Testing and Code Inspection

Group 7

HOLLO Project

### **Darias Skiedra, Mateusz Wszolek, Shreyas Gaonkar, Heeba Mohammed**

### **CS440 - Fall 2015**

### **University of Illinois Chicago**

**1 Test Cases:**

|  |  |
| --- | --- |
| *Test Case Identifier* | Login |
| ***Test Phase*** | Unit Test |
| ***Test Feature*** | Successful logon to the application  User is taken to Main Menu |
| ***Pass / Fail Criteria*** | User must be able to connect to the server with their credentials contained in a database |
| ***Pre-Conditions*** | * App is installed on the device * User has registered successfully |
| ***Test Procedure*** | * Open the application * Enter username and password in the fields provided in the login panel * System searches for User/Password combination in the databases * If credentials are matching in the database, fetch user’s information * Connect to app with the users’ account information |
| ***Special Requirements*** | * None |

PASS

|  |  |
| --- | --- |
| *Test Case Identifier* | Create Account |
| ***Test Phase*** | Unit Test |
| ***Test Feature*** | Successful account creation  User is taken to Main Menu |
| ***Pass / Fail Criteria*** | User must be able to create an account and the details are stored in the database. |
| ***Pre-Conditions*** | * App is installed on the device * Same account does not exist before. |
| ***Test Procedure*** | * Open the application * Enter details like name, password, phone number, email, dob, etc. in the fields provided in the registration panel * System searches whether the details exist already in the database * If the account does not exist already, a new account is created with the information provided * Take the user to the Main Menu |
| ***Special Requirements*** | * None |

PASS

|  |  |
| --- | --- |
| *Test Case Identifier* | Modify Contact |
| ***Test Phase*** | Unit Test |
| ***Test Feature*** | Successful modification of a contact  User is able to add/edit/delete a contact |
| ***Pass / Fail Criteria*** | User must be able to add/edit/delete a contact |
| ***Pre-Conditions*** | * User is logged in * User has some details to add/edit/delete the contact. |
| ***Test Procedure*** | * Open the application * Select the ‘Contacts’ menu. * To add a contact, click on add, then fill in all the details like name, phone number, email and address of the contact to be added.   To edit an existing contact, search for the contact, select the contact, click on edit and edit the details.  To delete an existing contact, search for the contact, select the contact, click on delete and delete the details.   * The contact is modified. |
| ***Special Requirements*** | * None |

PASS

|  |  |
| --- | --- |
| *Test Case Identifier* | Make Hollo Call |
| ***Test Phase*** | Unit Test |
| ***Test Feature*** | Successful Hollo call with another Hollo user |
| ***Pass / Fail Criteria*** | User is able to establish a Hollo call with a contact |
| ***Pre-Conditions*** | * User is logged in * The contact is also a Hollo user |
| ***Test Procedure*** | * Open the application * Select the ‘Contacts’ menu * Select a contact * Click on the option to make a Hollo call * The system establishes connection with the contact and displays the holographic projection |
| ***Special Requirements*** | * Holographic camera is installed |

FAIL

|  |  |
| --- | --- |
| *Test Case Identifier* | Voice commands |
| ***Test Phase*** | Unit Test |
| ***Test Feature*** | Certain features of the app can be controlled via voice commands |
| ***Pass / Fail Criteria*** | User is able to perform the functions correctly by issuing voice commands |
| ***Pre-Conditions*** | * User is logged in * Microphone is enabled and not in mute mode |
| ***Test Procedure*** | * The user clicks on ‘Listen’ button * The user gives various commands as below:   Hollo Call <contact name>  Select <contact name>  Traditional Call <contact name>  Where is <contact name>   * The system responds by doing the appropriate action as per the voice command |
| ***Special Requirements*** | * Holographic camera is enabled |

