

January | 17

Phone Cop

by Pavan Teja

GitHub Username: **pa1-teja**

Table of Contents

Description	1
Intended User	1
Features	1
User Interface Mocks	1
Key Considerations	5
How will your app handle data persistence?	5
Describe any libraries you'll be using and share your reasoning for including them.	5
Describe how you will implement Google Play Services.	5
Next Steps: Required Tasks.....	5
Task 1: Project Setup	5
Task 2: Implement UI for Each Activity and Fragment.....	11
Task 3: Create free and paid variants of the app	11
Task 4: Implement app's functionality	11
Task 5: Error Cases.....	11
Task 6: Accessibility and Localization.....	11

Description

Phone Cop is an app that is used to capture any kind of evidence in the case of emergency. This app will record audio and video clips of the surroundings upon trigger. This helps the victim to obtain evidence and helps them take necessary action.

Intended User

This app is intended to all the Android users of all the age groups across the globe.

Features


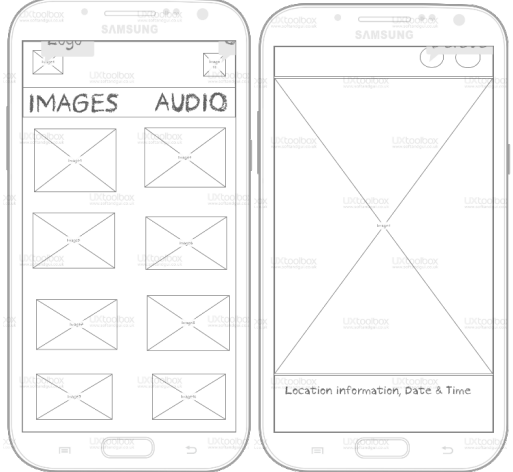

List the main features of your app. For example:

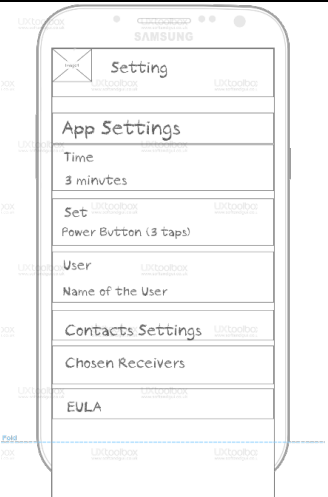

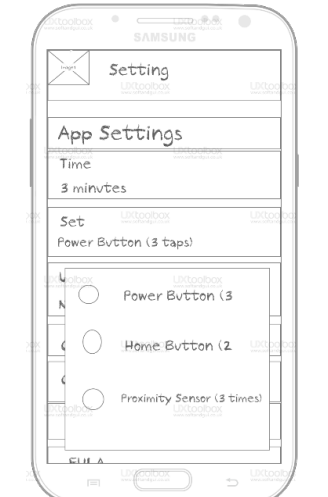
- Saves information locally
- Sends audio and image files to the list of chosen contacts by the user via WhatsApp, Email, MMS etc.
- Records audio and captures images in stealth mode.
- Allows users to set trigger to automatically start recoding in stealth.
- Allows user to customize the message sent to the users when the app receives the signal.
- Free version of the app displays Google Ads and paid version doesn't.




User Interface Mocks

This screen is the welcome screen greets the user with a nice background when the app is opened.



<p>The user will be navigated to a tabbed screen where the app displays the images and audio clips if there are any images captured and recorded audio files earlier. Otherwise the app will display a message saying “There are no image or audio evidences recorded”.</p>	
<p>The app will display all the images stored in the app’s SQLite database. User can tap on any image and view the full screen image with date, timestamp and the location where the image is captured.</p>	
<p>The app will also display audio clips stored in the app’s SQLite database and the user can tap on any audio clip from the list of audio clips to see the details of the clip such as date, timestamp, location details and play/pause button to play the audio clip</p>	

<p>The settings screen is displayed to the user when the user taps on the overflow menu in the App bar where the app lets the user customize the app. The user can set the timer, trigger and also modify user details, choose list of contacts from the contacts app etc.</p>	
<p>The app lets the user to specify a time frame (i.e. 1 minute, 2 minutes, 3 minutes, 5 minutes.) for the app to run in the background and record audio and capture images continuously till the specified time is complete.</p>	
<p>The app lets the user to set a trigger (i.e. Power Button (3 taps), Home Button (2 taps), Proximity Sensor (3 times in 5 seconds.) so that the user can conveniently use that trigger to start the app's process in the background.</p>	

