**Slide 1: Introduction**

* Assignment 2 Milestone 4.
* BTP400 Java Object-Oriented, Professor Peter Liu.
* Khai Phan and Carlianz Pura.

**Slide 2: Presentation Objective**

* Quick look at how Java especially useful for Android development.
* Short video demo of the program.
* Screenshow of how the app looks like.
* Go details about methods that use to support the app.

**Slide 3: Java in Android Development**

* Java is a platform independent, which means the app can be run almost anywhere, as long as the device contain a Java Virtual Machine (JVM) to execute the program.
* It is object-oriented.
* It has better security than any other language (such as not using pointer, only references and any other manual allocation that developer may forgot to “close” at the end of program which could cause memory leaks).
* Single inheritance not just allow developer to manage the code well but also allow them to use some method that has been declared without defining the object.
* Using multi-threading Is a big advantage for Java and especially Android. As some heavy app and with the development of Android phone nowadays, it is commonly to see an app that run at 2 or 3 physical cores at the same time to load the program.
* Java language is an open sources. Because of that, it contains a rich set of core features (which are completed and regularly update) and support framework for outside features such as networking, multi-threading, IO operations,…
* Moreover, it does support front-end and back-end which is prefectly for Mobile Development.

**Slide 4: APIs Usage in Application**

* So as you might have known from the Final Report I submitted, my application is running with 3 API Service.
* The first one is the Cloud Database that hold the record of username and password into a Cloud Server.
* The other 2 is an API that sort and fetch result of Flight and Hotel based on user’s argument of how they want the result to be.
* Latency refers to as the delay time that server take in average to return the result and the success rate is the number of times result will be return. (Since this is a free APIs, success rate will never guaranteed to be at 100%).

**Slide 5: Video Demo of the app**

* This is a short demo of how the app will be when it’s running.

**Slide 6: Assignment 2 Application (MainActivity.java)**

* This is the front page of the app, it has a username and password field, where the user will enter the information there and it will be fetch when the button Login is triggered. The data will be send to LoginRequest class with the Listener to do some checking to confirm user is exist before logged in. If there is any error occur, appropriate error message will be prompt.
* The Request will be added to a RequestQueue in case if there is another request waiting ahead.
* Button Register will direct user to RegisterActivity.java.

**Slide 7: Assignment 2 Application (LoginRequest.java)**

* This login request will post the username and password to an built-in login method in the server. It will check the data existed with the data received and return the confirmation in JSON Object. The Response Listener received from MainActivity.java will do the response checking.

**Slide 8: Assignment 2 Application (RegisterActivity.java)**

* The RegisterActivity will look similar to the MainActivity, instead it will have another field called Re-Enter Password to confirm that user know what they are typing. It will then do exact same step, send the information to RegisterRequest to upload the information and create an account for user. If there is any error occur, appropriate error message will be prompt.
* The Request will be added to a RequestQueue in case if there is another request waiting ahead.
* Go Back button will direct user back to MainActivity page if they no longer wish to craete an account.

**Slide 9: Assignment 2 Application (RegisterRequest.java)**

* About the same with LoginRequest, this method will send all of the data to a built-in register method in the server that store the object in the database. The server will confirm the data has been stored and return the confirmation in JSON Object. The Response Listener received from RegisterActivity.java will do the response checking.

**Slide 10: Assignment 2 Application (SecondActivity.java)**

* A second page of the app, where user can select between 1 of 2 API to use.
* Search Flight button will direct user to FlightActivity.java.
* Search Hotel button will direct user to HotelActivity.java.
* LogOut will log the user out of the program. User need to enter their credential again before accessing the APIs.

**Slide 11: Assignment 2 Application (FlightActivity.java)**

* The HotelActivity has 5 different field (argument) that user need to specify, origin place, destination place, fly-out and fly-in dates, with an optional budget price for each return flight.
* Field in the front-end user will be linked to the back-end so that whatever user type can be fetch at the back
* Price selection was using seekbar, where user can scroll horizontally to change the price.
* Fly-out and Fly-In date has a Calendar view that user can pick a date and it will parse it right into the field with the appropriate format. We use the Calendar class to fetch the calendar data in the system.

**Slide 12: Assignment 2 Application (FlightActivity.java continue)**

* This contain the most 2 important methods to fetch Flight result. The one on the left, is the validation that contains arguments from user after filling all the fields. This validation method use to ensure that the server and database is working before sending any further work.
* getQuotesList will have 2 array, for Quotes (price, dates) and Carrier (Airlines). A hashmap is created to help link the carrier name from the appropriate id returned from the server. After receiving the JSONObject, it will search through the object and find the appropriate data to display at FlightListAdapter.
* Each call will need a Request that contain a special url use to access the database, with a unique host name and api key to access the server. A response will be waiting and onCall after the database has been accessed to wait for a response from server.

**Slide 13: Assignment 2 Application (Flight.java)**

* Model for a Flight Object when displaying information (using System.out.print/println()).
* The left image is what will display when the server cannot find a result after sorting the database using user’s argument.

**Slide 14: Assignment 2 Application (FlightListAdapter.java)**

* This list Adapter act as a middle man where it takes the information from the Flight object and parse it into the the ViewHolder to display the data. The View, short for ViewHolder, is an .xml file that has the display result configured like the image on the left, where the FlightActivity return 2 results after it received from the server.

**Slide 15: Assignment 2 Application (HotelActivity.java)**

* The HotelActivity has 5 different field (argument) that user need to specify, destination place, number of guests, check-in and check-out dates, with an optional budget price per night.
* Similar to FlightActivity, price selection will have a seekbar and fields will be linked from front-end to back-end for usage.

**Slide 16: Assignment 2 Application (HotelActivity.java continue)**

* Calendar object is also using here, just like what we did for Flight for date selection.
* Search Hotel button will send all of the argument and some default value to HotelResult.java where it can filter the database.

**Slide 17: Assignment 2 Application (Hotel.java)**

* Model for a Hotel Object when displaying information (using System.out.print/println()).

**Slide 18: Assignment 2 Application (HotelResult.java)**

* There are 2 main method in this class that is use to fetch information. First one on the left is the getPlaceID. This will take the destination name entered by User and search in the API database for the matching ID of the destination. After the a matching ID has been found, it will pass to the right method which will fetch all of the result filtered using user’s argument.
* Just like Flight search, a spcial url with arguments, a unique host name and an api key is required to access the database and server.

**Slide 19: Assignment 2 Application (HotelListAdapter.java)**

* Same to FlightListAdapter, this HotelListAdapter will take the object Hotel, fetch the inner information and display it to the user.

**Slide 20: Finish**

* Attach a link of Google Drive