with 2 arguments,  
 \* to set the data member  
 \* to the value passed in the arguments  
 \* @param centreId value passed in the argument.  
 \* @param centreName value passed in the argument.  
 \*/*TestCentre::TestCentre(string centreId, string centreName) : centreId(std::move(centreId)), centreName(std::move(centreName)) {}  
  
*/\*\*  
 \* An user defined destructor  
 \*/*TestCentre::~TestCentre() {}  
  
*/\*\*  
 \* The getCentreId getter method is used to  
 \* obtain the centreId value  
 \* @return a string value of centreId  
 \*/*const string &TestCentre::getCentreId() const {  
 return centreId;  
}  
  
*/\*\*  
 \* The setCentreId setter method is used to  
 \* set the value of centreId data member.  
 \* @param centreId value to set the centreId data member.  
 \*/*void TestCentre::setCentreId(const string &centreId) {  
 TestCentre::centreId = centreId;  
}  
  
*/\*\*  
 \* The getCentreName getter method is used to  
 \* obtain the centreName value  
 \* @return a string value of centreName  
 \*/*const string &TestCentre::getCentreName() const {  
 return centreName;  
}  
  
*/\*\*  
 \* The setCentreName setter method is used to  
 \* set the value of centreName data member.  
 \* @param centreName value to set the centreName data member.  
 \*/*void TestCentre::setCentreName(const string &centreName) {  
 TestCentre::centreName = centreName;  
}  
  
*/\*\*  
 \* The output stream method is used to return  
 \* the detail TestCentre object information.  
 \* @param os the original ostream object  
 \* @param centre access the private data member within a TestCentre object.  
 \* @return detail TestCentre object information.  
 \*/*ostream &operator<<(ostream &os, const TestCentre &centre) {  
 os << "\nCentre ID: " << centre.centreId  
 << "\nCentre name: " << centre.centreName;  
 return os;  
}  
  
*/\*\*  
 \* The addNewStock used to append value to container.  
 \* @param testKit reference testKit object.  
 \*/*void TestCentre::addNewStock(TestKit &testKit) {  
 newStocks.push\_back(testKit);  
}  
  
*/\*\*  
 \* The showNewStocks support method which is used to  
 \* show new stock of Test Kit in container.  
 \*/*void TestCentre::showNewStocks() {  
 for(int i = 0; i < newStocks.size(); i++) {  
 if(newStocks[i].getKitId() == "newStock")  
 cout << i+1 << ". " << newStocks[i].getTestName() << endl;  
 }  
}  
  
*/\*\*  
 \* The getNumOfNewStocks support method which is used to  
 \* get num of new stocks in container.  
 \* @return int num of stock.  
 \*/*int TestCentre::getNumOfNewStocks() {  
 int newStock = 0;  
 for(auto & i : newStocks) {  
 if(i.getKitId() == "newStock")  
 newStock++;  
 }  
 return newStock;  
}  
  
*/\*\*  
 \* The getNewStock support method which is used to  
 \* get newStocks object address in container.  
 \* @return newStocks object address or nullptr if empty.  
 \*/*TestKit \*TestCentre::getNewStock() {  
 for(auto & newStock : newStocks) {  
 return &newStock;  
 }  
 return nullptr;  
}  
  
*/\*\*  
 \* The addTestKit used to append value to container.  
 \* @param testKit reference testKit object.  
 \*/*void TestCentre::addTestKit(TestKit &testKit) {  
 testKits.push\_back(testKit);  
}  
  
*/\*\*  
 \* The getTestKits used to get size of testKits container.  
 \* @return address of testKits container.  
 \*/*vector<TestKit> TestCentre::getTestKits() const {  
 return testKits;  
}  
  
*/\*\*  
 \* The showAllTestKit support method which is used to  
 \* show all Test Kit in the container.  
 \*/*void TestCentre::showAllTestKit() {  
 for (t = testKits.begin(); t < testKits.end(); t++)  
 cout << \*t << endl;  
}  
  
*/\*\*  
 \* The getTestKit support method which is used to  
 \* get testKits object address in container.  
 \* @return testKits object address or nullptr if empty.  
 \*/*TestKit \*TestCentre::getTestKit() {  
 for(auto & testKit : testKits) {  
 return &testKit;  
 }  
 return nullptr;  
}  
  
*/\*\*  
 \* The getTestKit support method which is used to  
 \* find teskits object address in container based on kitID.  
 \* @param kitID reference to compare.  
 \* @return testKits object address or nullptr if empty.  
 \*/*TestKit \*TestCentre::findTestKit(const string& kitID) {  
 for(auto & testKit : testKits) {  
 if(testKit.getKitId() == kitID)  
 return &testKit;  
 }  
 return nullptr;  
}  
  
*/\*\*  
 \* The getTestKit support method which is used to  
 \* compare teskits object address in container based on kitID.  
 \* @param kitID reference to compare.  
 \* @return true if found or false not found.  
 \*/*bool TestCentre::kitIDValidation(const string& kitID) {  
 for(auto & testKit : testKits) {  
 if(testKit.getKitId() == kitID)  
 return true;  
 }  
 return false;  
}  
  
*/\*\*  
 \* The showAvailableTestKit support method which is used to  
 \* show all available Test Kit in container.  
 \*/*void TestCentre::showAvailableTestKit() {  
 for(auto & testKit : testKits) {  
 if(testKit.getAvailableStock() > 0)  
 cout << testKit << endl;  
 }  
}

## CentreOfficer.cpp

*/\*\*  
 \* CentreOfficer concrete subclass of User  
 \* defines a simple object type that represents a CentreOfficer.  
 \* Student ID: E1900344  
 \* Date: 16 December 2021  
 \* C++ version: C++14  
 \* IDE : CLion  
 \*/*#include <iostream>  
#include <utility>  
#include "CTIS.h"  
  
*/\*\*  
 \* Default constructor without arguments,  
 \* to create CentreOfficer objects and initialize data member to default values.  
 \*/*CentreOfficer::CentreOfficer() : position("unknown") {}  
  
*/\*\*  
 \* Constructor with 4 arguments,  
 \* to set the data member  
 \* to the value passed in the arguments  
 \* @param usrname value passed in the argument.  
 \* @param pass value passed in the argument.  
 \* @param name value passed in the argument.  
 \* @param position value passed in the argument.  
 \*/*CentreOfficer::CentreOfficer(const string &usrname, const string &pass, const string &name, string position)  
 : User(usrname, pass, name), position(std::move(position)) {}  
  
*/\*\*  
\* An user defined destructor  
\*/*CentreOfficer::~CentreOfficer() {}  
  
*/\*\*  
 \* The getPosition getter method is used to  
 \* obtain the position value  
 \* @return a string value of position  
 \*/*const string &CentreOfficer::getPosition() const {  
 return position;  
}  
  
*/\*\*  
 \* The setPosition setter method is used to  
 \* set the value of position data member.  
 \* @param position value to set the position data member.  
 \*/*void CentreOfficer::setPosition(const string &position) {  
 CentreOfficer::position = position;  
}  
  
*/\*\*  
 \* The output stream method is used to return  
 \* the detail CentreOfficer object information.  
 \* @param os the original ostream object  
 \* @param officer access the private data member within a CentreOfficer object.  
 \* @return detail CentreOfficer object information.  
 \*/*ostream &operator<<(ostream &os, const CentreOfficer &officer) {  
 os << static\_cast<const User &>(officer)  
 << "\nPosition: " << officer.position;  
 return os;  
}  
  
*/\*\*  
 \* The viewTestHistory pure virtual method which is used to  
 \* show test information for Centre Manager and Tester.  
 \*/*void CentreOfficer::viewTestHistory() {  
 if(position == "Centre manager") {  
 for (t = testCentres.begin(); t < testCentres.end(); t++) {  
 for(int j = 0; j < t->getTestKits().size(); j++) {  
 for(int k = 0; k < t->getTestKit()[j].getCovidTests().size(); k++) {  
 cout << t->getTestKit()[j].getCovidTest()[k] << endl;  
 }  
 }  
 }  
 }  
 else if(position == "Tester") {  
 for (c = covidTests.begin(); c < covidTests.end(); c++)  
 cout << \*c << endl;  
 }  
}  
  
*/\*\*  
 \* The addTestCentre used to append value to container.  
 \* @param testCentre reference testCentre object.  
 \*/*void CentreOfficer::addTestCentre(TestCentre testCentre) {  
 testCentres.push\_back(testCentre);  
}  
  
*/\*\*  
 \* The getTestCentres used to get size of testCentres container.  
 \* @return address of testCentres container.  
 \*/*vector<TestCentre> CentreOfficer::getTestCentres() const {  
 return testCentres;  
}  
  
*/\*\*  
 \* The getTestCentre support method which is used to  
 \* get testCentres object address in container.  
 \* @return testCentres object address or nullptr if empty.  
 \*/*TestCentre \*CentreOfficer::getTestCentre() {  
 for(auto & testCentre : testCentres) {  
 return &testCentre;  
 }  
 return nullptr;  
}  
  
*/\*\*  
 \* The addCovidTest used to append value to container.  
 \* @param covidTest reference covidTest object.  
 \*/*void CentreOfficer::addCovidTest(CovidTest &covidTest) {  
 covidTests.push\_back(covidTest);  
}  
  
*/\*\*  
 \* The getCovidTests used to get size of covidTests container.  
 \* @return address of covidTests container.  
 \*/*vector<CovidTest> CentreOfficer::getCovidTests() const {  
 return covidTests;  
}  
  