FAIL

|  |  |
| --- | --- |
| *Test Case Identifier* | Make Traditional Call |
| ***Test Phase*** | Unit Test |
| ***Test Feature*** | Successful traditional call with another Hollo user |
| ***Pass / Fail Criteria*** | User is able to establish a traditional call with a contact |
| ***Pre-Conditions*** | * User is logged in * The contact is also a Hollo user |
| ***Test Procedure*** | * Open the application * Select the ‘Contacts’ menu * Select a contact * Click on the option to make a traditional call * The system establishes connection with the contact |
| ***Special Requirements*** | * None |

FAIL

|  |  |
| --- | --- |
| *Test Case Identifier* | Receive Traditional Call |
| ***Test Phase*** | Unit Test |
| ***Test Feature*** | Successful traditional call from another Hollo user |
| ***Pass / Fail Criteria*** | User is able to receive a traditional call with a contact |
| ***Pre-Conditions*** | * User is logged in * The caller is a Hollo user |
| ***Test Procedure*** | * The user receives a call and is indicated by a ring tone and display * The user clicks on ‘Answer traditional call’ button * The user is able to listen to the caller without any interruption |
| ***Special Requirements*** | * None |

FAIL

|  |  |
| --- | --- |
| *Test Case Identifier* | Receive Hollo Call |
| ***Test Phase*** | Unit Test |
| ***Test Feature*** | Successful Hollo call from another Hollo user |
| ***Pass / Fail Criteria*** | User is able to receive a Hollo call with a contact |
| ***Pre-Conditions*** | * User is logged in * The caller is a Hollo user |
| ***Test Procedure*** | * The user receives a call and is indicated by a ring tone and display * The user clicks on ‘Answer Hollo call’ button * The user is able to listen to the caller without any interruption and the holographic projection is enabled and working |
| ***Special Requirements*** | * Holographic camera is enabled |

FAIL

|  |  |
| --- | --- |
| *Test Case Identifier* | Locate User with Map |
| ***Test Phase*** | Unit Test |
| ***Test Feature*** | A contact’s current location is identified and displayed via holographic projection |
| ***Pass / Fail Criteria*** | User is able to locate a contact accurately |
| ***Pre-Conditions*** | * User is logged in * The contact is a Hollo user * The contact has enabled ‘Auto-detect location’ |
| ***Test Procedure*** | * The user selects a contact * The user clicks on ‘Locate contact’ button * The system checks if the contact has enabled ‘Auto-detect’ location * The system projects the location of a map as hologram, indicating the current location of the contact |
| ***Special Requirements*** | * Holographic camera is enabled |

FAIL

|  |  |
| --- | --- |
| *Test Case Identifier* | Select contact using voice command |
| ***Test Phase*** | Integration Test |
| ***Test Feature*** | System allows to select a contact using voice command and the contact’s image is projected correctly |
| ***Pass / Fail Criteria*** | User is able to select a contact by issuing voice command |
| ***Pre-Conditions*** | * User is in the Contacts menu * There is a list of contacts |
| ***Test Procedure*** | * The user issues a voice command: select <contact name> * The system selects the contact mentioned by the user * If the contact is not present in the contact list, the system throws a message “Contact not found” |
| ***Special Requirements*** | * A list of contacts in the Contacts menu |

FAIL

|  |  |
| --- | --- |
| *Test Case Identifier* | Make Hollo call using voice command |
| ***Test Phase*** | Integration Test |
| ***Test Feature*** | System allows to make a Hollo call using voice command and the contact’s image is projected correctly |
| ***Pass / Fail Criteria*** | User is able to make a Hollo call by issuing voice command |
| ***Pre-Conditions*** | * There is a list of contacts * A contact from the contact list is selected |
| ***Test Procedure*** | * The user issues a voice command: Hollo call <contact name> * The system establishes a connection with the contact and displays the holographic projection * If the contact is unavailable to attend the call, the system throws a message “contact unavailable to pick your call” and allows the user to leave a voice message to the contact |
| ***Special Requirements*** | * A list of contacts in the Contacts menu |