<p>The app lets the user to modify the user details anytime and use them when the app needs to send the recorded audio and captured images to the chosen list of contacts.</p>	
<p>The app lets the user to add list of contacts to the app to send the captured images, and audio clips along with the device's location. So that all the necessary steps can be taken by those contacts.</p>	
<p>The user see the End User License Agreement by tapping the EULA button in the settings screen.</p>	

Key Considerations

How will your app handle data persistence?

The app uses the SQLite database using Content Providers to store captured images and recorded audio clips along with the location details. The app also uses Shared Action Preferences to store the app's settings such as user details, the trigger that is chosen by the user, the window time set by the user etc. since this is important and little information which should be available easily to the app to initiate the mechanism and also to improve the app's performance at the same time.

Describe any libraries you'll be using and share your reasoning for including them.

The app uses Picasso image handling API to handle the loading and caching of images, Android design library for UI Material Design to make the UI look more polished with transitions which help user to understand to use the app. And Retrofit and OKHTTP APIs for the app's network activity.

Describe how you will implement Google Play Services.

The app uses Google Play Services to authenticate the user's Google account, to display Google Ads on free variant of the app, and for app analytics by adding the appropriate dependencies in to the **build.gradle** file.

Next Steps: Required Tasks

This is the section where you can take the main features of your app (declared above) and decompose them into tangible technical tasks that you can complete incrementally until you have a finished app.

Task 1: Project Setup

First we need to set the environment before starting the application development. Below are minimum system requirements and steps to set up the Android Studio IDE and start developing an Android App.

System Requirements

Windows

- Microsoft® Windows® 8/7/Vista/2003 (32 or 64-bit)
- 2 GB RAM minimum, 4 GB RAM recommended
- 400 MB hard disk space
- At least 1 GB for Android SDK, emulator system images, and caches
- 1280 x 800 minimum screen resolution
- Java Development Kit (JDK) 7

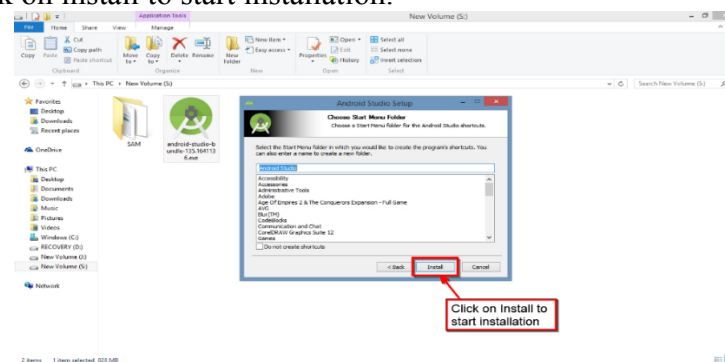
- Optional for accelerated emulator: Intel® processor with support for Intel® VT-x, Intel® EM64T (Intel® 64), and Execute Disable (XD) Bit functionality.

Tools Requirement

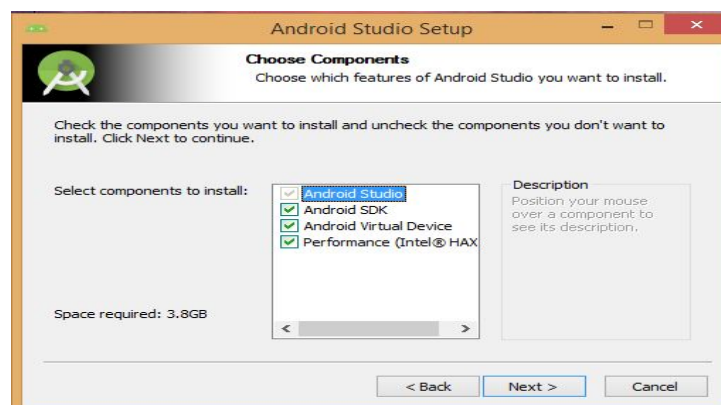
- Android Studio version 1.1.0 and above.
- Android SDK
- Java Development Kit (JDK) v7 and above.
- JDK v7 or above should be present on the system before installing the Android Studio.
 - If not installed download it from <http://www.oracle.com/technetwork/java/javase/downloads/index.html>.
 - After installing the JDK software, set the Class Path Environment variable by following the steps in <https://docs.oracle.com/javase/tutorial/essential/environment/paths.html> according to the platform on which you're working on.