*/\*\*  
 \* The getCovidTest support method which is used to  
 \* get covidTests object address in container.  
 \* @return covidTests object address or nullptr if empty.  
 \*/*CovidTest \*CentreOfficer::getCovidTest() {  
 for(auto & covidTest : covidTests) {  
 return &covidTest;  
 }  
 return nullptr;  
}

## Patient.cpp

*/\*\*  
 \* Patient concrete subclass of User  
 \* defines a simple object type that represents a Patient.  
 \* Student ID: E1900344  
 \* Date: 16 December 2021  
 \* C++ version: C++14  
 \* IDE : CLion  
 \*/*#include "CTIS.h"  
#include <iostream>  
#include <utility>  
  
using namespace std;  
  
*/\*\*  
 \* Default constructor without arguments,  
 \* to create Patient objects and initialize data member to default values.  
 \*/*Patient::Patient() : patientType("unknown"), symptoms("unknown") {  
 *// Create patient type by hard-coded* addPatientType("returnee");  
 addPatientType("quarantined");  
 addPatientType("close contact");  
 addPatientType("infected");  
 addPatientType("suspected");  
}  
  
 */\*\*  
 \* Constructor with 4 arguments,  
 \* to set the data member  
 \* to the value passed in the arguments  
 \* @param usrname value passed in the argument.  
 \* @param pass value passed in the argument.  
 \* @param name value passed in the argument.  
 \* @param patientType value passed in the argument.  
 \* @param symptoms value passed in the argument.  
 \*/*Patient::Patient(const string &usrname, const string &pass, const string &name, string patientType,  
 string symptoms) : User(usrname, pass, name),  
 patientType(std::move(patientType)),symptoms(std::move(symptoms)) {}  
  
*/\*\*  
\* An user defined destructor  
\*/*Patient::~Patient() {}  
  
*/\*\*  
 \* The getPatientType getter method is used to  
 \* obtain the patientType value  
 \* @return a string value of patientType  
 \*/*const string &Patient::getPatientType() const {  
 return patientType;  
}  
  
*/\*\*  
 \* The setPatientType setter method is used to  
 \* set the value of patientType data member.  
 \* @param patientType value to set the patientType data member.  
 \*/*void Patient::setPatientType(const string &patientType) {  
 Patient::patientType = patientType;  
}  
  
*/\*\*  
 \* The getSymptoms getter method is used to  
 \* obtain the symptoms value  
 \* @return a string value of symptoms  
 \*/*const string &Patient::getSymptoms() const {  
 return symptoms;  
}  
  
*/\*\*  
 \* The setSymptoms setter method is used to  
 \* set the value of symptoms data member.  
 \* @param symptoms value to set the symptoms data member.  
 \*/*void Patient::setSymptoms(const string &symptoms) {  
 Patient::symptoms = symptoms;  
}  
  
*/\*\*  
 \* The output stream method is used to return  
 \* the detail Patient object information.  
 \* @param os the original ostream object  
 \* @param patient access the private data member within a Patient object.  
 \* @return detail Patient object information.  
 \*/*ostream &operator<<(ostream &os, const Patient &patient) {  
 os << static\_cast<const User &>(patient)  
 << "\nPatient type: " << patient.patientType  
 << "\nSymptoms: " << patient.symptoms;  
 return os;  
}  
  
*/\*\*  
 \* The viewTestHistory pure virtual method which is used to  
 \* show test information for Patient.  
 \*/*void Patient::viewTestHistory() {  
 for (c = covidTests.begin(); c < covidTests.end(); c++)  
 cout << \*c << endl;  
}  
  
*/\*\*  
 \* The addPatientType used to append value to container.  
 \* @param patientType reference patientType object.  
 \*/*void Patient::addPatientType(string patientType) {  
 patientTypes.push\_back(patientType);  
}  
  
*/\*\*  
 \* The showAllPatientType support method which is used to  
 \* show all patient type in the container.  
 \*/*void Patient::showAllPatientType() {  
 for(int i = 0; i < patientTypes.size(); i++)  
 cout << i+1 << ". " << patientTypes[i] << endl;  
}  
  
*/\*\*  
 \* The getSelectedPatientType support method which is used to  
 \* get patient type address in the container.  
 \*/*string \*Patient::getSelectedPatientType() {  
 for(auto & i : patientTypes) {  
 return &i;  
 }  
 return nullptr;  
}  
  
*/\*\*  
 \* The addCovidTest used to append value to container.  
 \* @param covidTest reference covidTest object.  
 \*/*void Patient::addCovidTest(CovidTest &covidTest) {  
 covidTests.push\_back(covidTest);  
}  
  
*/\*\*  
 \* The getCovidTest support method which is used to  
 \* get covidTest object address in container.  
 \* @return covidTest object address or nullptr if empty.  
 \*/*CovidTest \*Patient::getCovidTest() {  
 for(auto & covidTest : covidTests) {  
 return &covidTest;  
 }  
 return nullptr;  
}  
  
*/\*\*  
 \* The getCovidTests used to get size of covidTests container.  
 \* @return address of covidTests container.  
 \*/*vector<CovidTest> Patient::getCovidTests() const {  
 return covidTests;  
}

## CTIS.cpp

#include "CTIS.h"  
#include <iostream>  
  
*/\*\*  
 \* A default constructor which is used to  
 \* initialise default value of CTIS System.  
 \*/*CTIS::CTIS() {  
 *// Add approved Test Centres by hard-coded* TestCentre testCentre1("approved","Tapah Health Clinic");  
 TestCentre testCentre2("approved","Kampar Health Clinic");  
 addTestCentre(testCentre1);  
 addTestCentre(testCentre2);  
}  
  
*/\*\*  
\* An user defined destructor  
\*/*CTIS::~CTIS() {}  
  
*/\*\*  
 \* The addCentreManager used to append value to container.  
 \* @param centreManager reference of centreManager object.  
 \*/*void CTIS::addCentreManager(CentreOfficer &centreManager) {  
 CTISCentreManagers.push\_back(centreManager);  
}  
  
*/\*\*  
 \* The addTestCentre used to append value to container.  
 \* @param testCentre reference of testCentre object.  
 \*/*void CTIS::addTestCentre(TestCentre &testCentre) {  
 CTISTestCentres.push\_back(testCentre);  
}  
  
*/\*\*  
 \* The addTester used to append value to container.  
 \* @param tester reference of tester object.  
 \*/*void CTIS::addTester(CentreOfficer &tester) {  
 CTISTesters.push\_back(tester);  
}  
  
*/\*\*  
 \* The addPatient used to append value to container.  
 \* @param patient reference of patient object.  
 \*/*void CTIS::addPatient(Patient &patient) {  
 CTISPatients.push\_back(patient);  
}  
  
*/\*\*  
 \* The findCentreManager support method which is used to  
 \* compare Centre Manager username in container.  
 \* @param username reference to compare.  
 \* @return true if found or false not found.  
 \*/*bool CTIS::findCentreManager(const string& username) {  
 for(auto & CTISCentreOfficer : CTISCentreManagers) {  
 if(CTISCentreOfficer.getUsername() == username)  
 return true;  
 }  
 return false;  
}  
  
  
  
*/\*\*  
 \* The findTester support method used to  
 \* get the address of a Test Center object based on the username in the container.  
 \* @param username reference to compare.  
 \* @return true if found or false not found.  
 \*/*bool CTIS::findTester(const string& username) {  
 for(auto & CTISTester : CTISTesters) {  
 if(CTISTester.getUsername() == username)  
 return true;  
 }  
 return false;  
}  
  
*/\*\*  
 \* The findPatient support method used to  
 \* get the address of a Patient object based on the username in the container.  
 \* @param username reference to compare.  
 \* @return true if found or false not found.  
 \*/*bool CTIS::findPatient(const string& username) {  
 for(auto & CTISPatient : CTISPatients) {  
 if(CTISPatient.getUsername() == username)  
 return false;  
 }  
 return true;  
}  
  
*/\*\*  
 \* The getTestCentreByID support method used to  
 \* get the address of a Test Center object based on the center ID in the container.  
 \* @param centreID reference to compare.  
 \* @return address of a Test Center object or nullptr if not found.  
 \*/*TestCentre \*CTIS::getTestCentreByID(const string& centreID) {  
 for(auto & CTISTestCentre : CTISTestCentres) {  
 if(CTISTestCentre.getCentreId() == centreID)  
 return &CTISTestCentre;  
 }  
 return nullptr;  
}  
  
*/\*\*  
 \* getTestCentre support method used to  
 \* get CTISTestCentres object address in container.  
 \* @return CTISTestCentres object address or nullptr if empty.  
 \*/*TestCentre \*CTIS::getTestCentre() {  
 for(auto & CTISTestCentre : CTISTestCentres) {  
 return &CTISTestCentre;  
 }  
 return nullptr;  
}  
  
*/\*\*  
 \* The getPatient support method used to  
 \* get the address of a Patient object based on the username in the container.  
 \* @param username reference to compare.  
 \* @return address of a Patient object or nullptr if not found.  
 \*/*Patient \*CTIS::getPatient(const string& username) {  
 for(auto & CTISPatient : CTISPatients) {  
 if(CTISPatient.getUsername() == username)  
 return &CTISPatient;  
 }  
 return nullptr;  
}  
  
