A UPnP BASED PHOTO DISPLAY SYSTEM

A Project

Presented to the faculty of the Department of Computer Science
California State University, Sacramento

Submitted in partial satisfaction of the requirements for the degree of

MASTER OF SCIENCE

in

Computer Science

by

Ashwini Suresh

FALL
2016
A UPnP BASED PHOTO DISPLAY SYSTEM

A Project

by

Ashwini Suresh

Approved by:

__________________________________, Committee Chair
Dr. Jinsong Ouyang

__________________________________, Second Reader
Dr. Jun Dai

____________________________
Date

iii
Student: Ashwini Suresh

I certify that this student has met the requirements for format contained in the University format manual, and that this project is suitable for shelving in the Library and credit is to be awarded for the project.

__________________________, Graduate Coordinator

Dr. Ying Jin

Date

Department of Computer Science
Abstract

of

A UPnP BASED PHOTO DISPLAY SYSTEM

by

Ashwini Suresh

Nowadays there are multiple data storage options available on the internet to store personal media. And to share the media within the home networks is possible through UPnP (Universal Plug and Play)/DLNA (Digital Living Network Alliance) enabled devices. One such media is images. We store photos on cloud, share it on the social media and so on. The number of photos stored are so large that we might not revisit them once stored. Thus viewing them on a display device that could dynamically show the photos will be helpful. Thus, an android application is developed to render the photos from the various sources like mobile devices and social media applications onto a photo displaying device. The features of the photo display system include - displaying photos from smart phone, social media – Facebook and controlling the display device from the android application.

_______________________, Committee Chair
Dr. Jinsong Ouyang

_______________________
Date

v
ACKNOWLEDGEMENTS

I hereby convey gratitude to Dr. Jinsong Ouyang for his guidance, motivation throughout the duration of the project. And my sincere thank you to my second advisor Dr. Jun Dai for his inputs for the improvement of the project. I would also like to sincerely thank the faculty of Department of Computer Science at California State University, Sacramento who have guided me through the Master’s program. Last but not least, I would like to thank my parents for their support throughout my education. And, my sincere thank you to my dearest husband who motivated me and supported me throughout the Master’s program.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Acknowledgements</th>
<th>List of Figures</th>
</tr>
</thead>
<tbody>
<tr>
<td>vi</td>
<td>ix</td>
</tr>
</tbody>
</table>

## Chapter

1. **INTRODUCTION** ........................................................................................................ 1

2. **PROJECT REQUIREMENTS** .......................................................................................... 2
   2.1 User Options ........................................................................................................... 2
   2.2 Display Device ......................................................................................................... 5

3. **ANDROID DEVELOPMENT BASICS** ............................................................................. 6
   3.1 Application Components ......................................................................................... 6
      3.1.1 Activities ......................................................................................................... 6
      3.1.2 Services .......................................................................................................... 9
      3.1.3 Content Providers ......................................................................................... 10
      3.1.4 Broadcast Receivers ...................................................................................... 11
      3.1.5 Fragments ...................................................................................................... 11
   3.2 Setup Android Environment .................................................................................... 12
   3.3 Create Android Application .................................................................................... 13
   3.4 Launch Android Application ................................................................................... 14
      3.4.1 Launch App On Emulator ............................................................................... 16
      3.4.2 Launch App On Device .................................................................................. 16
4. UPnP BASICS ......................................................................................... 18
5. FACEBOOK INTEGRATION BASICS ....................................................... 20
6. IMPLEMENTATION .................................................................................. 23
   6.1 Android Application ......................................................................... 23
      6.1.1 Application Launch Page .......................................................... 23
      6.1.2 Image Source Selector Page ...................................................... 27
      6.1.3 Facebook Login Page ................................................................. 29
      6.1.4 Facebook Albums Page ............................................................... 31
      6.1.5 Facebook Pictures Page ............................................................. 33
   6.2 Java Desktop Application ..................................................................... 35
      6.2.1 Photo Frame ............................................................................... 36
      6.2.2 Photo Frame Screen ................................................................... 39
      6.2.3 Display Device Server ................................................................. 42
      6.2.4 Facebook Operations ................................................................. 43
   6.3 Future Work ...................................................................................... 44
7. CONCLUSION .......................................................................................... 45
References .................................................................................................. 46
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figures</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Use Case Diagram</td>
<td>2</td>
</tr>
<tr>
<td>2.</td>
<td>Choose Device Flow Diagram</td>
<td>3</td>
</tr>
<tr>
<td>3.</td>
<td>Select Sources Flow Diagram</td>
<td>4</td>
</tr>
<tr>
<td>4.</td>
<td>Activity Life Cycle</td>
<td>8</td>
</tr>
<tr>
<td>5.</td>
<td>Life Cycle of Fragment</td>
<td>13</td>
</tr>
<tr>
<td>6.</td>
<td>New Android Application Wizard</td>
<td>15</td>
</tr>
<tr>
<td>7.</td>
<td>Activity Name Window</td>
<td>15</td>
</tr>
<tr>
<td>8.</td>
<td>AVD Manager</td>
<td>17</td>
</tr>
<tr>
<td>9.</td>
<td>Add New App on Facebook</td>
<td>21</td>
</tr>
<tr>
<td>10.</td>
<td>Android Section</td>
<td>22</td>
</tr>
<tr>
<td>11.</td>
<td>Application Launch Page</td>
<td>24</td>
</tr>
<tr>
<td>12.</td>
<td>Image Source Selector Page</td>
<td>27</td>
</tr>
<tr>
<td>13.</td>
<td>Select Picture App Page</td>
<td>28</td>
</tr>
<tr>
<td>14.</td>
<td>Delete Source from Playlist</td>
<td>29</td>
</tr>
<tr>
<td>15.</td>
<td>Facebook Login Page</td>
<td>30</td>
</tr>
<tr>
<td>16.</td>
<td>Facebook Albums Page</td>
<td>32</td>
</tr>
<tr>
<td>17.</td>
<td>Facebook Pictures Page</td>
<td>36</td>
</tr>
</tbody>
</table>
Chapter 1