FAIL

|  |  |
| --- | --- |
| *Test Case Identifier* | Run Hollo on different platforms |
| ***Test Phase*** | Compatibility Test |
| ***Test Feature*** | App runs on different configurations of hardware and software |
| ***Pass / Fail Criteria*** | App runs successfully on different platforms |
| ***Pre-Conditions*** | * There are different configurations of hardware and software * There are different types of hand-held devices |
| ***Test Procedure*** | * For each type of hand-held device, install the app * After successful installation, try different combinations of functionalities like call contact, locate contact, etc. * The functionalities work correctly in each of the hand-held devices. * The calls made from one type of hardware and software is able to be received from another type of hardware and software |
| ***Special Requirements*** | * None |

PASS

|  |  |
| --- | --- |
| *Test Case Identifier* | Change user status |
| ***Test Phase*** | Integration Test |
| ***Test Feature*** | Test if the user can successfully change their status in Database |
| ***Pass / Fail Criteria*** | Status change reflected in the Database |
| ***Pre-Conditions*** | * User has created an account with a status |
| ***Test Procedure*** | * Open app and login * Write a new status and hit change status button * Newly changed status is reflected in Database |
| ***Special Requirements*** | * None |

## 

PASS

|  |  |
| --- | --- |
| *Test Case Identifier* | Make Hollo Call |
| ***Test Phase*** | Usability Test |
| ***Test Feature*** | The “Make Hollo Call” feature is easy to use |
| ***Pass / Fail Criteria*** | Ease of use |
| ***Pre-Conditions*** | * Tester is a first time user of the app |
| ***Test Procedure*** | * Open the app * Select the contact * Click on ‘Make Hollo Call’ button * The system establishes connection with the contact * Indicate the usability of the feature |
| ***Special Requirements*** | * None |

FAIL

|  |  |
| --- | --- |
| *Test Case Identifier* | Make Hollo conference Call |
| ***Test Phase*** | Multiuser Test |
| ***Test Feature*** | Make a Hollo conference call |
| ***Pass / Fail Criteria*** | Supports multiple users in the same Hollo call |
| ***Pre-Conditions*** | * Users have Hollo account and are logged in |
| ***Test Procedure*** | * Open the app * Select the contact * Click on ‘Make Hollo Call’ button * Click on the ‘Add more contacts to call’ button * All the contacts added to call are connected and holographic projection is displayed |
| ***Special Requirements*** | * None |

FAIL

|  |  |
| --- | --- |
| *Test Case Identifier* | Make call |
| ***Test Phase*** | System Test |
| ***Test Feature*** | Make a call |
| ***Pass / Fail Criteria*** | Contact is called successfully |
| ***Pre-Conditions*** | * User has Hollo account * User is logged in to the application |
| ***Test Procedure*** | * Open the app * Add a contact * Click on ‘Make Hollo Call’ button * Connection is established with the contact * Holographic projection is displayed |
| ***Special Requirements*** | * None |

FAIL

|  |  |
| --- | --- |
| *Test Case Identifier* | Hollo |
| ***Test Phase*** | Acceptance Test |
| ***Test Feature*** | Test if all the requirements are met |
| ***Pass / Fail Criteria*** | All the requirements are met |
| ***Pre-Conditions*** | * App is developed |
| ***Test Procedure*** | * Download the app * Install the app * Create account * Login to the app * System takes the user to the Main Screen * Add contacts * Make calls/Receive calls * Establish connection successfully * Click on exit * Exit the app |
| ***Special Requirements*** | * None |

## 

PARTIAL PASS

|  |  |
| --- | --- |
| *Test Case Identifier* | Send message |
| ***Test Phase*** | Acceptance Test |
| ***Test Feature*** | Test if the messaging system works |
| ***Pass / Fail Criteria*** | Message is sent |
| ***Pre-Conditions*** | * App is developed |
| ***Test Procedure*** | * Select a contact to send a message to * Type in text * Press Send |
| ***Special Requirements*** | * None |

