CNT5517/CIS4930 Mobile Computing

Spring 2024

Professor Sumi Helal

Lab 2 - Android Getting Started -- The APISearchAndroid App

Due: 12:00 Noon Thursday 6 February 2024

What to submit on Canvas:

- README.txt: your development environment, execution screen shots, and anything else you want to communicate to the TA
- APISearchAndroid.zip

INTRODUCTION

In this lab, you will learn how to setup the Android development environment and install all necessary developer tools on your computer in preparation for writing applications for Android. You will also learn how to use basic development tools to develop Android applications, from project creation to installation.

GOALS

- Setup the Android development environment (Install Android Studio).
- Create a *"Hello World"* Android application and run it on the Android Virtual Device (emulator).
- Understand the basic components of an Android project.
- Download and import the *APISearchAndroid* application and run it on the emulator.
- Make specific changes to the *APISearchAndroid* application.

1. Setting Up the Android Development Environment

To set up the development environment for Android, you first need to install Android Studio, which can be found at https://developer.android.com/studio/index.html
The download page also features the steps needed to install Android Studio. What follows is a quick summary of the steps.

Windows

1) Download the executable from the official page.
2) Run the installer
3) Follow the prompts. You can leave all of the options as their defaults.
4) Run Android Studio and follow the prompts. You can leave all of the options as their defaults. **Notice:** Make sure that you select the *Android Emulator Image* option when installing the SDK.

Mac

1) Download the dmg from the official page.
2) Mount the image and drag Android Studio into the Applications Folder
3) Run Android Studio and follow the prompts. You can leave all of the options as their defaults. **Notice:** Make sure that you select the *Android Emulator Image* option when installing the SDK.

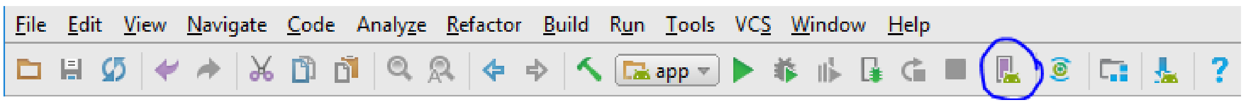
Linux

1) Download the Android Studio archive from the official page.
2) Depending on your distro, use your package manager to install Android Studio's needed dependencies (libc6 libncurses5 libstdc++6 lib32z1 libbz2-1.0). **Notice:** If your machine is 64-bit, you must install the 32-bit versions of the aforementioned libraries
3) Unarchive Android Studio to your installation folder of choice (Google recommends `/opt`) and run `android-studio/bin/studio.sh`. You can leave all of the options as their defaults. **Notice:** Make sure that you select the *Android Emulator Image* option when installing the SDK.

2. Creating and Running *Hello World* Application

In this lab, we only require you to be able to run your app on a device emulator (Android Virtual Device or AVD). Of course, you should be able to run your app on an actual Android device if you have access to one, but you are not required to. To set up an AVD:

- Select the menu **Tools** => **Android** => **AVD Manager**, or click on the phone shaped icon in the toolbar. (**Notice:** If you have followed the installation steps, then a device should already be listed as *Nexus 5X API 25 x86, or later devices*)


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
app

If you do not see any devices, then follow these steps to create an Android Emulator.

- Open the **Android Virtual Devices** window.
- Click the **Create Virtual Device** button (located in the bottom left).
- Select **Nexus 5X**.
- On the next screen, select the target build that you would like to run the application on, e.g., "**Nougat**".
- Click **Next** and then **Finish** to terminate the SDK/AVD Manager Window.

Now, you have created an Android emulator. Next, create the *Hello World* app:

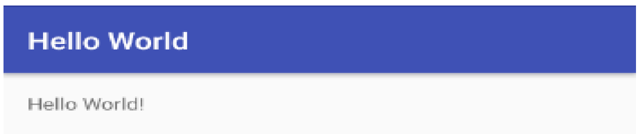
- Click the menu **File** -> **New** -> **Android Application Project**.
- Name the project “Hello World” and set location of project, or use default location
- Check the latest release/API level from the Build Target SDK list, and leave other settings as default.
- On the next screen, you can select an activity template from which to begin building your app. For this project, select **Empty Activity** and click **Next**.

  *By selecting **Empty Activity**, a Hello World app is automatically generated. How convenient!! The ADK is doing the assignment for you *
- Both the activity and its associated layout must be named. Leave the default selections and click **Finish**.

Now, run Hello World App on the Emulator:

- Select **Run** => **Run App** in the toolbar, then select your AVD and click **OK**.
- Select or make sure you are under **Android Application.** Then type in or browse and select your project in the **Android** tab. Then in **Target** tab, choose the emulator you just created. Click **Run**.