INTRODUCTION

The Photo Display android application is a UPnP control point application, which allows the android user to choose the display devices within the home network to display images as slideshow and to control the slideshow with the help of control buttons on the android screens. The application thus provides a convenient way to revisit the old memories without the user having to click on each image manually.

The major features of the android app include detecting the UPnP devices within the network, integration with Facebook website, control buttons to control the display device. In addition, the application also acts as a media server for the images. This means that the display device can also render the images from the android mobile device the Photo Display App is running on. In order for the user to use the Facebook feature the user must have Facebook account.
Chapter 2
PROJECT REQUIREMENTS

2.1 User Options

To interact with the android application, the application provides the following options to the user. The Figure 1 shows the use case diagram.

- Choose the display device
- Select or de-select the sources for the image slideshow
- Play, Pause, Next and Previous buttons to control the slideshow
- Control the speed of the slideshow

![Figure 1: Use Case Diagram](image-url)
Choose the display device

When the user launches the application, the screen with the list of the devices that are present within the home network are displayed. There will be a green checked symbol next to the device name if the user had selected this device the last time the application was opened. If the user is opening the application for the first time, none of the devices are checked. The user can only select one device at a time. The devices that are displayed on the screen will be capable of displaying images from Facebook and from the device the android application is running on. The Figure 2 shows the flow diagram.

![Flow Diagram](image)

Figure 2: Choose Device Flow Diagram

Select or de-select the sources for the image slideshow

The application provides two sources – Facebook and Local Device – to choose from in order to display the images onto the display device. The user has an option of selecting one or both of the sources. By clicking on Local Device, the user will choose the photos stored on the android device whereas when clicked on Facebook, user’s albums list is displayed. The flow diagram for selecting the sources is shown in Figure 3.

When the user chooses Local Device, he/she is provided with an option of choosing the images from the Local Device. This means that the images that are stored on the android phone can be selected to be added to the slideshow playlist. Images can be selected either
from Gallery app or Photos app (android applications which can display the images which are stored on the device).

When the user chooses Facebook, he/she is provided with an option of choosing the albums that the user has uploaded onto their Facebook account. If the user is launching the application for the first time, the application requires the user to Login to their Facebook account. Once the login process is successful, the list of user’s albums is displayed. The user can then browse through the pictures under these albums and select the album to add it to the slideshow playlist.

In addition, the user can also deselect the source from the list. If deselect is chosen, then all the pictures that belong to that source will not be displayed on the display device.

Figure 3: Select Sources Flow Diagram
Play, Pause, Next and Previous buttons to control the slideshow

Similar to the control options for playing the music from the playlist, the Photo Display app provides the user with control options to control the image slideshow. While the images are being displayed on the display device, the user will have the option to control the slideshow by clicking on Play, Pause, Next and Previous buttons on the screen. The slideshow will be paused after displaying the respective image, if the user clicks on Next or Previous buttons. Then the next or previous images can be viewed using the respective buttons. If the user chooses to resume the automatic slideshow, then the user can then manually click on the Play button.

Control the speed of the slideshow

While the slideshow is playing, the user also has an option of controlling the speed of the slideshow. The options include 1 to 5 seconds. This is the time interval between the images in the slideshow.