## PASS

|  |  |
| --- | --- |
| *Test Case Identifier* | Receive message |
| ***Test Phase*** | Acceptance Test |
| ***Test Feature*** | Test if messaging system works |
| ***Pass / Fail Criteria*** | Message is received |
| ***Pre-Conditions*** | * Another user sent a message |
| ***Test Procedure*** | * Open contacts * Select user from whom you have a message * Message is displayed in the text window |
| ***Special Requirements*** | * None |

## 

PASS

**2 Unit Tests:**

Unit tests added:

* Concurrent access to the User Manager (Server)
* Receiving message from a user who is not logged in (Server)
* Sending errors to users (Server)
* Log in input (Server)
* Input Field for username (Desktop Client)
* Input Field for Password (Desktop Client)
* Status updates (Desktop Client)
* User Verification (Server)
* Log-in (Android Client)
* Account Create (Android Client)
* Message sending (Android Client)

**3 System Testing:**

Functional Tests notes:

Completing user creation is intuitive. Log in is also simple and quick. No ability to send messages when hitting enter, a “send” button must be pressed instead, and this should be looked at. There are missing labels for some functionality, which is self-explanatory though (For example, the status selection on the desktop client.) On the Android client the text starts in the middle of the screen.

Performance Testing notes:

1. Server works with limited number of concurrent users. Right now only one Amazon server is deployed
2. Messages are delivered quickly
3. Connecting to server from clients is fairly immediate, no extensive delays noticed.
4. Messages are stored in-memory, no current system for non-volatile storage.

**4 Code Inspection notes:**

1. **Android:**
   1. Some comments were not updated to reflect code
   2. Some important functionality is not currently commented
   3. Some poor-reuse of code, only small portion
2. **Desktop Application:**
   1. No header comments in few of the functions
   2. Include code has specific paths pointing to the local server
   3. Debug messages in production
   4. Unfinished TODO comments
   5. Unreachable code
3. **Server:**
   1. The {} are not used consistently
   2. Indentation is not consistent
   3. Some comments were not updated to reflect code
   4. Unused code (definitions for calling and video conferences)

**5 Appendix:**

1. Android Code:
   1. Account Create

**public static void** create(String name, String password)

{

*//*

**try**

{

**if**(name.length() <= 16 && password.length() <= 16) {

**while** (name.length() < 16)

name += String.*valueOf*(**'\0'**);

**while** (password.length() < 16)

password += String.*valueOf*(**'\0'**);

*encode*(name+password, Type.***CREAT\_ACC***, 0, 0, 0);

}

**else** *warning*(**"Name and Password must be MAX 16 characters!"**);

}

**catch**(Exception e)

{

*//*

}

}

* 1. Log-in

**public static void** login(String name, String password)

{

*//*

**try**

{

**if**(name.length() <= 16 && password.length() <= 16) {

*user* = **new** User(-1, name, password);

**while**(name.length() < 16)

name += String.*valueOf*(**'\0'**);

**while**(password.length() < 16)

password += String.*valueOf*(**'\0'**);

*encode*(name + password, Type.***LOG\_M***, Code.***L\_IN***, 0, 0);

}

**else** *warning*(**"Name and Password must be MAX 16 characters!"**);

}

**catch**(Exception e)

{

*//*

}

}

**private static void** login(**int** to)

{

Log.*d*(**"login - to"**, String.*valueOf*(to));

*user*.setUserID(to);

*user*.setStatus(1);

*contact*(Code.***F\_GET\_PENDING***, *user*.getUserID(), 0, **""**);

*state*.startActivity(**new** Intent(*state*, Contacts.**class**));

}

* 1. Message Send

**public static void** sendMessage(String s)

{

**try**

{

*encode*(s, Type.***TEXT\_M***, Code.***TEXT\_C***, *user*.getUserID(), *contact*.getUserID());