*/\*\*  
 \* The getTestKit support method used to  
 \* get the address of a TestKit object based on the kitID in the container.  
 \* @param kitID reference to compare.  
 \* @return address of a TestKit object or nullptr if not found.  
 \*/*TestKit \*CTIS::getTestKit(const string& kitID) {  
 for(auto & CTISTestCentre : CTISTestCentres) {  
 for(int j = 0; j< CTISTestCentre.getTestKits().size(); j++) {  
 if(CTISTestCentre.getTestKit()[j].getKitId() == kitID) {  
 return &CTISTestCentre.getTestKit()[j];  
 }  
 }  
 }  
 return nullptr;  
}  
  
*/\*\*  
 \* The getCentreManager support method used to  
 \* get the address of a CentreOfficer object based on the centreName in the container.  
 \* @param centreName reference to compare.  
 \* @return address of a CentreOfficer object or nullptr if not found.  
 \*/*CentreOfficer \*CTIS::getCentreManager(const string& centreName) {  
 for(auto & CTISCentreOfficer : CTISCentreManagers) {  
 for(int j = 0; j < CTISCentreOfficer.getTestCentres().size(); j++) {  
 if(CTISCentreOfficer.getTestCentre()[j].getCentreName() == centreName) {  
 return &CTISCentreOfficer;  
 }  
 }  
 }  
 return nullptr;  
}  
  
*/\*\*  
 \* The getTestKitByManager support method used to  
 \* get the address of a TestKit object based on  
 \* the Centre Manager object and kit ID in the container.  
 \* @param centreManager reference object to compare.  
 \* @param kitID reference to compare.  
 \* @return address of a TestKit object or nullptr if not found.  
 \*/*TestKit \*CTIS::getTestKitByManager(CentreOfficer &centreManager, const string& kitID) {  
 for(int x = 0; x < centreManager.getTestCentres().size(); x++) {  
 for(int y = 0; y < centreManager.getTestCentre()[x].getTestKits().size(); y++) {  
 if(centreManager.getTestCentre()[x].getTestKit()[y].getKitId() == kitID) {  
 return &centreManager.getTestCentre()[x].getTestKit()[y];  
 }  
 }  
 }  
 return nullptr;  
}  
  
*/\*\*  
 \* The getCovidTestOfTester support method used to  
 \* get the address of a CovidTest object based on the test ID in the container.  
 \* @param testID reference to compare.  
 \* @return address of a CovidTest object or nullptr if not found.  
 \*/*CovidTest \*CTIS::getCovidTestOfTester(const string& testID) {  
 for(auto & CTISTester : CTISTesters) {  
 for(int j = 0; j < CTISTester.getCovidTests().size(); j++) {  
 if(CTISTester.getCovidTest()[j].getTestId() == testID)  
 return &CTISTester.getCovidTest()[j];  
 }  
 }  
 return nullptr;  
}  
  
*/\*\*  
 \* The getPatientByCovidTest support method used to  
 \* get the address of a Patient object based on the test ID in the container.  
 \* @param testID reference to compare.  
 \* @return address of a Patient object or nullptr if not found.  
 \*/*Patient \*CTIS::getPatientByCovidTest(const string& testID) {  
 for(auto & CTISPatient : CTISPatients) {  
 for(int j = 0; j < CTISPatient.getCovidTests().size(); j++) {  
 if(CTISPatient.getCovidTests()[j].getTestId() == testID)  
 return &CTISPatient;  
 }  
 }  
 return nullptr;  
}  
  
*/\*\*  
 \* The getTestCentreByCovidTest support method used to  
 \* get the address of a TestCentre object based on the test ID in the container.  
 \* @param testID reference to compare.  
 \* @return address of a TestCentre object or nullptr if not found.  
 \*/*TestCentre \*CTIS::getTestCentreByCovidTest(const string& testID) {  
 for(auto & CTISCentreOfficer : CTISCentreManagers) {  
 for(int j = 0; j < CTISCentreOfficer.getTestCentres().size(); j++) {  
 for(int k =0; k< CTISCentreOfficer.getTestCentre()[j].getTestKits().size(); k++ ) {  
 for(int l = 0; l < CTISCentreOfficer.getTestCentre()[j].getTestKit()[k].getCovidTests().size(); l++) {  
 if(CTISCentreOfficer.getTestCentre()[j].getTestKit()[k].getCovidTest()[l].getTestId() == testID) {  
 return &CTISCentreOfficer.getTestCentre()[j];  
 }  
 }  
 }  
 }  
 }  
 return nullptr;  
}  
  
*/\*\*  
 \* The getTestKitByCovidTest support method used to  
 \* get the address of a TestKit object based on the test ID in the container.  
 \* @param testID reference to compare.  
 \* @return address of a TestKit object or nullptr if not found.  
 \*/*TestKit \*CTIS::getTestKitByCovidTest(const string& testID) {  
 for(auto & CTISCentreOfficer : CTISCentreManagers) {  
 for(int j = 0; j < CTISCentreOfficer.getTestCentres().size(); j++) {  
 for(int k =0; k< CTISCentreOfficer.getTestCentre()[j].getTestKits().size(); k++ ) {  
 for(int l = 0; l < CTISCentreOfficer.getTestCentre()[j].getTestKit()[k].getCovidTests().size(); l++) {  
 if(CTISCentreOfficer.getTestCentre()[j].getTestKit()[k].getCovidTest()[l].getTestId() == testID) {  
 return &CTISCentreOfficer.getTestCentre()[j].getTestKit()[k];  
 }  
 }  
 }  
 }  
 }  
 return nullptr;  
}  
  
*/\*\*  
 \* The getPatientCovidTest support method used to  
 \* get the address of a CovidTest object based on the test ID in the container.  
 \* @param testID reference to compare.  
 \* @return address of a CovidTest object or nullptr if not found.  
 \*/*CovidTest \*CTIS::getPatientCovidTest(const string& testID) {  
 for(auto & CTISPatient : CTISPatients) {  
 for(int j = 0; j < CTISPatient.getCovidTests().size(); j++) {  
 if(CTISPatient.getCovidTests()[j].getTestId() == testID)  
 return &CTISPatient.getCovidTest()[j];  
 }  
 }  
 return nullptr;  
}  
  
*/\*\*  
 \* The getCovidTet support method used to  
 \* get the address of a CovidTest object based on the test ID in the container.  
 \* @param testID reference to compare.  
 \* @return address of a CovidTest object or nullptr if not found.  
 \*/*CovidTest \*CTIS::getCovidTet(const string& testID) {  
 for(auto & CTISCentreOfficer : CTISCentreManagers) {  
 for(int j = 0; j < CTISCentreOfficer.getTestCentres().size(); j++) {  
 for(int k =0; k< CTISCentreOfficer.getTestCentre()[j].getTestKits().size(); k++ ) {  
 for(int l = 0; l < CTISCentreOfficer.getTestCentre()[j].getTestKit()[k].getCovidTests().size(); l++) {  
 if(CTISCentreOfficer.getTestCentre()[j].getTestKit()[k].getCovidTest()[l].getTestId() == testID) {  
 return &CTISCentreOfficer.getTestCentre()[j].getTestKit()[k].getCovidTest()[l];  
 }  
 }  
 }  
 }  
 }  
 return nullptr;  
}  
  
*/\*\*  
 \* The adminLoginValidation support method used to  
 \* compare usernames and passwords of Centre Manager objects in containers.  
 \* @param usrname username to compare  
 \* @param pass password to compare  
 \* @return CTISCentreOfficer object address or nullptr if empty.  
 \*/*CentreOfficer \*CTIS::adminLoginValidation(const string& usrname, const string& pass) {  
 for(auto & CTISCentreOfficer : CTISCentreManagers) {  
 if (CTISCentreOfficer.getUsername() == usrname && CTISCentreOfficer.getPassword() == pass)  
 return &CTISCentreOfficer;  
 }  
 return nullptr;  
}  
  
*/\*\*  
 \* The patientLoginValidation support method used to  
 \* compare usernames and passwords of Patient objects in containers.  
 \* @param username username to compare  
 \* @param password password to compare  
 \* @return CTISPatient object address or nullptr if empty.  
 \*/*Patient \*CTIS::patientLoginValidation(const string& username, const string& password) {  
 for (auto & CTISPatient : CTISPatients) {  
 if(CTISPatient.getUsername() == username && CTISPatient.getPassword() == password) {  
 return &CTISPatient;  
 }  
 }  
 return nullptr;  
}  
  
*/\*\*  
 \* The testerLoginValidation support method used to  
 \* compare usernames and passwords of Tester objects in containers.  
 \* @param username username to compare  
 \* @param password password to compare  
 \* @return CTISTester object address or nullptr if empty.  
 \*/*CentreOfficer \*CTIS::testerLoginValidation(const string& username, const string& password) {  
 for(auto & CTISTester : CTISTesters) {  
 if (CTISTester.getUsername() == username && CTISTester.getPassword() == password)  
 return &CTISTester;  
 }  
 return nullptr;  
}  
  
*/\*\*  
 \* getNumOfApprovedTestCentres support method used to  
 \* calculate num of approved Test Centre object.  
 \* @return num of approved Test Centre object.  
 \*/*int CTIS::getNumOfApprovedTestCentres() {  
 int approved = 0;  
 for(auto & CTISTestCentre : CTISTestCentres) {  
 if(CTISTestCentre.getCentreId() == "approved")  
 approved++;  
 }  
 return approved;  
}  
  