2.2 Display Device

In order to display the images from internet sources and from the local device, a device which has a capability of rendering images using URL is required. Also, the device must have a display screen to display the rendered images. The device must also be capable of announcing its presence within the home network and other devices should be able to control the slideshow.
Chapter 3

ANDROID DEVELOPMENT BASICS

Android is an operating system developed using Java programming language and the open source operating system is based on Linux which runs on devices like tablets and smart mobile phones [1]. The developed applications for android operating system are packaged into .apk file which can be installed on the device which is running on android operating system. The applications can be installed either from Google Play or from the computer by downloading .apk files. Google Play is a store of android apps, where all the apps are reviewed and then published to the users. Whereas, .apk files downloaded from Internet may not be reviewed. For development and testing purposes the .apk files can be installed on the device from the computer.

3.1 Application Components

Each android application contains Manifest file which is a configuration file that contains information about other components of application. The following are application components that can be incorporated into the android application. Based on the requirement of the project one or more of these components can be used.

3.1.1 Activities

Every android application requires some form of user interface and this is catered by Activities. Each activity has a pair of XML file and a java file and the Activity names are mentioned in the Application’s manifest.xml file. The .MainActivity is the first activity
that is launched and the page associated with it is displayed when the application is opened by the user. Any activity can be assigned as the first activity to be displayed by using android.intent.category.LAUNCHER in the <category> tag under the activity tag in the manifest file.

The layout of a page is mentioned in the XML file and all the necessary logic behind handling the user events on the screen are written in the java file. For example, an activity named Main has the layout XML file activity_main.xml and has the java file named MainActivity.java. Also, any activity can be launched from other sources other than user launching an application. One such method is by using intent filter. Developers can mention if the activity can be started from other applications by mentioning the <intent-filter> tag under the <activity> tag in the manifest file.

In addition to the Main activity, the application can have many more activities. This also means that only one activity is shown to the user at a time. Hence, an activity goes through different life cycle stages during the application session and is shown in the Figure 4. The figure shows that when an activity is in a particular stage that means that the method associated with that stage will be executed. Hence, the developers need to override these methods to obtain the necessary results.
**onCreate** - This method is invoked when the activity is created. Hence, all the static binding of the variables and initial setup of the activity can be included in this method.

**onStart** - The method is invoked when the screen associated with the activity is about to become visible to the user.

**onResume** - This method is called when the user starts to interact with the page. For example, by clicking on the buttons or filling the information.
• onPause - This method is called just before resuming the another activity. This means that the user will not interact with this activity until resumed. Hence, saving the data, and closing up the connections that are not required anymore can be written under this method.

• onStop - This method is called when the activity is completely not visible to the user either because of new activity covering the old one or the current activity is being destroyed.

• onDestroy - This method is called when the activity is being destroyed. Unbinding the service connections and closing any other open connections can be performed in this method. The activity will be destroyed when the activity calls finish() method or when the instance of this activity is being destroyed to save space.

• onRestart - This method is called just before starting the activity after being stopped.

3.1.2 Services

Service is a long running application component which runs on the background and they do not interact with user. The main uses of Services include time consuming operations like Network operations, File I/O, Image and Video streaming. Once the Service is started, it is alive throughout the life of the application. Hence, any activity can start or bind with the Service and use its features. The behavior of Service is little different when activities bind to the Service when compared to starting them with startService method.
Multiple activities can bind to the Service and only when there are no more activities bound, the Service will be destroyed. The following methods are to be overridden to implement a Service.

- **onStartCommand** – This method is called when an activity calls startService method to start the service. Once Service is started, the developer must stop the service using stopService method or else the Service will run indefinitely.

- **onBind** – This method is called when the activity binds to the Service using bindService method.

- **onCreate** – This method is called when the Service is created and just before executing onStartCommand or onBind method. The method can be used for the initial setup of the Service.

- **onDestroy** – This method is called when the Service is being destroyed when no other activities are bound to it.

In order for activities to either start a service or bind to it, the Service class has to be mentioned in the manifest.xml file under the <application> tag with the template `<service android:name = “.ServiceClassName”/>`.

### 3.1.3 Content Providers

Content Providers as the name suggests acts as database meant for managing the data and providing them to applications. This component can be used to share the data between applications. The data is either stored in SQLite databases or on the File system of the
device. The applications can access the data using query which is of the form
content://<authority>/<data type>/<id> where,

- **Authority** – represents the name of the content provider.
- **Data type** – represents type of the data content provider supports
- **Id** – represents identification of the object

### 3.1.4 Broadcast Receivers

The Broadcast Receivers are the components that are used to handle the messages like events or intents that are broadcasted either by other applications or by the system. The component needs to be created and registered to handle the messages broadcasted by the system.

### 3.1.5 Fragments

Fragments are the sub components of the activities. Fragment has a separate layout XML and java file just like the Activity XML file. Hence, the Fragments can be reused in many activities and can be dynamically added or removed from the activity while the activity is still alive. To add a fragment to the activity, `<fragment>` tag is to be added under the `<activity>` tag in the activity’s layout XML file. The lifecycle of the Fragment depends on the activity it is attached to and is shown in the Figure 5. To create a fragment at least the following methods needs to be implemented.
onCreate – This method is called when the fragment is created. Initializations of
the variables, or initial setup of the fragment views can be written under this
method.

onCreateView – This method is created just before the user interface for the
fragment is rendered.

onPause - This method is called when the fragment is either removed or replaced
from the activity.