*chat*.append(*user*.getUserName() + **": "** + s + **"\n"**);

}

**catch**(Exception e)

{

}

}

**public static void** encode(String s, **int** type, **int** code, **int** from, **int** to)

{

String x;

**switch**(String.*valueOf*(s.length()).length())

{

**case** 1: x = **" "** + String.*valueOf*(s.length()); **break**;

**case** 2: x = **" "** + String.*valueOf*(s.length()); **break**;

**case** 3: x = **" "** + String.*valueOf*(s.length()); **break**;

**case** 4: x = String.*valueOf*(s.length()); **break**;

**default**: x = **" 0"**; **break**;

}

x += **new** String(ByteBuffer.*allocate*(4).order(ByteOrder.***LITTLE\_ENDIAN***).putInt(type).array(), Charset.*forName*(**"UTF-8"**));

x += **new** String(ByteBuffer.*allocate*(4).order(ByteOrder.***LITTLE\_ENDIAN***).putInt(code).array(), Charset.*forName*(**"UTF-8"**));

x += **new** String(ByteBuffer.*allocate*(4).order(ByteOrder.***LITTLE\_ENDIAN***).putInt(from).array(), Charset.*forName*(**"UTF-8"**));

x += **new** String(ByteBuffer.*allocate*(4).order(ByteOrder.***LITTLE\_ENDIAN***).putInt(to).array(), Charset.*forName*(**"UTF-8"**));

x += s;

Log.*d*(**"Encode - x"**, x);

*SM*.setMessage(x);

Log.*d*(**"Encode"**, **"Past setMessage"**);

}

1. Desktop Client Login  
     
   //include the header files for login from the GUI

#include "login.h"

#include "ui\_login.h"

Login::Login(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::Login)

{

ui->setupUi(this);

//to show image in the Login GUI

QPixmap pix("/media/shreyas/DATA/Photography/Shortlisted/bean.jpg");

ui->label\_image->setPixmap(pix);

//to check if connection to the Database is established or not

if(!connOpen())

ui->label->setText("Failed to open the Database");

else

ui->label->setText("Connected to Database");

}

Login::~*Login*()

{

delete ui;

}

QString username, password, status, userEmail;

/\*Put it outside the function to make it accessible by other function\*/

void Login::on\_pushButton\_clicked()

{

//Use the value from the textbox and store into variables for later use

username = ui->lineEdit\_Username->text();

password = ui->lineEdit\_Password->text();

if(!connOpen())

{

qDebug()<<"Failed to Open the Database";

return;

}

//Verify the user's password from the Database named login and open connection

connOpen();

QSqlQuery qry; //defined to initialize SQL Query in QT Creator

qry.prepare("SELECT \* from login WHERE Name='"+username +"' and Password='"+password+"'");

if(qry.exec())

{

int count=0;

while(qry.next())

{count++;}

if(count==1){

ui->label->setText("Username and password is correct");

connClose();

this->hide();

MyAccount myaccount;

myaccount.setModal(true);

myaccount.*exec*();

}

if(count>1)

ui->label->setText("Duplicate Username and password");

if(count<1)

ui->label->setText("INCORRECT Username and password!");

}

//use the user's status update in my account

connOpen();

QSqlQuery qry1;

qry1.prepare("SELECT Status from login WHERE Name='"+username);

}

1. Desktop Client My Account

/\* include my account header files from the GUI \*/

#include "myaccount.h"

#include "ui\_myaccount.h"

#include <login.h>

/\* To use MessageBox functions from the default library \*/

#include <QMessageBox>

/\*To open the web page\*/

#include <QDesktopServices>

#include <QUrl>

/\*Digital Clock\*/

#include<QTimer>

#include<QDateTime>

#include<QDate>

extern QString username, password, status, userEmail;

/\*used as Global variable from the login window to be used in my account window\*/

MyAccount::MyAccount(QWidget \*parent) :