*/\*\*  
 \* showApprovedTestCentres support method used to  
 \* display approved Test Center object center name.  
 \*/*void CTIS::showApprovedTestCentres() {  
 for(int idx = 0; idx < CTISTestCentres.size(); idx++) {  
 if(CTISTestCentres[idx].getCentreId() == "approved")  
 cout << idx+1 << ". " << CTISTestCentres[idx].getCentreName() << endl;  
 }  
}  
  
*/\*\*  
 \* showAllTestCentres support method used to  
 \* display all Test Center object center name.  
 \*/*void CTIS::showAllTestCentres() {  
 for(int idx = 0; idx < CTISTestCentres.size(); idx++)  
 cout << idx+1 << ". " << CTISTestCentres[idx].getCentreName() << endl;  
}  
  
*/\*\*  
 \* showAllCovidTestID support method used to  
 \* displays the test ID of the pending Covid Test object.  
 \*/*void CTIS::showAllCovidTestID() {  
 for(auto & CTISTester : CTISTesters) {  
 for (int idx = 0; idx < CTISTester.getCovidTests().size(); idx++) {  
 if(CTISTester.getCovidTest()[idx].getStatus() == "pending")  
 cout << CTISTester.getCovidTest()[idx].getTestId() << endl;  
 }  
 }  
}  
  
*/\*\*  
 \* The getNumOfPendingCovidTest support method used to  
 \* get the num of pending Covid Test object.  
 \* @return true if found or false not found.  
 \*/*bool CTIS::getNumOfPendingCovidTest() {  
 for(auto & CTISTester : CTISTesters) {  
 for(int j = 0; j< CTISTester.getCovidTests().size(); j++) {  
 if(CTISTester.getCovidTest()[j].getStatus() == "pending")  
 return true;  
 }  
 }  
 return false;  
}  
  
*/\*\*  
 \* The covidTestIDValidation support method used to  
 \* compare test ID of Tester objects in containers.  
 \* @param testID reference to compare.  
 \* @return true if found or false not found.  
 \*/*bool CTIS::covidTestIDValidation(const string& testID) {  
 for(auto & CTISTester : CTISTesters) {  
 for(int j = 0; j < CTISTester.getCovidTests().size(); j++) {  
 if(CTISTester.getCovidTest()[j].getTestId() == testID)  
 return true;  
 }  
 }  
 return false;  
}

# Test Data CTIS

## Register Test Centre 1

### Register Test Centre Manager 1

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Scenario Name** | Register Test Centre | **Case name** | Register Test Centre Manager 1 |
| **Description** | Register a Test Centre Manager account. | **Priority** | High |
| **Pre-condition** | Username cannot be the same | **Post-condition** | The Centre Manager account is set up as Centre Officer |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Step** | **User Input** | **Desired Output Response** | **Real Output Response** | **Result** | **Comment** |
| 1 | Run the program | N/A | The program displays the Covid Testing Information System (CTIS) menu | The program displays the Covid Testing Information System (CTIS) menu | Pass | N/A |
| 2 | Select register test centre manager menu | 1 | Display the Test Centre Manager registration form | Display the Test Centre Manager registration form | Pass | N/A |
| 3 | Enter username | adminsurya | Username is valid and password is form is displayed | Username is valid and password is form is displayed | Pass | N/A |
| 4 | Enter password | admin\*&^2 | Password is valid and name form is displayed | Password is valid and name form is displayed | Pass | N/A |
| 5 | Enter name | Admin Surya | Name is valid and register success | Name is valid and register success | Pass | N/A |
| 6 | The system sets the position of the Centre Manager's account as the Centre Officer | N/A | The Centre Manager account is set up as Centre Officer | The Centre Manager account is set up as Centre Officer | Pass | N/A |

### Test Centre Manager Login 1

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Name** | Register test centre | **Case name** | Login Test Centre Manager 1 |
| **Description** | Test Centre Manager Login using Test Centre Manager account | **Priority** | High |
| **Pre-condition** | The Centre Manager Account has been created and has been set up as Centre Officer. Username and password must be the same as registered | **Post-condition** | Test Centre Manager can access test centre manager menu |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Step** | **User Input** | **Desired Output Response** | **Real Output Response** | **Result** | **Comment** |
| 1 | Select test centre manager login | 2 | The program displays the test centre manager login menu | The program displays the test centre manager login menu | Pass | N/A |
| 2 | Enter username | adminsurya | Username is valid and password is form is displayed | Username is valid and password is form is displayed | Pass | N/A |
| 3 | Enter password | admin\*&^2 | Password is valid and admin menu displayed | Password is valid and admin menu displayed | Pass | N/A |

### Register Test Centre 1

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Name** | Register Test Centre | **Case name** | Register Test Centre 1 |
| **Description** | Test Centre Manager select the approved Test Centre | **Priority** | High |
| **Pre-condition** | The Test Centre has been approved and the Centre Manager Account has been created and has been set up as Centre Officer | **Post-condition** | Test Centre Manager can register centre’s name |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Step** | **User Input** | **Desired Output Response** | **Real Output Response** | **Result** | **Comment** |
| 1 | Select register test centre | 1 | The program displays the approved test centre’s name | The program displays the approved test centre’s name | Pass | Test Centre has been created using hard coded in CTIS System with approved status |
| 2 | Select centre name | 2 | The centre ID is generated. Register Test Centre success, Test Centre created and recorded for the Centre Manager | The centre ID is generated. Register Test Centre success, Test Centre created and recorded for the Centre Manager | Pass | Test Centre status changed from approved to centre ID |
| 3 | Press (E/e) to log out | e | Back to Covid Testing Information System (CTIS) menu | Back to Covid Testing Information System (CTIS) menu | Pass | N/A |

## Register Test Centre 2

### Register Test Centre Manger 2

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Name** | Register Test Centre | **Case name** | Register Test Centre Manager 2 |
| **Description** | Register a Test Centre Manager account. | **Priority** | High |
| **Pre-condition** | Username cannot be the same | **Post-condition** | The Centre Manager account is set up as Centre Officer |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Step** | **User Input** | **Desired Output Response** | **Real Output Response** | **Result** | **Comment** |
| 1 | Select register test centre manager menu | 1 | Display the Test Centre Manager registration form | Display the Test Centre Manager registration form | Pass | N/A |
| 2 | Enter username | adminsurya | Username is valid and password is form is displayed | Invalid username and username form is displayed again | Pass | Validation: admin username cannot be the same |
| 3 | Enter new username | adminpradipta | Username is valid and password is form is displayed | Username is valid and password is form is displayed | Pass | N/A |
| 4 | Enter password | admin123 | Password is valid and name form is displayed | Password is valid and name form is displayed | Pass | N/A |
| 5 | Enter name | Admin Pradipta | Name is valid and register success | Name is valid and register success | Pass | N/A |
| 6 | The system sets the position of the Centre Manager's account as the Centre Officer | N/A | The Centre Manager account is set up as Centre Officer | The Centre Manager account is set up as Centre Officer | Pass | N/A |

### Test Centre Manager Login 2

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Name** | Register test centre | **Case name** | Login Test Centre Manager 2 |
| **Description** | Test Centre Manager Login using Test Centre Manager account | **Priority** | High |
| **Pre-condition** | The Centre Manager Account has been created and has been set up as Centre Officer. Username and password must be the same as registered | **Post-condition** | Test Centre Manager can access test centre manager menu |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Step** | **User Input** | **Desired Output Response** | **Real Output Response** | **Result** | **Comment** |
| 1 | Select test centre manager login | 2 | The program displays the test centre manager login menu | The program displays the test centre manager login menu | Pass | N/A |
| 2 | Enter username | adminpradipta | Username is valid and password is form is displayed | Username is valid and password is form is displayed | Pass | N/A |
| 3 | Enter password | admin | Password is valid and admin menu displayed | Warning message is shown “Can't find your Test Centre Manager account”.  Back to Covid Testing Information System (CTIS) menu | Pass | Validation: username and password must be the same as registered |
| 4 | Select test centre manager login | 2 | The program displays the test centre manager login menu | The program displays the test centre manager login menu | Pass | N/A |
| 5 | Enter username | admin | Username is valid and password is form is displayed | Username is valid and password is form is displayed | Pass | N/A |
| 6 | Enter password | admin123 | Password is valid and admin menu displayed | Invalid password and warning message is shown “Can’t find your Test Centre Manager account”.  Back to Covid Testing Information System (CTIS) menu | Pass | Validation: username and password must be the same as those registered in vector |
| 7 | Select test centre manager login | 2 | The program displays the test centre manager login menu | The program displays the test centre manager login menu | Pass | N/A |
| 8 | Enter username | adminpradipta | Username is valid and password is form is displayed | Username is valid and password is form is displayed | Pass | N/A |
| 9 | Enter password | admin123 | Password is valid and admin menu displayed | Password is valid and admin menu displayed | Pass | N/A |