Installing Android Studio on Windows Platform

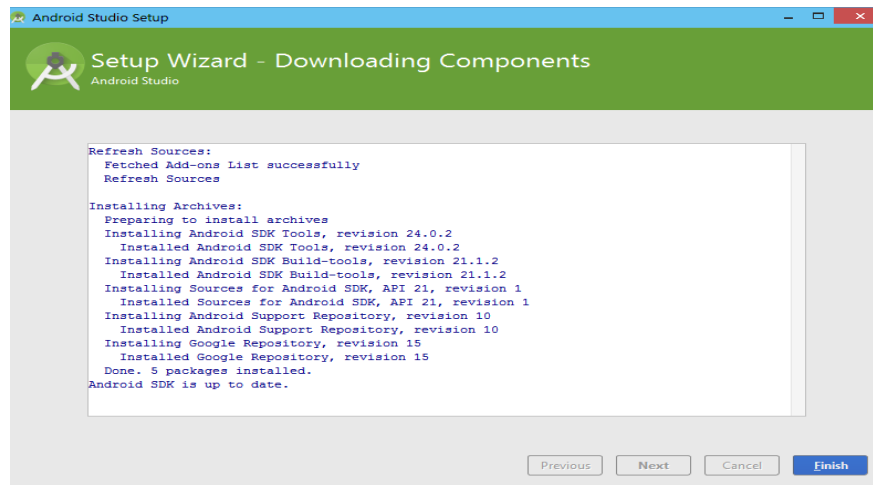
- Download the Android Studio IDE from <https://developer.android.com/sdk/index.html>
- Go to the directory where the file is downloaded and launch the .exe file.
- Follow the setup wizard to install Android Studio and any necessary SDK tools.
- Steps to install Android Studio
 1. Click on install to start installation.



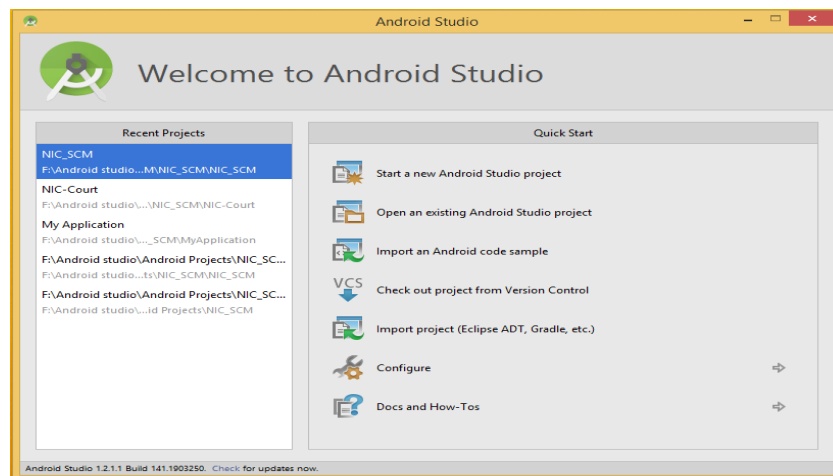
2. Select all the checkbox and click on next button