3.2 Setup Android Environment

Android development can be done using various development environments like Eclipse,
Android Studio. For this project, Eclipse is used for the app development. Before starting
the development, Eclipse needs to be prepped with the Android Development Tools
(ADT). All the necessary tools and plugins are available in the Eclipse marketplace.

Once ADT plugin is installed, start the android sdk manager to install additional tools.
Multiple versions of the android sdk can also be installed on Eclipse using sdk manager.
But for every application created, the developer needs to mention the sdk version.
3.3 Create Android Application

To create a new android application, click on “New” on toolbar and select “Android Application Project”. This displays android project wizard as shown in the Figure 6. In this screen, application name needs to be filled and the minimum required sdk, target sdk and sdk version of android to compile the android app needs to be chosen. Clicking the “Next” button will show the Configure screen, where project properties like allowing the
project for custom launcher icon and choose the workspace location is done. After completing the configure screen, if the custom launcher icon is checked then the Configure Launcher Icon screen appears to choose the launcher icon for the android app. Or else if the create activity in the Configure screen is checked, Create Activity screen appears where a template for the activity can be chosen. At the last stage after choosing the template for the activity, a screen to enter the activity name appears and is as shown in the Figure 7. Once the finish button is clicked, a new android project with the manifest file and many other default resource files are created.

3.4 Launch Android Application

Android applications are mainly launched on the mobile device. But for the purpose of testing the applications they can also be launched on the emulators. Android Developments Tools (ADT) also provide various default emulator device configurations to test the applications and new device configurations can be found by starting sdk manager.
Figure 6: New Android Application Wizard

Figure 7: Activity Name Window
3.4.1 Launch App on Emulator

In addition to the emulator provided by the ADT, there are other third party emulators available. For example, Genymotion. Yet the process of launching the application is same for all the emulators.

The first step to launch is to create and start the emulator. There are many configurations available for emulator, and these configurations are handled by Android Virtual Device (AVD) manager. Click on AVD manager in the toolbar, which opens the window as shown in the Figure 8. New devices can be created using the default device configurations found under Device Definitions Tab. Choose the device and click on Create Device. A copy of that device with the same configuration will be created and is displayed under Android Virtual Devices tab. To start the emulator, choose the device to start on Android Virtual Devices tab and click start. Once the device is fully started, click on the android project and run the application. This causes the AVD manager to open showing the emulator device being On. Choose the device and the app will be installed and launched.

3.4.2 Launch App on Device

An android mobile device which is connected to PC via USB cable is required to launch the android app. The mobile device need to be configured for debugging on PC. This can be done by opening the Settings app on the android device and click on Developer Options. The list of developer options will be displayed among which USB debugging
option is present. The USB debugging must be on for the application to be installed from Eclipse IDE. Once the device is configured for USB debugging, click on the android project to be installed on the device displayed under the project explorer tab and click run as android application. If the application is installed for the first time on the device, (Android Virtual Device) AVD manager is opened that displays the devices that the application can be launched on. The AVD manager also displays the connected device. Select the connected device and the app will be installed and launched on the device. To view the logging messages from the device, LogCat window can be used.

Figure 8: AVD Manager
Chapter 4

UPnP BASICS

Universal Plug and Play (UPnP) is a combination of existing standard network protocols which are designed for discovery of networked devices and their services within the home network and for easy communication and data sharing between them. The protocol is designed such that it is easy for the networking devices to detect and connect to other devices without device specific configurations. The protocol also helps the devices to gracefully leave the network [4]. The devices like audio speakers, personal computers, smart TV, phones, tablets and so on can be part of the UPnP network which can stream data from one device to another.

UPnP enabled network consists of components like UPnP enabled devices that provides certain services. The service can be used by providing the commands via control points like remote or smartphone apps. In order for a control point to find a device that provides a service, service template provided by device is looked for within the home network and this phase is called discovery. The content to be rendered can be present locally in the network (like phone) or in the cloud storage (like iCloud) and the component is considered as Media Server. The component that is capable of rendering the data from the Media Server is called as Media Renderer or Media Player. The UPnP device can be a group of embedded devices where each embedded device can provide a unique service. In addition, the device can be a group of components, for example a device can be a combination of Media Player and Control point like a mobile device. Another important
aspect of the UPnP protocol is the event notification system known as General Event Notification Architecture (GENA), which allows the device under control to send changes in their state variables to the controlling device.