### Register Test Centre 2

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Name** | Register Test Centre | **Case name** | Register Test Centre |
| **Description** | Test Centre Manager select the approved Test Centre | **Priority** | High |
| **Pre-condition** | The Test Centre has been approved and the Centre Manager Account has been created and has been set up as Centre Officer | **Post-condition** | Test Centre Manager can register centre’s name |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Step** | **User Input** | **Desired Output Response** | **Real Output Response** | **Result** | **Comment** |
| 1 | Select register test centre | 1 | The program displays the approved test centre’s name | The program displays the approved test centre’s name | Pass | Test Centre has been created using hard coded in CTIS System with approved status |
| 2 | Select centre name | 1 | The centre ID is generated. Register Test Centre success, Test Centre created and recorded for the Centre Manager | The centre ID is generated. Register Test Centre success, Test Centre created and recorded for the Centre Manager | Pass | Test Centre status changed from approved to centre ID |
| 3 | Select centre name | 1 | The warning message appears “Admin is only responsible for one test centre!”. Back to admin menu | The warning message appears “Admin is only responsible for one test centre!”. Back to admin menu | Pass | Validation: there is only one admin per test centre who will be responsible for managing the test centre |
| 4 | Press (E/e) to log out | E | Back to Covid Testing Information System (CTIS) menu | Back to Covid Testing Information System (CTIS) menu | Pass | N/A |

## Record Tester 1

### Test Centre Manager Login 1

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Name** | Record Tester | **Case name** | Login Test Centre Manager 1 |
| **Description** | Test Centre Manager Login using Test Centre Manager account | **Priority** | High |
| **Pre-condition** | The Centre Manager Account has been created and has been set up as Centre Officer. Username and password must be the same as registered | **Post-condition** | Test Centre Manager can access test centre manager menu |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Step** | **User Input** | **Desired Output Response** | **Real Output Response** | **Result** | **Comment** |
| 1 | Select test centre manager login | 2 | The program displays the test centre manager login menu | The program displays the test centre manager login menu | Pass | N/A |
| 2 | Enter username | adminsurya | Username is valid and password is form is displayed | Username is valid and password is form is displayed | Pass | N/A |
| 3 | Enter password | admin\*&^2 | Password is valid and admin menu displayed | Password is valid and admin menu displayed | Pass | N/A |

### Record Tester 1

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Name** | Record Tester | **Case name** | Record Tester 1 |
| **Description** | The Test Centre Manager records a new Test Centre Officer with the position of “Tester” | **Priority** | High |
| **Pre-condition** | The Centre Manager account is set up as Centre Officer. Username cannot be the same | **Post-condition** | Test Centre Manager can record testers |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Step** | **User Input** | **Desired Output Response** | **Real Output Response** | **Result** | **Comment** |
| 1 | Select register tester | 2 | Display the Tester registration form | Display the Tester registration form | Pass | N/A |
| 2 | Enter username | testersurya | Username is valid and password is form is displayed | Username is valid and password is form is displayed | Pass | N/A |
| 3 | Enter password | tester987\*\*& | Password is valid and name form is displayed | Password is valid and name form is displayed | Pass | N/A |
| 4 | Enter name | Tester Surya | Name is valid and register success | Name is valid and register success | Pass | N/A |
| 5 | The system sets the position of the Centre Officer account as the Tester | N/A | The Centre Officer account as the Tester | The Centre Officer account as the Tester | Pass | N/A |
| 4 | Press (E/e) to log out | E | Back to Covid Testing Information System (CTIS) menu | Back to Covid Testing Information System (CTIS) menu | Pass | N/A |

## Record Tester 2

### Test Centre Manager Login 2

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Name** | Record Tester | **Case name** | Login Test Centre Manager 2 |
| **Description** | Test Centre Manager Login using Test Centre Manager account | **Priority** | High |
| **Pre-condition** | The Centre Manager Account has been created and has been set up as Centre Officer. Username and password must be the same as registered | **Post-condition** | Test Centre Manager can access test centre manager menu |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Step** | **User Input** | **Desired Output Response** | **Real Output Response** | **Result** | **Comment** |
| 1 | Select test centre manager login | 2 | The program displays the test centre manager login menu | The program displays the test centre manager login menu | Pass | N/A |
| 2 | Enter username | adminpradipta | Username is valid and password is form is displayed | Username is valid and password is form is displayed | Pass | N/A |
| 3 | Enter password | admin123 | Password is valid and admin menu displayed | Password is valid and admin menu displayed | Pass | N/A |

### Record Tester 2

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Name** | Record Tester | **Case name** | Record Tester 2 |
| **Description** | The Test Centre Manager records a new Test Centre Officer with the position of “Tester” | **Priority** | High |
| **Pre-condition** | The Centre Manager account is set up as Centre Officer. Username cannot be the same | **Post-condition** | Test Centre Manager can record testers |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Step** | **User Input** | **Desired Output Response** | **Real Output Response** | **Result** | **Comment** |
| 1 | Select register tester | 2 | Display the Tester registration form | Display the Tester registration form | Pass | N/A |
| 2 | Enter username | testersurya | Username is valid and password is form is displayed | Invalid username and username form is displayed again | Pass | Validation: tester username cannot be the same |
| 3 | Enter new username | testerpradipta | Username is valid and password is form is displayed | Username is valid and password is form is displayed | Pass | N/A |
| 3 | Enter password | tester123 | Password is valid and name form is displayed | Password is valid and name form is displayed | Pass | N/A |
| 4 | Enter name | Tester Pradipta | Name is valid and register success | Name is valid and register success | Pass | N/A |
| 5 | The system sets the position of the Centre Officer account as the Tester | N/A | The Centre Officer account as the Tester | The Centre Officer account as the Tester | Pass | N/A |
| 4 | Press (E/e) to log out | E | Back to Covid Testing Information System (CTIS) menu | Back to Covid Testing Information System (CTIS) menu | Pass | N/A |

## Manage Kit Stock

### Test Centre Manager Login 1

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Name** | Record Tester | **Case name** | Login Test Centre Manager 1 |
| **Description** | Test Centre Manager Login using Test Centre Manager account | **Priority** | High |
| **Pre-condition** | The Centre Manager Account has been created and has been set up as Centre Officer. Username and password must be the same as registered | **Post-condition** | Test Centre Manager can access test centre manager menu |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Step** | **User Input** | **Desired Output Response** | **Real Output Response** | **Result** | **Comment** |
| 1 | Select test centre manager login | 2 | The program displays the test centre manager login menu | The program displays the test centre manager login menu | Pass | N/A |
| 2 | Enter username | adminsurya | Username is valid and password is form is displayed | Username is valid and password is form is displayed | Pass | N/A |
| 3 | Enter password | admin\*&^2 | Password is valid and admin menu displayed | Password is valid and admin menu displayed | Pass | N/A |

### Record New Stock 1

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Name** | Manage Kit Stock | **Case name** | Record New Stock 1 |
| **Description** | Test Centre Manager receives and records new test kit stock | **Priority** | High |
| **Pre-condition** | Test Centre Manager receives new stock of test kits | **Post-condition** | Test Centre Manager can record the stock of new test kits |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Step** | **User Input** | **Desired Output Response** | **Real Output Response** | **Result** | **Comment** |
| 1 | Select manage test kit stock | 3 | Display the manage test kit stock menu | Display the manage test kit stock menu | Pass | N/A |
| 2 | Select record new stock | 1 | Display new stock of test kits | Display new stock of test kits | Pass | N/A |
| 3 | Select test kit name | 3 | Test kit name is valid and display stock input form | Test kit name is valid and display stock input form | Pass | New stock Test Kit hard code in Test Centre with kit ID new stock |
| 4 | Enter stock | 2 | The new stock of Test Kit is recorded to the Test Centre and kit ID is generated | The new stock of Test Kit is recorded to the Test Centre and kit ID is generated | Pass | The new stock changed to ID |

### Record New Stock 2

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Name** | Manage Kit Stock | **Case name** | Record New Stock 2 |
| **Description** | Test Centre Manager receives and records new test kit stock | **Priority** | High |
| **Pre-condition** | Test Centre Manager receives new stock of test kits | **Post-condition** | Test Centre Manager can record the stock of new test kits |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Step** | **User Input** | **Desired Output Response** | **Real Output Response** | **Result** | **Comment** |
| 1 | Select manage test kit stock | 3 | Display the manage test kit stock menu | Display the manage test kit stock menu | Pass | N/A |
| 2 | Select record new stock | 1 | Display new stock of test kits | Display new stock of test kits | Pass | N/A |
| 3 | Select test kit name | 2 | Test kit name is valid and display stock input form | Test kit name is valid and display stock input form | Pass | New stock Test Kit hard code in Test Centre with kit ID new stock |
| 4 | Enter stock | 1 | The new stock of Test Kit is recorded to the Test Centre and kit ID is generated | The new stock of Test Kit is recorded to the Test Centre and kit ID is generated | Pass | The new stock changed to ID |
| 5 | Press (E/e) to log out | E | Back to Covid Testing Information System (CTIS) menu | Back to Covid Testing Information System (CTIS) menu | Pass | N/A |

### Test Centre Manager Login 2

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Name** | Record Tester | **Case name** | Login Test Centre Manager 2 |
| **Description** | Test Centre Manager Login using Test Centre Manager account | **Priority** | High |
| **Pre-condition** | The Centre Manager Account has been created and has been set up as Centre Officer. Username and password must be the same as registered | **Post-condition** | Test Centre Manager can access test centre manager menu |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Step** | **User Input** | **Desired Output Response** | **Real Output Response** | **Result** | **Comment** |
| 1 | Select test centre manager login | 2 | The program displays the test centre manager login menu | The program displays the test centre manager login menu | Pass | N/A |
| 2 | Enter username | adminpradipta | Username is valid and password is form is displayed | Username is valid and password is form is displayed | Pass | N/A |
| 3 | Enter password | admin123 | Password is valid and admin menu displayed | Password is valid and admin menu displayed | Pass | N/A |