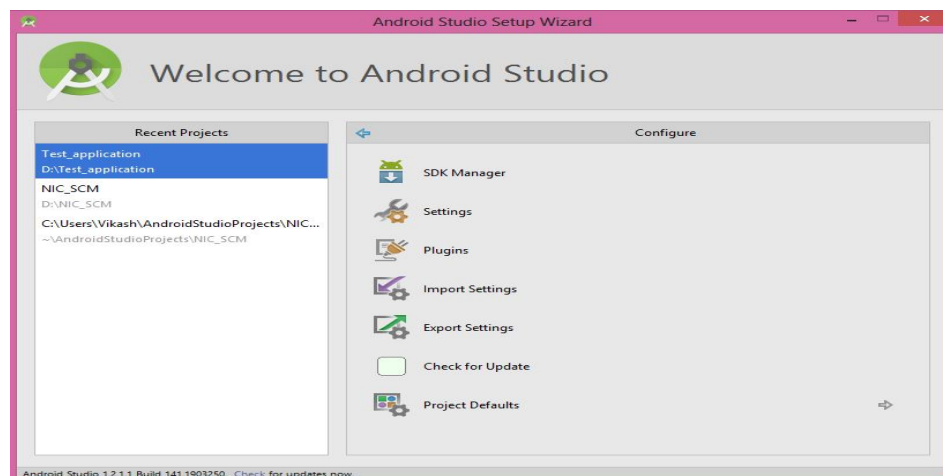
3. After finishing the SDK download, click “finish” to complete the installation.



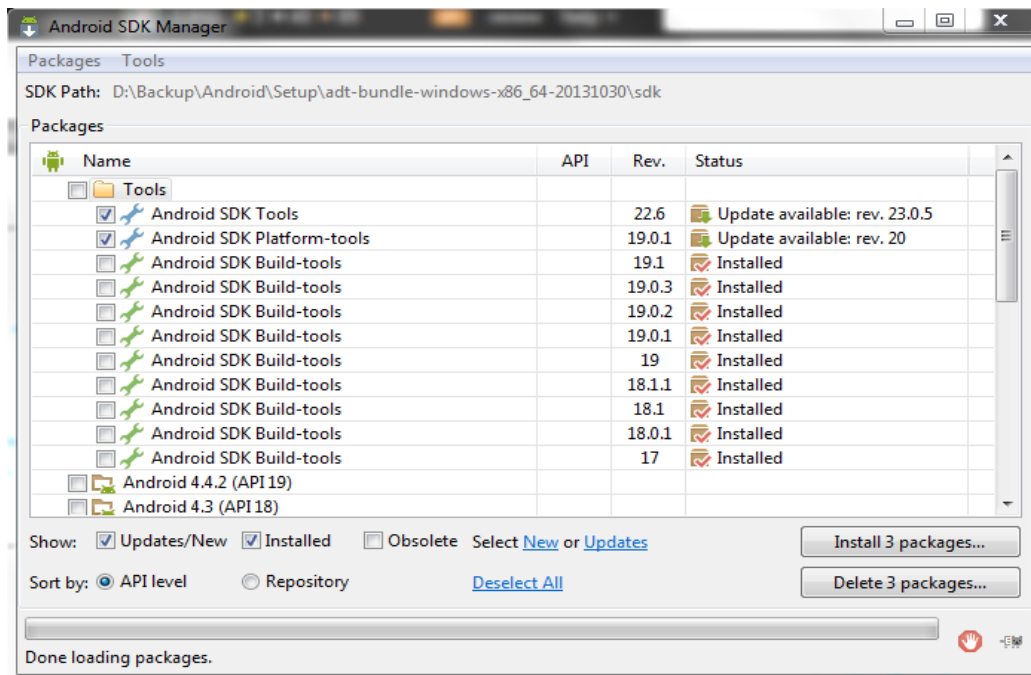
4. Click on 'configure' in the quick start menu to configure your Studio IDE.