There are several standardized specifications available for the type of components the home network can have and is maintained by Open Connectivity Foundation and the information is found online “at [5]”. Using these specifications, developers can implement the devices and their services that follow UPnP standards. Furthermore, third party libraries are available that provide APIs for using the UPnP features. To implement the UPnP features in the Photo Display App Cling library is used.

Cling library is an open source library implemented for the purpose of easy usage of UPnP protocol [6]. The java library supports both Java and Android development. There are four main modules offered – Cling Core, Cling Support, Cling Workbench and Cling MediaRenderer. The Photo Display app mainly uses Cling Core module. Currently the latest version of Cling requires Android version to be at least 15. The APIs used in the project are explained in detail under chapter 6 - Implementation.
Chapter 5
FACEBOOK INTEGRATION BASICS

The Photo Display app requires accessing the Facebook user’s data and is achieved with the help of Facebook Api. To use some of the features like Facebook login, graph requests the android application needs to be registered with the Facebook website [7]. The following sequence of steps are required for using Facebook features in the android app.

1) It is important for the android device for which we are developing the application to have Facebook app installed on it. For Facebook login feature or for use of Facebook login/logout button to work with the new android app, the android device needs to have Facebook app installed so that the Facebook sdk can interact with. The Facebook app can be installed from Google Play – a marketplace for android applications or from the Facebook website.

2) To integrate with Facebook and to publish the android app to the users, the developer of the application needs to have a Facebook developer account. The account can be obtained by clicking the Create Developer Account button [7].

3) Download Facebook sdk for android. The latest version of the sdk is available “in [8]”.

4) Obtain the app Id for the android application. Login to the developer account and on the menu click on Add a new App. Enter the name and contact information for the app.
5) Once the app name is registered, click on the app name under the menu which shows a screen as shown in the Figure 9. Click on Settings -> Basic -> Add Platform and select Android which adds an Android section to the app and is shown in Figure 10.

6) Enter the package name of the android application and the name of the activity where the Facebook sdk is initialized. The key hash needs to be entered, which is used to compare with the actual graph requests made by the android application.

![Figure 9: Add New App on Facebook](image)
Figure 10: Android Section
Chapter 6

IMPLEMENTATION

The UPnP related functionalities in the android application and the display device are implemented with the help of Cling Library. This is an open source library that implements UPnP stack and few of the functionalities used in this project are

- Creating a UPnP device
- Search for devices within the home network
- GENA event subscription
- Execute control actions on the UPnP device

The detailed usage of these features are explained in the subsequent sections.

6.1 Android Application

6.1.1 Application Launch Page

This is the first screen displayed to the user when the android application is launched for the first time or when the application is reopened. The page displays the list of display devices capable of providing the display service and capable of rendering the images from Facebook and from the Device the android application is running on. Figure 11 shows the Launch Page.

The Launch Page is like the Home Screen of the application where all the major controls of the slideshow and choosing of the source and devices happen. Hence, the activity associated with this page is the main activity.
The main activity is associated with the xml file activity_main.xml that contains the layout configuration of the launch page. The layout for this page includes Navigation bar at the bottom of the screen and fragments for Devices List View, control buttons for the slideshow. The XML tags for the respective views are mentioned under the Linear Layout tag.

The layout for the Home screen is designed in such a way that, Devices List View page and the list of sources (Facebook and Local device) page can be easily switched without creating new activities when the buttons on Navigation bar is clicked. This kind of tabbed view is achieved with the help of frame layout that can contain the fragment. A fragment
for Devices list view and Album Source list view are created separately. Each of the fragment could be included in the frame layout and displayed one at a time. This can be achieved with the help of fragment manager and fragment transaction.

In order for the android application to use the UPnP feature Cling library is used. A service named “AndroidUpnpServiceImpl” is added in the manifest file under the application tag. The activities within the android application can then bind or unbind to the Service to use their functionalities. In order to display the devices that are present in the home network a registry listener is created that tracks the devices that are added or removed in the network. The following code is executed when the remote device is added to the network. The device is only added if the device provides the service type DisplayPhoto.

```java
if (remotedevice.findService(new UDAServiceId("DisplayPhoto")) != null &&
    deviceListAdapter.getPosition(remotedevice.getDetails().getFriendlyName().toString()) < 0) {
    listDevices.add
        (remotedevice.getDetails().getFriendlyName().toString());
    deviceListAdapter.notifyDataSetChanged();
}
```

Once the user clicks on the device from the list, this means that the device is chosen and going forward, all the control actions are performed on this device. The control actions available in the launch page are Play, Pause, Prev and Next to control the slideshow. In
addition, a Seekbar is used to control the speed of the slideshow. These actions are achieved by calling the respective method on the chosen display device.