### Record New Stock 3

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Name** | Manage Kit Stock | **Case name** | Record New Stock 3 |
| **Description** | Test Centre Manager receives and records new test kit stock | **Priority** | High |
| **Pre-condition** | Test Centre Manager receives new stock of test kits | **Post-condition** | Test Centre Manager can record the stock of new test kits |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Step** | **User Input** | **Desired Output Response** | **Real Output Response** | **Result** | **Comment** |
| 1 | Select manage test kit stock | 3 | Display the manage test kit stock menu | Display the manage test kit stock menu | Pass | N/A |
| 2 | Select record new stock | 1 | Display new stock of test kits | Display new stock of test kits | Pass | N/A |
| 3 | Select test kit name | 1 | Test kit name is valid and display stock input form | Test kit name is valid and display stock input form | Pass | New stock Test Kit hard code in Test Centre with kit ID new stock |
| 4 | Enter stock | 2 | The new stock of Test Kit is recorded to the Test Centre and kit ID is generated | The new stock of Test Kit is recorded to the Test Centre and kit ID is generated | Pass | The new stock changed to ID |
| 5 | Select manage test kit stock | 3 | Display the manage test kit stock menu | Display the manage test kit stock menu | Pass | N/A |
| 6 | Select record new stock | 1 | Warning message is shown “All new test kit stock has been received” | Warning message is shown “All new test kit stock has been received” | Pass | Validation: only three new Test Kit stocks received |

### Update Stock

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Name** | Manage Kit Stock | **Case name** | Update stock |
| **Description** | Test Centre Manager update test kit stock | **Priority** | High |
| **Pre-condition** | Test Centre of Centre Manager has been records kit stock and kit ID has been generated | **Post-condition** | The Centre Manager can update the stock of test kit |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Step** | **User Input** | **Desired Output Response** | **Real Output Response** | **Result** | **Comment** |
| 1 | Select manage test kit stock | 3 | Display the manage test kit stock menu | Display the manage test kit stock menu | Pass | N/A |
| 2 | Select update stock | 2 | Show test kits stored by the test centre where the centre manager works | Show test kits stored by the test centre where the centre manager works | Pass | N/A |
| 3 | Enter kit ID | K002 | Kit ID is valid and the stock update form input is displayed | Warning message is shown “Invalid kit ID! Try another” and enter kit ID form input is displayed again | Pass | Validation: Input kit ID must be valid as shown |
| 4 | Enter kit ID | K003 | Kit ID is valid and the stock update form input is displayed | Kit ID is valid and the stock update form input is displayed | Pass | N/A |
| 5 | Update stock | 5 | The stock of Test Kit is updated | The stock of Test Kit is updated | Pass | N/A |
| 6 | Press (E/e) to log out | E | Back to Covid Testing Information System (CTIS) menu | Back to Covid Testing Information System (CTIS) menu | Pass | N/A |

## Record New Test

### Tester Login 1

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Name** | Record New Test | **Case name** | Tester Login 1 |
| **Description** | Tester Login using Tester account | **Priority** | High |
| **Pre-condition** | The Tester account has been recorded and has been set up as Tester. Username and password must be the same as registered | **Post-condition** | Tester can access tester menu |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Step** | **User Input** | **Desired Output Response** | **Real Output Response** | **Result** | **Comment** |
| 1 | Select tester login | 3 | The program displays the tester login menu | The program displays the tester login menu | Pass | N/A |
| 2 | Enter username | testersurya | Username is valid and password is form is displayed | Username is valid and password is form is displayed | Pass | N/A |
| 3 | Enter password | tester987\*\*& | Password is valid and tester menu displayed | Password is valid and tester menu displayed | Pass | N/A |

### Record New Test 1

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Name** | Record New Test | **Case name** | Record New Test 1 |
| **Description** | Tester records or updates Patient information and administer test for registered Patient. | **Priority** | High |
| **Pre-condition** | If the Patient has not taken the test, the Tester records the Patient information. If the Patient has taken the test, the Tester updates the Patient information. | **Post-condition** | Tester can administer tests for registered Patients |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Step** | **User Input** | **Desired Output Response** | **Real Output Response** | **Result** | **Comment** |
| 1 | Select record new test | 1 | The program displays the patient registration form | The program displays the patient registration form | Pass | N/A |
| 2 | Enter username | surya | Username is valid and password input form is displayed | Username is valid and password is form is displayed | Pass | N/A |
| 3 | Enter password | surya&86 | Password is valid and name input form is displayed | Password is valid and tester menu displayed | Pass | N/A |
| 4 | Enter name | Surya Pradipta | Name is valid and patient types is displayed | Name is valid and patient types is displayed | Pass | N/A |
| 5 | Select patient type | 4 | Patient type is valid and symptoms input form is displayed | Patient type is valid and symptoms input form is displayed | Pass | Patient type has been created by hard-coded in Patient class |
| 6 | Enter symptoms | a high temperature | Symptoms is valid and Test Centre’s name is displayed | Symptoms is valid and Test Centre’s name is displayed | Pass | N/A |
| 7 | Select Test Centre | 2 | Test Centre is valid and available test kits is displayed | Test Centre is valid and available test kits is displayed | Pass | N/A |
| 8 | Enter kit ID | K002 | Kit ID is valid. Covid test date and test id created and test status pending. Patient registration is successful and the covid test is recorded to the Patient, Tester, and Test Kit.  Available stock Test Kit minus one | Kit ID is valid. Covid test date and test id created and test status pending. Patient registration is successful and the covid test is recorded to the Patient, Tester, and Test Kit.  Available stock Test Kit minus one | Pass | N/A |

### Record New Test 2

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Name** | Record New Test | **Case name** | Record New Test 2 |
| **Description** | Tester records or updates Patient information and administer test for registered Patient. | **Priority** | High |
| **Pre-condition** | If the Patient has not taken the test, the Tester records the Patient information. If the Patient has taken the test, the Tester updates the Patient information. | **Post-condition** | Tester can administer tests for registered Patients |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Step** | **User Input** | **Desired Output Response** | **Real Output Response** | **Result** | **Comment** |
| 1 | Select record new test | 1 | The program displays the patient registration form | The program displays the patient registration form | Pass | N/A |
| 2 | Enter username | amara | Username is valid and password input form is displayed | Username is valid and password is form is displayed | Pass | N/A |
| 3 | Enter password | amara123 | Password is valid and name input form is displayed | Password is valid and tester menu displayed | Pass | N/A |
| 4 | Enter name | Amara | Name is valid and patient types is displayed | Name is valid and patient types is displayed | Pass | N/A |
| 5 | Select patient type | 3 | Patient type is valid and symptoms input form is displayed | Patient type is valid and symptoms input form is displayed | Pass | Patient type has been created by hard-coded in Patient class |
| 6 | Enter symptoms | continuous cough | Symptoms is valid and Test Centre’s name is displayed | Symptoms is valid and Test Centre’s name is displayed | Pass | N/A |
| 7 | Select Test Centre | 2 | Test Centre is valid and available test kits is displayed | Test Centre is valid and available test kits is displayed | Pass | There are two Test Centres that provide tests with available test kits |
| 7 | Enter kid ID | K004 | Kit ID is valid. Covid test date and test id created and test status pending. Patient registration is successful and the covid test is recorded to the Patient, Tester, and Test Kit.  Available stock Test Kit minus one | Warning message is shown “Invalid kit ID! Try again” | Pass | Validation: there are only one available test kit with status K001 |
| 8 | Enter kit ID | K001 | Kit ID is valid. Covid test date and test id created and test status pending. Patient registration is successful and the covid test is recorded to the Patient, Tester, and Test Kit.  Available stock Test Kit minus one | Kit ID is valid. Covid test date and test id created and test status pending. Patient registration is successful and the covid test is recorded to the Patient, Tester, and Test Kit.  Available stock Test Kit minus one | Pass | N/A |

### Record New Test 3

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Name** | Record New Test | **Case name** | Record New Test 3 |
| **Description** | Tester records or updates Patient information and administer test for registered Patient. | **Priority** | High |
| **Pre-condition** | If the Patient has not taken the test, the Tester records the Patient information. If the Patient has taken the test, the Tester updates the Patient information. | **Post-condition** | Tester can administer tests for registered Patients |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Step** | **User Input** | **Desired Output Response** | **Real Output Response** | **Result** | **Comment** |
| 1 | Select record new test | 1 | The program displays the patient registration form | The program displays the patient registration form | Pass | N/A |
| 2 | Enter username | surya | Username is valid and password input form is displayed | Username is valid and password is form is displayed | Pass | N/A |
| 3 | Enter password | surya&86 | Password is valid and name input form is displayed | Password is valid and tester menu displayed | Pass | N/A |
| 4 | Enter name | Surya Pradipta | Name is valid and patient types is displayed | Name is valid and patient types is displayed | Pass | N/A |
| 5 | Select patient type | 2 | Patient type is valid and symptoms input form is displayed | Patient type is valid and symptoms input form is displayed | Pass | Patient type has been created by hard-coded in Patient class |
| 6 | Enter symptoms | continuous cough | Symptoms is valid and Test Centre’s name is displayed | Symptoms is valid and Test Centre’s name is displayed | Pass | N/A |
| 7 | Select Test Centre | 2 | Test Centre is valid and available test kits is displayed | Test Centre is valid and available test kits is displayed | Pass | N/A |
| 8 | Enter kit ID | K001 | Kit ID is valid. Covid test date and test id created and test status pending. Patient has taken test before, Tester update the patient type and symptoms only. The covid test is recorded to the Patient, Tester, and Test Kit.  Available stock Test Kit minus one | Kit ID is valid. Covid test date and test id created and test status pending. Patient has taken test before, Tester update the patient type and symptoms only. The covid test is recorded to the Patient, Tester, and Test Kit.  Available stock Test Kit minus one | Pass | Patient has taken test, Tester update the patient type and symptoms only |
| 9 | Press (E/e) to log out | e | Back to Covid Testing Information System (CTIS) menu | Back to Covid Testing Information System (CTIS) menu | Pass | N/A |