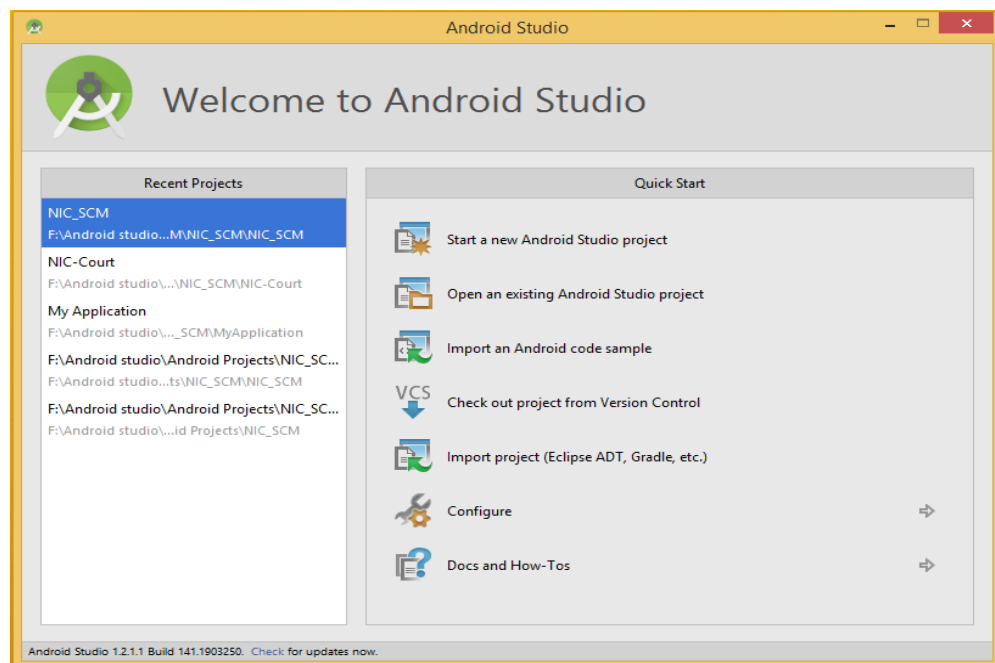
5. Click on SDK manager to download all the required APIs to start developing the android application.



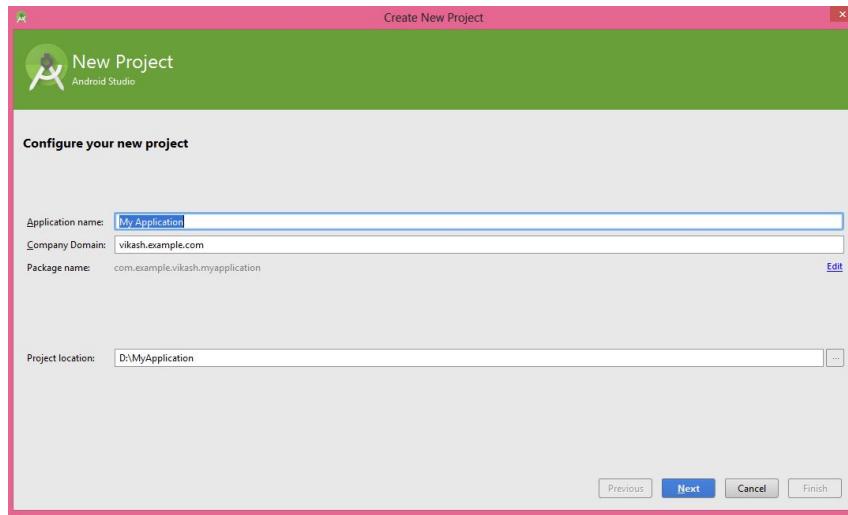
6. Select the required tools from the SDK manger window and click on Install.



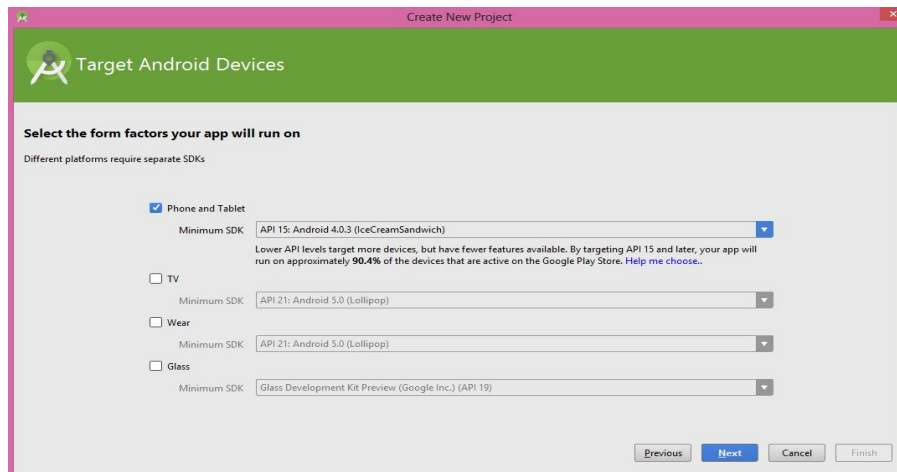
7. Click on “Start anew Android Studio project” to create a new project.