There are scenarios where the Slideshow is paused automatically when the user clicks on Next or Previous button. Hence, the android app needs to be updated with the change in status of Play state variable of the display device. This is implemented by using GENA event notification system. An event subscription callback method is registered that implements various types of events received from the display device. The following code shows the event subscription method `eventReceived`, where the activity subscribes for changes in state variables that are related to the display Service type and the events are refreshed for every 100 seconds.

```java
ImageView playButton = (ImageView) findViewById(R.id.play);
ImageView pauseButton = (ImageView) findViewById(R.id.pause);
Bundle b = new Bundle();
if(Integer.parseInt(arg0.getCurrentValues().get("Play").toString()) == 0)
{
    b.putBoolean("playpause", false);
    playpausemsg.setData(b);
    mHandler.sendMessage(playpausemsg);
}
else
{
    b.putBoolean("playpause", true);
    playpausemsg.setData(b);
    mHandler.sendMessage(playpausemsg);
}
```
6.1.2 Image Source Selector Page

As mentioned in the previous section, this page is same as the launch page but with a different ListView that contains the sources (Facebook and Local Device) and is shown in Figure 12. This toggle screen is achieved by adding the ListView as Fragments.

When the user clicks on Local Device, this means that the images from the device can be chosen to be displayed onto the display device. The Figure 13 shows the applications which allows user to browse through images and pick them. This is achieved with the help of implicit intents. The applications which are present on the phone that can cater to the actions requested by the intent are displayed on the screen. So, an implicit intent with action Intent.ACTION_PICK which acts on the External content of the device is used.
The option EXTRA_ALLOW_MULTIPLE is used to allow the user to pick multiple images from the app (provided the app supports this feature).

Once the image is selected either from the Gallery app or Photos app, the results are available in the onActivityResult method. The result includes the URI of the image selected which can be retrieved using intent’s getData or getClipData method. When the user clicks on Facebook, activity FacebookAlbums related to the Facebook operations is started by using Intent instance and startActivityForResult method.

Figure 13: Select Picture App Page
The page also provides the option of deselecting the source which causes the images not to be displayed from that source. The Figure 14 shows the alert dialog to confirm the removal of source from Playlist.

![Figure 14: Delete Source from Playlist](image)

### 6.1.3 Facebook Login Page

The login page for Facebook is the starting point for all of the other interactions the application is going to make with the Facebook. The screenshot of the page is shown in Figure 15. The layout associated with this page contains widget LoginButton provided by the Facebook sdk. This widget can be obtained from com.facebook.login.widget.

In order for my application to access the Facebook user’s images, the application requires access token with the read permissions. Hence, the successful login of the Facebook user
is required in order to obtain the access token. The successful login means that the Facebook user has granted the application to access his/her photos that are uploaded on the Facebook website. All the interactions with the Facebook is possible by using the sdk provided by Facebook for android applications. And, in order for the Facebook LoginButton to show on the screen, the Facebook sdk needs to be initialized using FacebookSdk.sdkInitialize() method before setting the content of the view for fragment or activity.

Figure 15: Facebook Login Page
The LoginButton of Facebook sdk handles the login and logout process. This means that when the Login button is clicked, the Facebook activity that belongs to the Facebook application which is installed on the phone is shown to the user. Once the user enters their username and password and clicks login, the Facebook activity handles the login and returns the result back to our application. In the midst of this process to get all the necessary permissions (for example user_photos) for the future requests the application is going to make to the Facebook website, the permissions needs to be set using setReadPermissions() method of Facebook login button class instance. To obtain the access token once the login is successful, a call back method needs to be registered for the login button and this call back is invoked in onActivityResult method. The activity related to the Facebook Albums page will be started by using startActivity method. This logic is mentioned in onSuccess method which is executed when registered callback manager invokes the Facebook callback method upon successful login.

6.1.4 Facebook Albums Page

The Figure 16 shows the list of albums of the Facebook user. The layout associated with this page has list of ImageView that can contain the cover picture of the album. To access the user’s data on Facebook, the activity needs to use the access token to request for data. Once the login process is successful in the Facebook login page the application is granted an access token and can then be accessed in all the other activities within the application. This is achieved by using getCurrentAccessToken method of AccessToken class.
The picture of the user can be obtained by creating a Facebook graph request using `newMeRequest` method with the access token assigned for the application from Facebook website. The following code shows how to set the parameters for the request.

```java
Bundle parameters = new Bundle();
parameters.putString("fields","albums.fields(id,name,cover_photo.fields(images))");
requestForImage.setParameters(parameters);
```

Figure 16: Facebook Albums Page

On selecting the album, a new page that allows the user to browse through all the photos under this album is shown to the user. The `onActivityResult` of this activity is invoked
when the control from Pictures page (mentioned in the next section) is returned. This means that the user either selected the album to be displayed onto the selected display device or he/she did not choose the album. If the user chose the album, the respective album ID is sent to the display device by invoking DisplayFacebookPictures action. The reference to action can be obtained by using service’s getAction method.