### Tester Login 2

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Name** | Record New Test | **Case name** | Tester Login 2 |
| **Description** | Tester Login using Tester account | **Priority** | High |
| **Pre-condition** | The Tester account has been recorded and has been set up as Tester. Username and password must be the same as registered | **Post-condition** | Tester can access tester menu |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Step** | **User Input** | **Desired Output Response** | **Real Output Response** | **Result** | **Comment** |
| 1 | Select tester login | 3 | The program displays the tester login menu | The program displays the tester login menu | Pass | N/A |
| 2 | Enter username | testerpradipta | Username is valid and password is form is displayed | Username is valid and password is form is displayed | Pass | N/A |
| 3 | Enter password | tester12 | Password is valid and admin menu displayed | Warning message is shown “Can't find your Tester account”.  Back to Covid Testing Information System (CTIS) menu | Pass | Validation: username and password must be the same as registered |
| 4 | Select tester login | 3 | The program displays the tester login menu | The program displays the tester login menu | Pass | N/A |
| 5 | Enter username | testerpradipta | Username is valid and password is form is displayed | Username is valid and password is form is displayed | Pass | N/A |
| 6 | Enter password | tester123 | Password is valid and admin menu displayed | Password is valid and admin menu displayed | Pass | N/A |

### Record New Test 4

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Name** | Record New Test | **Case name** | Record New Test 4 |
| **Description** | Tester records or updates Patient information and administer test for registered Patient. | **Priority** | High |
| **Pre-condition** | If the Patient has not taken the test, the Tester records the Patient information. If the Patient has taken the test, the Tester updates the Patient information. | **Post-condition** | Tester can administer tests for registered Patients |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Step** | **User Input** | **Desired Output Response** | **Real Output Response** | **Result** | **Comment** |
| 1 | Select record new test | 1 | The program displays the patient registration form | The program displays the patient registration form | Pass | N/A |
| 2 | Enter username | rani | Username is valid and password input form is displayed | Username is valid and password is form is displayed | Pass | N/A |
| 3 | Enter password | Ran56% | Password is valid and name input form is displayed | Password is valid and tester menu displayed | Pass | N/A |
| 4 | Enter name | Rani | Name is valid and patient types is displayed | Name is valid and patient types is displayed | Pass | N/A |
| 5 | Select patient type | 3 | Patient type is valid and symptoms input form is displayed | Patient type is valid and symptoms input form is displayed | Pass | Patient type has been created by hard-coded in Patient class |
| 6 | Enter symptoms | loss or change in the sense of smell or taste | Symptoms is valid and Test Centre’s name is displayed | Symptoms is valid and Test Centre’s name is displayed | Pass | N/A |
| 7 | Select Test Centre | 1 | Test Centre is valid and available test kits is displayed | Test Centre is valid and available test kits is displayed | Pass | N/A |
| 8 | Enter kit ID | K003 | Kit ID is valid. Covid test date and test id created and test status pending. Patient registration is successful and the covid test is recorded to the Patient, Tester, and Test Kit.  Available stock Test Kit minus one | Kit ID is valid. Covid test date and test id created and test status pending. Patient registration is successful and the covid test is recorded to the Patient, Tester, and Test Kit.  Available stock Test Kit minus one | Pass | N/A |
| 9 | Press (E/e) to log out | e | Back to Covid Testing Information System (CTIS) menu | Back to Covid Testing Information System (CTIS) menu | Pass | N/A |

## Update Result

### Tester Login 1

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Name** | Update Result | **Case name** | Tester Login 1 |
| **Description** | Tester Login using Tester account | **Priority** | High |
| **Pre-condition** | The Tester account has been recorded and has been set up as Tester. Username and password must be the same as registered | **Post-condition** | Tester can access tester menu |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Step** | **User Input** | **Desired Output Response** | **Real Output Response** | **Result** | **Comment** |
| 1 | Select tester login | 3 | The program displays the tester login menu | The program displays the tester login menu | Pass | N/A |
| 2 | Enter username | testerpradipta | Username is valid and password is form is displayed | Username is valid and password is form is displayed | Pass | N/A |
| 3 | Enter password | tester123 | Password is valid and tester menu displayed | Password is valid and tester menu displayed | Pass | N/A |

### Update Result 1

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Name** | Update Result | **Case name** | Update Result 1 |
| **Description** | Tester updates Patient's covid Results | **Priority** | High |
| **Pre-condition** | Patient has record covid test | **Post-condition** | Tester can update test information |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Step** | **User Input** | **Desired Output Response** | **Real Output Response** | **Result** | **Comment** |
| 1 | Select update Result | 1 | Program displays test ID | Program displays test ID | Pass | N/A |
| 2 | Enter test ID | T004 | Covid test, patient, test centre, test kit and result test information is displayed | Covid test, patient, test centre, test kit information is displayed | Pass | N/A |
| 3 | Select result test | 1 | Result is updated. The result date is recorded and status of covid test is set to “complete” | Result is updated | Pass | 1 is positive and 2 is negative |
| 4 | Press (E/e) to log out | e | Back to Covid Testing Information System (CTIS) menu | Back to Covid Testing Information System (CTIS) menu | Pass | N/A |

### Tester Login 2

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Name** | Update Result | **Case name** | Tester Login 1 |
| **Description** | Tester Login using Tester account | **Priority** | High |
| **Pre-condition** | The Tester account has been recorded and has been set up as Tester. Username and password must be the same as registered | **Post-condition** | Tester can access tester menu |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Step** | **User Input** | **Desired Output Response** | **Real Output Response** | **Result** | **Comment** |
| 1 | Select tester login | 3 | The program displays the tester login menu | The program displays the tester login menu | Pass | N/A |
| 2 | Enter username | testersurya | Username is valid and password is form is displayed | Username is valid and password is form is displayed | Pass | N/A |
| 3 | Enter password | tester987\*\*& | Password is valid and tester menu displayed | Password is valid and tester menu displayed | Pass | N/A |

### Update Result 2

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Name** | Update Result | **Case name** | Update Result 2 |
| **Description** | Tester updates Patient's covid Results | **Priority** | High |
| **Pre-condition** | Patient has record covid test | **Post-condition** | Tester can update test information |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Step** | **User Input** | **Desired Output Response** | **Real Output Response** | **Result** | **Comment** |
| 1 | Select update Result | 1 | Program displays test ID | Program displays test ID | Pass | N/A |
| 2 | Enter test ID | T003 | Covid test, patient, test centre, test kit and result test information is displayed | Covid test, patient, test centre, test kit information is displayed | Pass | N/A |
| 3 | Select result test | 1 | Result is updated. The result date is recorded and status of covid test is set to “complete” | Result is updated | Pass | 1 is positive and 2 is negative |

### Update Result 3

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Name** | Update Result | **Case name** | Update Result 3 |
| **Description** | Tester updates Patient's covid Results | **Priority** | High |
| **Pre-condition** | Patient has record covid test | **Post-condition** | Tester can update test information |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Step** | **User Input** | **Desired Output Response** | **Real Output Response** | **Result** | **Comment** |
| 1 | Select update Result | 1 | Program displays test ID | Program displays test ID | Pass | N/A |
| 2 | Enter test ID | T002 | Covid test, patient, test centre, test kit and result test information is displayed | Covid test, patient, test centre, test kit information is displayed | Pass | N/A |
| 3 | Select result test | 2 | Result is updated. The result date is recorded and status of covid test is set to “complete” | Result is updated | Pass | 1 is positive and 2 is negative |
| 4 | Press (E/e) to log out | e | Back to Covid Testing Information System (CTIS) menu | Back to Covid Testing Information System (CTIS) menu | Pass | N/A |

## View Test History

### Login Patient 1

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Name** | View Test History | **Case name** | Login Patient 1 |
| **Description** | Patient login using Patient account | **Priority** | High |
| **Pre-condition** | Patient has been taken a test. Username and password must be the same as registered | **Post-condition** | Patients can see the history of tests that have been taken |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Step** | **User Input** | **Desired Output Response** | **Real Output Response** | **Result** | **Comment** |
| 1 | Select patient login | 4 | The program displays the patient login menu | The program displays the patient login menu | Pass | N/A |
| 2 | Enter username | surya | Username is valid and password is form is displayed | Username is valid and password is form is displayed | Pass | N/A |
| 3 | Enter password | surya&86 | Password is valid and patient menu displayed | Password is valid and patient menu displayed | Pass | N/A |

### View Test History 1

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Name** | View Test History | **Case name** | View Test History 1 |
| **Description** | Patient login using Patient account | **Priority** | High |
| **Pre-condition** | Patient has been taken a test. | **Post-condition** | Patients can see the history of tests that have been taken |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Step** | **User Input** | **Desired Output Response** | **Real Output Response** | **Result** | **Comment** |
| 1 | Select view test history | 1 | The program displays test history of Patient | The program displays test history of Patient | Pass | N/A |
| 2 | Press (E/e) to log out | e | Back to Covid Testing Information System (CTIS) menu | Back to Covid Testing Information System (CTIS) menu | Pass | N/A |