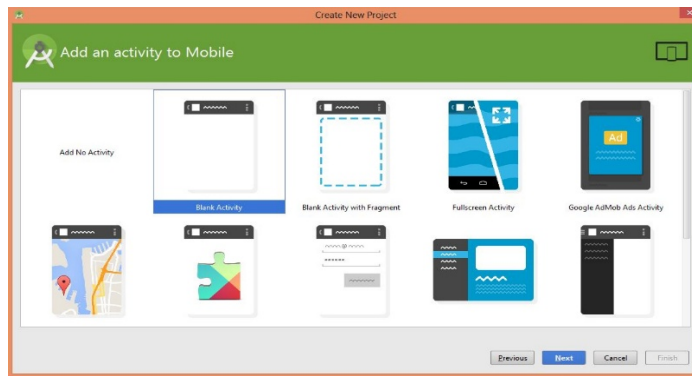
8. Type the name of the project in the “Application Name” text box and click on next.



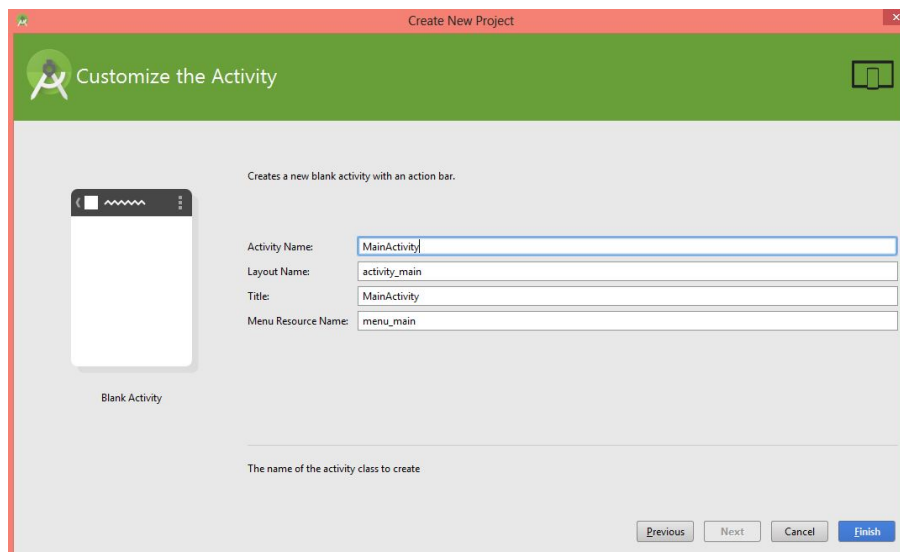
9. On the next window, we have to select the device which we target our project (i.e. phone or tablet, TV, Glass, Wear). And also select the minimum SDK from the dropdown and click on next.



10. On the next window, select the type of activity to use in your project.



11. Name your selected activity, and click on finish to create the project.



Adding the dependencies to the project

- Expand the project view in the IDE and open the **build.gradle** file and add the following dependencies.

```
dependencies {
    compile fileTree(include: ['*.jar'], dir: 'libs')
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
    compile 'com.android.support:appcompat-v7:25.1.0'
    compile 'com.android.support:design:25.1.0'
    testCompile 'junit:junit:4.12'
    compile 'com.google.android.gms:play-services:10.0.1'
    compile 'com.squareup.retrofit2:retrofit:2.1.0'
    compile 'com.squareup.okhttp3:okhttp:3.5.0'
    compile 'com.squareup.picasso:picasso:2.5.2'
}
```

Task 2: Implement UI for Each Activity and Fragment

- Build UI for MainActivityFragment with dummy data and text transitions.
- Build UI for SettingsActivity with all the options.
- Implement UI transitions.
- Test run the app to check if app runs smoothly.

Task 3: Create free and paid variants of the app

- Create free and paid variants of the app from the **build.gradle** file with separate packages.
- Ensure that the variants perfectly built.

Task 4: Implement app's functionality

- Implement code to fetch actual data using the added libraries.
- Store the data in the SQLite Database.
- Implement code to sync the wearable device with latest information.

Task 5: Error Cases

- Check the whole code for any bugs, app crashes and fix them.

Task 6: Accessibility and Localization

- Implement talk back feature and make the app available in multiple languages spoken in India.