6.1.5 Facebook Pictures Page

As shown in the Figure 17, the layout for the activity is list of ImageView to display the pictures under the selected album and a Select button. To obtain the pictures for the album, a Facebook graph request is invoked by setting the albumID/photos as the graph request path with input parameters “id” and “images”. The path is set using graphrequest’s setGraphPath method and the input parameters are mentioned using the Bundle class with key “fields”. The “/photos” represents all the photos under the albumID.

The JSON object obtained from the response contains the URL of the image, hence an http request needs to be made to retrieve the image bitmap. The logic to load all the images of the album to the screen is shown by the following code.

```java
public void loadimage(JSONObject object)
{
    try
    {
        String picUrl = null;
        JSONArray imageUrlArray;
        int height;
        int width;
        JSONArray pictureArray = object.getJSONArray("data");
```
pictureUrl = new URL[pictureArray.length()];
pictureId = new String[pictureArray.length()];

for (int i =0; i< pictureArray.length(); i++)
{
    obj = (JSONObject)pictureArray.get(i);
    imageUrlArray = obj.getJSONArray("images");

    for(int j=0;j<imageUrlArray.length();j++)
    {
        picUrl = ((JSONObject)imageUrlArray.get(j)).getString("source");
        height = ((JSONObject)imageUrlArray.get(j)).getInt("height");
        width = ((JSONObject)imageUrlArray.get(j)).getInt("width");

            if(height > 100 & height < 400 )
            {
                break;
            }
    }

    pictureUrl[i] = new URL(picUrl);
    pictureId[i] = obj.getString("id");
}

final URL[] url = pictureUrl;

new Thread()
{
    @Override
    public void run()
    {
        // TODO Auto-generated method stub

        Message msg;
        Bitmap bitmap;
        Bundle b = new Bundle();

        try
        {
            for(int i = 0; i< url.length ;i++)
            {
                HttpURLConnection conn = (HttpURLConnection)
            url[i].openConnection();
            if(conn == null)
            {
                Log.e("BrowseFacebookPictures:loadimage", "conn with "+ url[i] +"is null ");
            }
        }
}
After browsing through the images, the user can choose to click on Select button. When the select button is clicked, the result which represents that the album is selected and such album is to be displayed onto the display device, is sent back to the Albums page.

### 6.2 Java Desktop Application

The java application is designed to simulate the behavior of a display device capable of rendering images from the internet and from within the home network. The application is
created for the purpose of representing the computer as display device and announce to
the home network that it can provide services of displaying a picture. The application
uses Cling library to implement these features. The application is divided into four major
functionalities and are explained in detail in the following sections.

![Facebook Pictures Page](image)

**Figure 17: Facebook Pictures Page**

### 6.2.1 Photo Frame

The class represents the UPnP device state variables and actions that can be invoked by
control points like Android application. The state variables are annotated with
@UpnpStateVariable and the actions are annotated with @UpnpAction.
The Photo Frame actions available to the control points are

- **Set Play** – to set the Play state of the device
- **Display Next Picture** – to display the next picture in the slideshow
- **Display Prev Picture** – to display the previous picture in the slideshow
- **Display Facebook Pictures** – to stream the images of a user from Facebook website
- **Set paired device IP** – to store the IP address of the paired control point device
- **Set Slideshow Speed** – set the Speed state of the device
- **Get Play** – to send the Play status of the device to control point device
- **Deselect Source** – to deselect either local device or Facebook source

This means that the control device within the network can control the state of the Photo Frame device, by invoking the above actions. The detailed implementation of some of the actions are explained as follows.

The Set Play action is invoked when the control point tries to change the play status of slideshow. The action is implemented using the following code.

```java
public void displayPictures()
{
    PhotoFrame ph = this;
    if(t== null || !t.isAlive())
    {
        t = new Thread()
        {
            public void run()
            {
                p.startDisplayFromFolder(ph);
            }
        };
        t.start();
    }
}
```
To add the Facebook images to the playlist, albumID and accesstoken is required to make the request to Facebook website. So, the following code represents setting the albumId and accesstoken state variables received from the control point.

```java
if(play != newValue && newValue == false )
{
    //turn off the slideshow or pause the show
    //Send event for change in play variable
    play = newValue;

    try
    {
        synchronized (this)
        {
            this.notifyAll();
        }
    }
    catch(Exception e)
    {
        System.out.println("exception in pause" + e);
        e.printStackTrace();
    }
}
else if(play != newValue && newValue == true )
{
    //Send event for change in play variable
    play = newValue;
    if(t == null)
    {
        displayPictures();
    }
    else
    {
        synchronized (this)
        {
            this.notifyAll();
        }
    }
}
6.2.2 Photo Frame Screen

The class represents the JFrame screen that is used to display the images rendered from the android device or from internet. To display the images, the application reads the files from the location where the streamed images are stored. And, for the slideshow to be able to play, pause and show the previous and next images the JFrame thread needs to monitor the state of the respective variables. The following code shows the implementation of the slideshow controls.