The emulator will be launched (after some delays, so be patient). If all goes well, you should see the following result.


Hello World

Hello World!

Congratulations, you've just created and ran your first Android Application!
(Buy yourself an Oatmeal cookie)

3. Directories and Files in an Android Project

Now let us look at the workspace of the *Hello World* app and get to know the useful directories and files in an Android project. Double click your *HelloWorld* project in the Package Explorer to expand it, if it is not already expanded. You should see the following directories:

- `src/`: directory for app's main source code (includes Activity class)
- `AndroidManifest.xml`: describes all metadata about the app that the Android OS will need to run the app properly.
    - Package name used to identify the application.
    - List of Activities, Services, Broadcast Receivers, and Content Provider classes and all of their necessary information, including permissions.
    - Libraries and API levels that the application will use.
- `res/`: directory for app resources. The main content categories include drawables, layouts, and values.
    - `drawable/`: directory that holds image and animation files designed for screens with different densities.
    - `layout/`: directory for files that define your app's user interface. It already contains a file called `activity_main.xml` which defines the user interface for your HelloWorld Activity class. Double clicking on this file will open up the Android UI Editor that you can use to help generate the xml layout files.
    - `values/`: directory where you define (in XML) your globally used colors, dimensions, strings and styles.
- `gen/`: directory where code is generated for all the resources defined in your res folder. It already contains one file called "R.java" which is a special static class that is used for referencing the data contained in your resource files.
- `assets/`: directory where stores miscellaneous files you need to access in your code as raw data. Unlike files in the res folder, the only way to load something from assets is to programmatically open it as a file.

4. Activities

An Activity is an application component that provides a screen with which the user and the application can interact. For instance the user may interact with the application by dialing a phone number, taking a photo shot, sending an email, or viewing a map. Each activity has an associated **Activity Layout** (in XML) that is provided as a window with user interface widgets.

An application usually consists of multiple activities that are loosely bound to each other. Typically, one activity in an application is specified as the "main" activity, which is presented to the user when launching the application for the first time. Each activity can then start another activity using Intent in order to perform different actions. Each time a new activity starts, the previous activity is stopped, but the system preserves the activities in a stack (the "back stack"). Activities must be declared in the manifest file in order for them to be accessible to the system. Creating an activity in a project through **File** => **New** => **Activity**, which will automatically create a declaration for that activity in the manifest file. However, an activity can also be declared (or a declaration edited) by opening the manifest file and adding or editing an <activity> element as a child of the <application> element.

For more information about how to implement an Activity, how to create and use Intent, how to implement user interface layout for an Activity, how to declare the Activity in the Manifest file and so forth, read Android developer webpage.

5. Download and Import the *APISearchAndroid* App

- Download the source code of the *APISearchAndroid* app covered in the Android lecture from Canvas->Files->Lab2->APISearchAndroid.zip
- Unpack the ZIP file and copy/paste the *APISearchAndroid* folder to your Android Studio workspace (i.e., the folder where your *Hello World* app is located).
- Next, to open the *APISearchAndroid* project into your Android Studio project, click the menu **File** => **Open**
- In this screen, browse and select the directory where you just saved your *APISearchAndroid* folder. Click **OK**, then **Next**, and then **Finish**.

Now you have opened the *APISearchAndroid* App in your Android Studio.

6. Setting up the *APISearch* Server

You can now run the starting point for the APISearch App, but it won't have an API to connect to. To remedy this, you will run a server locally which will let you make some queries. Download the source for the API server from Canvas->Files->Lab2->mobile_computing_server.zip The API server is a small node app, meaning you will need to download *npm* from your package manager or from https://www.npmjs.com/. Once installed, change into the API server's directory in your command line and run *npm install*, followed by *npm start*. This will cause the server to start running locally on port 8080. You can check this by navigating to http://localhost:8080/api/s/ in your web browser; this is the base URL for making search queries manually or in your app. If you stop the server, remember to start it again with *npm start* before testing your Android app in the simulator.

7. Making Changes to *APISearchAndroid* App

In this project, you are required to make following two changes to the APISearchAndroid application you just imported.

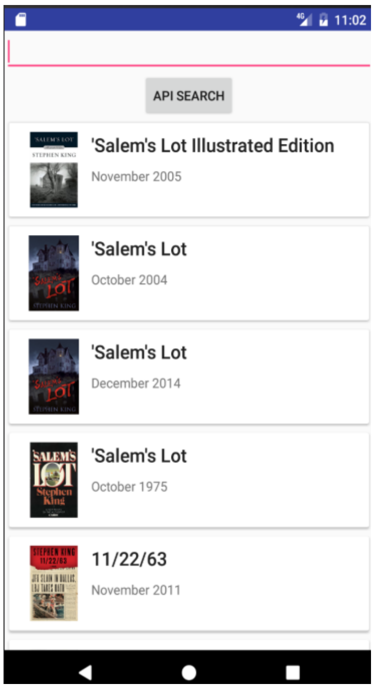
1. A user can choose any result from the list of the returned results by clicking on it. When the user clicks a particular result, it will navigate to a new screen which displays the user selected result information (title, text, image).