### Login Patient 2

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Name** | View Test History | **Case name** | Login Patient 2 |
| **Description** | Patient login using Patient account | **Priority** | High |
| **Pre-condition** | Patient has been taken a test. Username and password must be the same as registered | **Post-condition** | Patients can see the history of tests that have been taken |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Step** | **User Input** | **Desired Output Response** | **Real Output Response** | **Result** | **Comment** |
| 1 | Select patient login | 4 | The program displays the patient login menu | The program displays the patient login menu | Pass | N/A |
| 2 | Enter username | amara | Username is valid and password is form is displayed | Username is valid and password is form is displayed | Pass | N/A |
| 3 | Enter password | amara123 | Password is valid and patient menu displayed | Password is valid and patient menu displayed | Pass | N/A |

### View Test History 2

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Name** | View Test History | **Case name** | View Test History 2 |
| **Description** | Patient login using Patient account | **Priority** | High |
| **Pre-condition** | Patient has been taken a test. | **Post-condition** | Patients can see the history of tests that have been taken |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Step** | **User Input** | **Desired Output Response** | **Real Output Response** | **Result** | **Comment** |
| 1 | Select view test history | 1 | The program displays test history of Patient | The program displays test history of Patient | Pass | N/A |
| 2 | Press (E/e) to log out | e | Back to Covid Testing Information System (CTIS) menu | Back to Covid Testing Information System (CTIS) menu | Pass | N/A |

### Login Patient 3

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Name** | View Test History | **Case name** | Login Patient 3 |
| **Description** | Patient login using Patient account | **Priority** | High |
| **Pre-condition** | Patient has been taken a test. Username and password must be the same as registered | **Post-condition** | Patients can see the history of tests that have been taken |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Step** | **User Input** | **Desired Output Response** | **Real Output Response** | **Result** | **Comment** |
| 1 | Select patient login | 4 | The program displays the patient login menu | The program displays the patient login menu | Pass | N/A |
| 2 | Enter username | rani | Username is valid and password is form is displayed | Username is valid and password is form is displayed | Pass | N/A |
| 3 | Enter password | Ran56% | Password is valid and patient menu displayed | Password is valid and patient menu displayed | Pass | N/A |

### View Test History 3

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Name** | View Test History | **Case name** | View Test History 3 |
| **Description** | Patient login using Patient account | **Priority** | High |
| **Pre-condition** | Patient has been taken a test. | **Post-condition** | Patients can see the history of tests that have been taken |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Step** | **User Input** | **Desired Output Response** | **Real Output Response** | **Result** | **Comment** |
| 1 | Select view test history | 1 | The program displays test history of Patient | The program displays test history of Patient | Pass | N/A |
| 2 | Press (E/e) to log out | e | Back to Covid Testing Information System (CTIS) menu | Back to Covid Testing Information System (CTIS) menu | Pass | N/A |

## Generate Test Report

### Test Centre Manager Login 1

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Name** | Generate Test Report | **Case name** | Test Centre Manager Login 1 |
| **Description** | Test Centre Manager Login using Test Centre Manager account | **Priority** | High |
| **Pre-condition** | The Centre Manager Account has been created and has been set up as Centre Officer. Username and password must be the same as registered | **Post-condition** | Test Centre Manager can access test centre manager menu |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Step** | **User Input** | **Desired Output Response** | **Real Output Response** | **Result** | **Comment** |
| 1 | Select test centre manager login | 2 | The program displays the test centre manager login menu | The program displays the test centre manager login menu | Pass | N/A |
| 2 | Enter username | adminsurya | Username is valid and password is form is displayed | Username is valid and password is form is displayed | Pass | N/A |
| 3 | Enter password | admin\*&^2 | Password is valid and admin menu displayed | Password is valid and admin menu displayed | Pass | N/A |

### View Test History Report 1

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Name** | Generate Test Report | **Case name** | View Test History Report 1 |
| **Description** | Test Centre Officer view a test report | **Priority** | High |
| **Pre-condition** | Test history of all tests that have been administered at the test centre where the officer is employed | **Post-condition** | Centre Manager and Tester can view test report |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Step** | **User Input** | **Desired Output Response** | **Real Output Response** | **Result** | **Comment** |
| 1 | Select test view test history report | 4 | The program displays test history of Patient | The program displays test history of Patient | Pass | There are three patient tests displayed: T001, T002, and T003 |
| 2 | Press (E/e) to log out | e | Back to Covid Testing Information System (CTIS) menu | Back to Covid Testing Information System (CTIS) menu | Pass | N/A |

### Test Centre Manager Login 2

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Name** | Generate Test Report | **Case name** | Test Centre Manager Login 2 |
| **Description** | Test Centre Manager Login using Test Centre Manager account | **Priority** | High |
| **Pre-condition** | The Centre Manager Account has been created and has been set up as Centre Officer. Username and password must be the same as registered | **Post-condition** | Test Centre Manager can access test centre manager menu |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Step** | **User Input** | **Desired Output Response** | **Real Output Response** | **Result** | **Comment** |
| 1 | Select test centre manager login | 2 | The program displays the test centre manager login menu | The program displays the test centre manager login menu | Pass | N/A |
| 2 | Enter username | adminpradipta | Username is valid and password is form is displayed | Username is valid and password is form is displayed | Pass | N/A |
| 3 | Enter password | admin123 | Password is valid and admin menu displayed | Password is valid and admin menu displayed | Pass | N/A |

### View Test History Report 2

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Name** | Generate Test Report | **Case name** | View Test History Report 2 |
| **Description** | Test Centre Officer view a test report | **Priority** | High |
| **Pre-condition** | Test history of all tests that have been administered at the test centre where the officer is employed | **Post-condition** | Centre Manager and Tester can view test report |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Step** | **User Input** | **Desired Output Response** | **Real Output Response** | **Result** | **Comment** |
| 1 | Select test view test history report | 4 | The program displays test history of Patient | The program displays test history of Patient | Pass | There is one patient tests displayed: T004 |
| 2 | Press (E/e) to log out | e | Back to Covid Testing Information System (CTIS) menu | Back to Covid Testing Information System (CTIS) menu | Pass | N/A |

### Tester Login 1

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Name** | Generate Test Report | **Case name** | Tester Login 1 |
| **Description** | Tester login using Tester account | **Priority** | High |
| **Pre-condition** | The Tester account has been recorded and has been set up as Tester. Username and password must be the same as registered | **Post-condition** | Tester can access tester menu |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Step** | **User Input** | **Desired Output Response** | **Real Output Response** | **Result** | **Comment** |
| 1 | Select tester login | 3 | The program displays the tester login menu | The program displays the tester login menu | Pass | N/A |
| 2 | Enter username | testersurya | Username is valid and password is form is displayed | Username is valid and password is form is displayed | Pass | N/A |
| 3 | Enter password | tester987\*\*& | Password is valid and tester menu displayed | Password is valid and tester menu displayed | Pass | N/A |

### View Test History Report 3

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Name** | Generate Test Report | **Case name** | View Test History Report 3 |
| **Description** | Test Centre Officer view a test report | **Priority** | High |
| **Pre-condition** | Test history of all tests that have been administered at the test centre where the officer is employed | **Post-condition** | Centre Manager and Tester can view test report |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Step** | **User Input** | **Desired Output Response** | **Real Output Response** | **Result** | **Comment** |
| 1 | Select test view test history report | 3 | The program displays test history of Patient | The program displays test history of Patient | Pass | There are three patient tests displayed: T001, T002, T003 |
| 2 | Press (E/e) to log out | e | Back to Covid Testing Information System (CTIS) menu | Back to Covid Testing Information System (CTIS) menu | Pass | N/A |

### Tester Login 2

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Name** | Generate Test Report | **Case name** | Tester Login 2 |
| **Description** | Tester login using Tester account | **Priority** | High |
| **Pre-condition** | The Tester account has been recorded and has been set up as Tester. Username and password must be the same as registered | **Post-condition** | Tester can access tester menu |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Step** | **User Input** | **Desired Output Response** | **Real Output Response** | **Result** | **Comment** |
| 1 | Select tester login | 3 | The program displays the tester login menu | The program displays the tester login menu | Pass | N/A |
| 2 | Enter username | testerpradipta | Username is valid and password is form is displayed | Username is valid and password is form is displayed | Pass | N/A |
| 3 | Enter password | tester123 | Password is valid and tester menu displayed | Password is valid and tester menu displayed | Pass | N/A |

### View Test History Report 4

|  |  |  |  |
| --- | --- | --- | --- |
| **Scenario Name** | Generate Test Report | **Case name** | View Test History Report 4 |
| **Description** | Test Centre Officer view a test report | **Priority** | High |
| **Pre-condition** | Test history of all tests that have been administered at the test centre where the officer is employed | **Post-condition** | Centre Manager and Tester can view test report |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **No** | **Step** | **User Input** | **Desired Output Response** | **Real Output Response** | **Result** | **Comment** |
| 1 | Select test view test history report | 3 | The program displays test history of Patient | The program displays test history of Patient | Pass | There is one patient tests displayed: T004 |
| 2 | Press (E/e) to log out | e | Back to Covid Testing Information System (CTIS) menu | Back to Covid Testing Information System (CTIS) menu | Pass | N/A |
| 3 | Press (E/e) to exit | e | Program closed | Program closed | Pass | N/A |