```java
if(oldalbumId != albumID)
{
    oldalbumId = albumID;
}

if(accesstoken != token.toString())
{
    accesstoken = token.toString();
}
FacebookOperations fops = new FacebookOperations(token);
fops.retrievePictures(albumID);
if(play == false)
{
    boolean oldvalue = play;
    play = true;
    // sendevent for play variable change
}
displayPictures();
```
public void displayPicsList(PhotoFrame ph, File[] list, String source) {
    Image img;
    Image newImg;
    boolean endLoop = false;
    int i = 0;
    File pictures;
    while (i < list.length) {
        pictures = list[i];
        System.out.println("in while loop i: " + i + "length is " +
        list.length);
        try {
            if (pictures.toString().contains(".jpg") )
        }
        try {
            //read image from pictures
        }
        catch (FileNotFoundException f) {
            System.err.println("File was deleted: " + f);
            continue;
        }
        //scale image to the size of JFrame
        ImageIcon ic = new ImageIcon (newImg);
        if (ph.getPlay() == false && ph.next == false && ph.prev == false) {
            synchronized (ph) {
                System.out.println("pauseDisplay:: in wait");
                ph.wait();
            }
        } else if (ph.next == true) {
            System.out.println("next"+ ":: in true");
            ph.next = false;
        if (source.contains("local device") ) {
            if (ph.deSelectLocalDevice == true) {
                break;
            }
        }
else
{
    if(ph.deSelectFacebook == true)
    {
        break;
    }
}
pics.setIcon(ic);
i++;

synchronized (ph)
{
    System.out.println("nextDisplay:: in wait");
    ph.wait();
}

else if (ph.prev == true)
{
    System.out.println("prev"+ ":: in true");
    ph.prev = false;
    if(i >= 2)
    {
        pictures = list[i-2];
        i = i-2; 
    }
    if(i!=0)
    {
        i--;
    }

    // read from pictures and scale image to JFrame size
    if(source.contains("local device") )
    {
        if(ph.deSelectLocalDevice == true)
        {
            break;
        }
    }
    else
    {
        if(ph.deSelectFacebook == true)
        {
            break;
        }
    }
}
pics.setIcon(imageIcon);
6.2.3 Display Device Server

The class is implemented to handle the UPnP device configurations and to handle the control requests received from control points. Display device required for the project is created with the name “Photo Frame” and the type of device is “Media Renderer” with version 3. Device type is created using cling library class UDADeviceType with Media renderer as parameter which is a standardized device template. And other device specific
information like manufacture details and device specific details can be placed in the discovery messages. This is achieved with the help of ManufacturerDetails and ModelDetails class of the cling library.

6.2.4 Facebook Operations

The class handles all the operations related to interacting with the Facebook website to retrieve the images. To obtain the images access token obtained from the android application is used. The request to retrieve the images from the album must contain “albumId/photos” string which provides the list of url for all the images under the albumId. A Http request is made to stream image from the each url using Rest API and then is stored in a folder location on the display device and is shown by the following code snippet.
Future Work

The application is currently designed to integrate with Facebook website. Many other photo sharing websites can be integrated. In addition, cloud storage options like Amazon cloud where photos can be stored could also be used as source for images.

```java
JSONArray jArraypic = picturefromalbum.getJsonArray("data");
for(int j=0; j< jArraypic.length(); j++)
{
    picId = jArraypic.getJSONObject(j).get("id").toString();
    //facebook request picId/picture with redirect = false option
    picUrl = pic.getJSONObject("data").get("url").toString();
    try
    {
        URI picUri = new URI (picUrl);
        BufferedImage img = ImageIO.read(picUri.toURL());
        System.out.println("img height" + img.getHeight());

        ImageIO.write(img, "jpg", new File(folderlocation + picId+ ".jpg");
    }
    catch(Exception e)
    {
        System.err.println("Exception occured: " + e);
    }
}
```
Chapter 7

CONCLUSION

The Photo Display System will be very useful to view large number of photos without the user having to click on each picture either on the phone or in the photo sharing websites. The Photo Display app provides user friendly features like pairing with the display device easily, choosing only the photos which the user wishes to see frequently, control the image slideshow and also to control the speed of the slideshow. The control point of the Photo Display System – Photo Display App - is implemented using Android sdk and the Media Renderer – Photo Frame (a desktop application) –is implemented using Java programming language.

Lastly, developing the Photo Display System was a great learning experience to gain valuable knowledge about the UPnP protocol and its different network components. In addition, the project also helped me understand the real-time use of REST web services and APIs in particular.
REFERENCES