QDialog(parent),

ui(new Ui::MyAccount)

{

/\*To show the current Time and Date\*/

ui->setupUi(this);

QTimer \*timer=new QTimer(this);

connect(timer, SIGNAL(timeout()), this, SLOT(showTime()));

timer->start();

QDate dateTime = QDate::currentDate();

QString dateTime\_text = dateTime.toString();

ui->label\_time->setText(dateTime\_text);

/\*Check the Database connection\*/

Login conn;

if(!conn.connOpen())

ui->label\_sec\_status->setText("Failed to open the Database");

else

ui->label\_sec\_status->setText("Connected to Database");

//Custom Name for each logged in user

ui->label\_username->setText("Welcome " + username + userEmail);

}

//Show current time in HH:MM format

void MyAccount::showTime()

{

QTime time = QTime::currentTime();

QString time\_text=time.toString("hh : mm");

/\* tick timer every second \*/

if ((time.second() % 2) == 0)

{

time\_text[3] = ' ';

}

ui->label\_clock->setText(time\_text);

}

MyAccount::~*MyAccount*()

{

delete ui;

}

//Display Status

void MyAccount::on\_pushButton\_clicked()

{

Login conn;

QString status;

//username = ui->lineEdit\_Username->text();

status = ui->lineEdit\_status->text();

if(!conn.connOpen())

{

qDebug()<<"Failed to Open the Database";

return;

}

conn.connOpen();

QSqlQuery qry;

//Use the status from the DB into the textbox

qry.prepare("UPDATE login set Status='"+status+"' WHERE Name='"+username +"' ");

if(status!=NULL)

{

if(qry.exec()) //if valid status, then save the updated status and display message

{

QMessageBox::critical(this,tr("Save"),tr("Changes saved!"));

conn.connClose();

}

else

{

QMessageBox::critical(this,tr("Error"),qry.lastError().text());

}

}

else //if status not entered and clicked update status

{

QMessageBox::critical(this,tr("Error"),tr("Enter a valid status"));

}

}

//Sign out from the app

void MyAccount::on\_button\_signOut\_clicked()

{

//this->hide();

this->close();

}

//When Contact button is clicked, open the website in Default Browser

void MyAccount::on\_pushButton\_dev\_clicked()

{

QString link = "http://www.beta.shreyasg.com";

QDesktopServices::openUrl(QUrl(link));

}

//Update user password into the Database

void MyAccount::on\_pushButton\_password\_clicked()

{

Login conn;

QString password;

//username = ui->lineEdit\_Username->text();

password = ui->lineEdit\_password->text();

if(!conn.connOpen())

{

qDebug()<<"Failed to Open the Database";

return;

}

conn.connOpen();

QSqlQuery qry;

//qry.prepare("select into login (Password) values ('"+status+"')WHERE Name='"+username +"' ");

qry.prepare("UPDATE login set Password='"+password+"' WHERE Name='"+username +"' ");

/\* Entered value in the field \*/

if(password!=NULL)

{

if(qry.exec())

{

QMessageBox::information(this,tr("Save"),tr("Password successfully changed"));

conn.connClose();

}

else //Show the exact error

{

QMessageBox::critical(this,tr("Error"),qry.lastError().text());

}

}

/\* For NULL password \*/

else

{

QMessageBox::critical(this,tr("Error"),tr("Enter value in password"));

}

}

1. Desktop Client main.cpp

#include "login.h"

#include <QApplication>

#include<QSplashScreen>

#include<QTimer>

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

QSplashScreen \*splash = new QSplashScreen;

splash->setPixmap(QPixmap("/media/shreyas/DATA/Photography/Shortlisted/splashscreen.png"));

splash->show();

Login w;

//Splash Screen will appear for 2.5sec (2500msec) and then start the application

QTimer::singleShot(2500,splash,SLOT(close()));

QTimer::singleShot(2500,&w,SLOT(show()));

return a.exec();

}