To achieve 1, you will need to create a new Empty Activity (name it ResultDisplayActivity) to display the information about the user selected result. Then you will need to add new function to the APISearchActivity class to be able to parse the user selected item from the user interface and initiate the ResultDisplayActivity (So now you are doing Java programming). Finally, you will have to reconfigure the application Manifest to use your new name retrieval Activity on startup.

2. Add back button functionality to the layout of the result display (result.xml). This button allows the user to finish the current activity and load the previous screen view (i.e., from result to the main layout) by clicking on it. Note that, at any screen (e.g., main layout) that is loaded by pressing back button, it should still be able to perform its function properly (e.g, in the main view, user is still able to click on any result from the list to navigate to a new screen where its info is displayed).

Extra Credit.

For extra credit, implement a way to add a result as a “favorite” result. The storage of those marked as favorites should be on the local device (not on the API endpoint). Feel free to use an ImageButton with the image of a star as the button to mark a result as a favorite. **Make sure that the ID of the button is starButton**. This will allow for our grading scripts to be able to interface with those buttons. Now your modified app should look like.



'Salem's Lot Illustrated Edition
November 2005

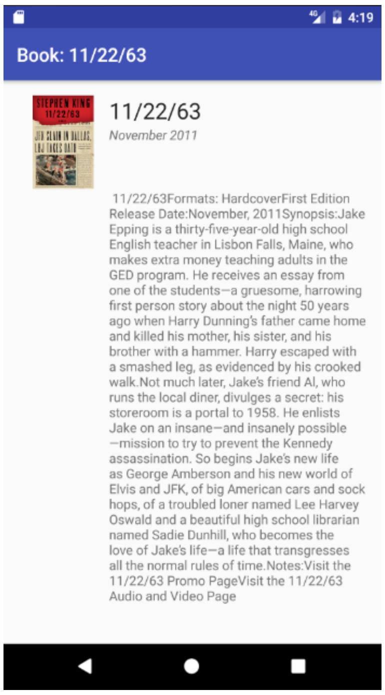
'Salem's Lot
October 2004

'Salem's Lot
December 2014

'Salem's Lot
October 1975

11/22/63
November 2011

APISearchActivity



Book: 11/22/63

11/22/63
November 2011

11/22/63Formats: HardcoverFirst Edition Release Date:November, 2011Synopsis:Jake Epping is a thirty-five-year-old high school English teacher in Lisbon Falls, Maine, who makes extra money teaching adults in the GED program. He receives an essay from one of the students—a gruesome, harrowing first person story about the night 50 years ago when Harry Dunning's father came home and killed his mother, his sister, and his brother with a hammer. Harry escaped with a smashed leg, as evidenced by his crooked walk.Not much later, Jake's friend Al, who runs the local diner, divulges a secret: his storeroom is a portal to 1958. He enlists Jake on an insane—and insanely possible —mission to try to prevent the Kennedy assassination. So begins Jake's new life as George Amberson and his new world of Elvis and JFK, of big American cars and sock hops, of a troubled loner named Lee Harvey Oswald and a beautiful high school librarian named Sadie Dunhill, who becomes the love of Jake's life—a life that transgresses all the normal rules of time.Notes:Visit the 11/22/63 Promo PageVisit the 11/22/63 Audio and Video Page

ResultDisplayActivity

8. Exporting Android Application and Deliverables

The Android system requires that all installed applications be digitally signed with a certificate. The system will not install an application on an emulator or a device if it is not signed. However, Android Studio has been automatically signing your application with its own "debug" certificate and compile your source code as .apk file under res/ folder. Although signing with the "debug" certificate is not acceptable for release to the general public, an application with “debug” certificate can be installed on an emulator or personal Android device. As of now, you've only been executing your application on an emulator by launching it through Android Studio’s AVD or your personal Android device. So debug certificate should be sufficient for our use. For more information (e.g., signing your application in release mode), read http://developer.android.com/tools/publishing/app-signing.html

To complete this lab you are required to submit your source code through Canvas. Pack the folder of your project into a .zip file and submit the compressed file to Canvas. For example, if you have not changed the project name, you should compress the folder “APISearchAndroid” into APISearchAndroid.zip. Make sure your folder can be successfully imported and run in the Android Studio before submitting your code.