

# **DAYANANDA SAGAR COLLEGE OF ENGG.**

(An Autonomous Institute Affiliated to Visvesvaraya Technological University, Belagavi & Approved by AICTE, New Delhi.)

**Department of Computer Science And Engg.**

**Subject: Mobile Application Development(MAD)**

**By**

**Dr. Rohini T V,  
Associate Professor  
V: A & B Section**

**Department of Computer Science & Engineering**

**Aca. Year ODD SEM /2022-23**

# Permission (Module-5)

---

Students will be learning Permissions, Performance and Security ,  
Firebase and AdMob ,Publish

# Topic Name

- Permissions, Performance and Security
- Firebase and AdMob
- Publish

## Permissions

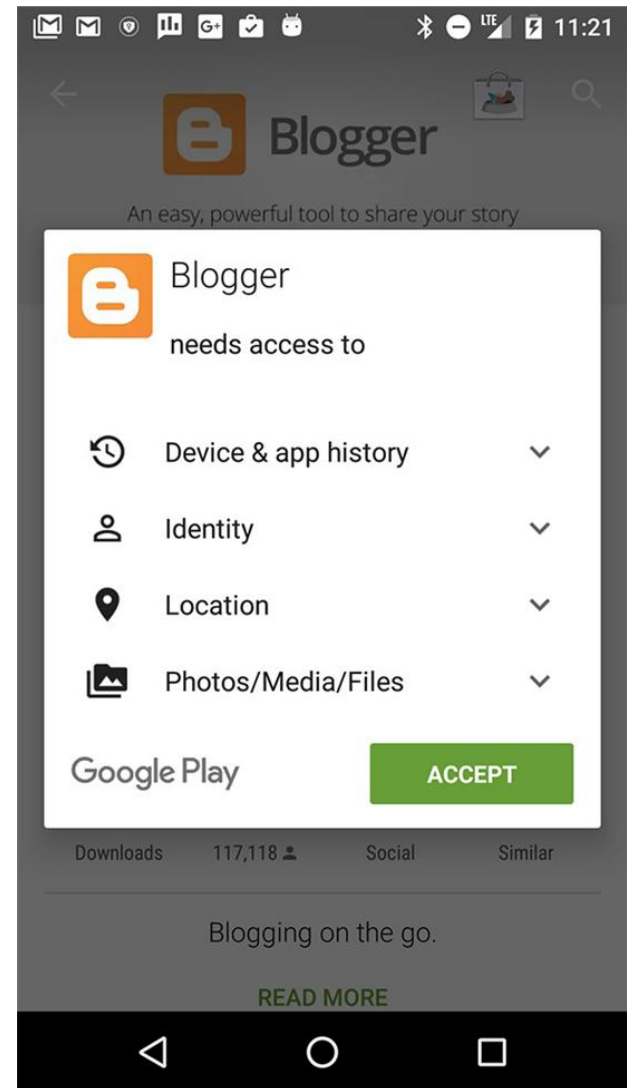
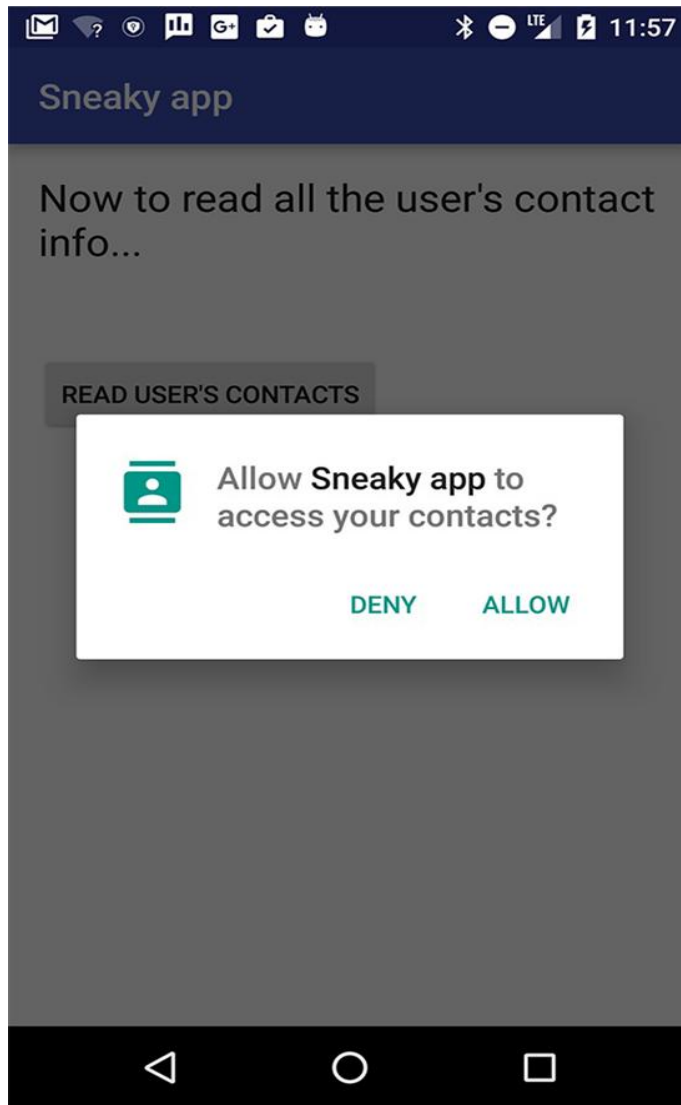
As you have worked through the practical's, there were times when your app needed to get permission to do something, including when it needed to:

- connect to the Internet.
- use a content provider in another app.

Ask permission if it isn't yours An app is free to **use** any **resources** or **data** that it creates, but must get **permission** to use anything—data, resources, hardware, software—that does not belong to it. For example, your app must get permission to read the user's **Contacts** data, or to use the device's camera. It makes sense that an app needs permission to read a user's Contacts, but you might wonder why it needs permission to use the camera. It is because the **camera** hardware does not belong to the app, and your app must always get permission to use anything that is not part of the app itself.

## Normal and dangerous permissions :

- Android classifies permissions as normal or dangerous.
- A normal permission is for actions that do not affect user privacy or user data, such as connecting to the Internet.
- A dangerous permission is for an action that does affect user privacy or user data, such as permission to write to the user's voicemail.
- Android automatically grants normal permissions but asks the user to explicitly grant dangerous permissions.



## Best practices for permissions

- When an app asks for too many permissions, users get suspicious. Make sure your app only requests permission for **features** and **tasks** it really needs, and make sure the **user understands** why they are **needed**.
- Wherever possible, use an **Intent** instead of asking for permission to do it yourself.
- For example, if your app needs to use the camera, send an **Intent to the camera app**, and that way the camera app will do all your work for you and your app **does not need to get permission** to use the camera (and it will be much easier for you to write the code than if you accessed the camera APIs directly).

# Keep long-running tasks off the main thread

This course has already talked about moving work off the main thread into the background to help keep the UI smooth and responsive for the user. The hardware that renders the display to the screen typically updates the screen every 16 milliseconds, so if the main thread is doing work that takes longer than 16 milliseconds, the app might skip frames, stutter or hang, all of which are likely to annoy your users.

You can check how well your app does at rendering screens within the 16 millisecond limit by using the Profile GPU

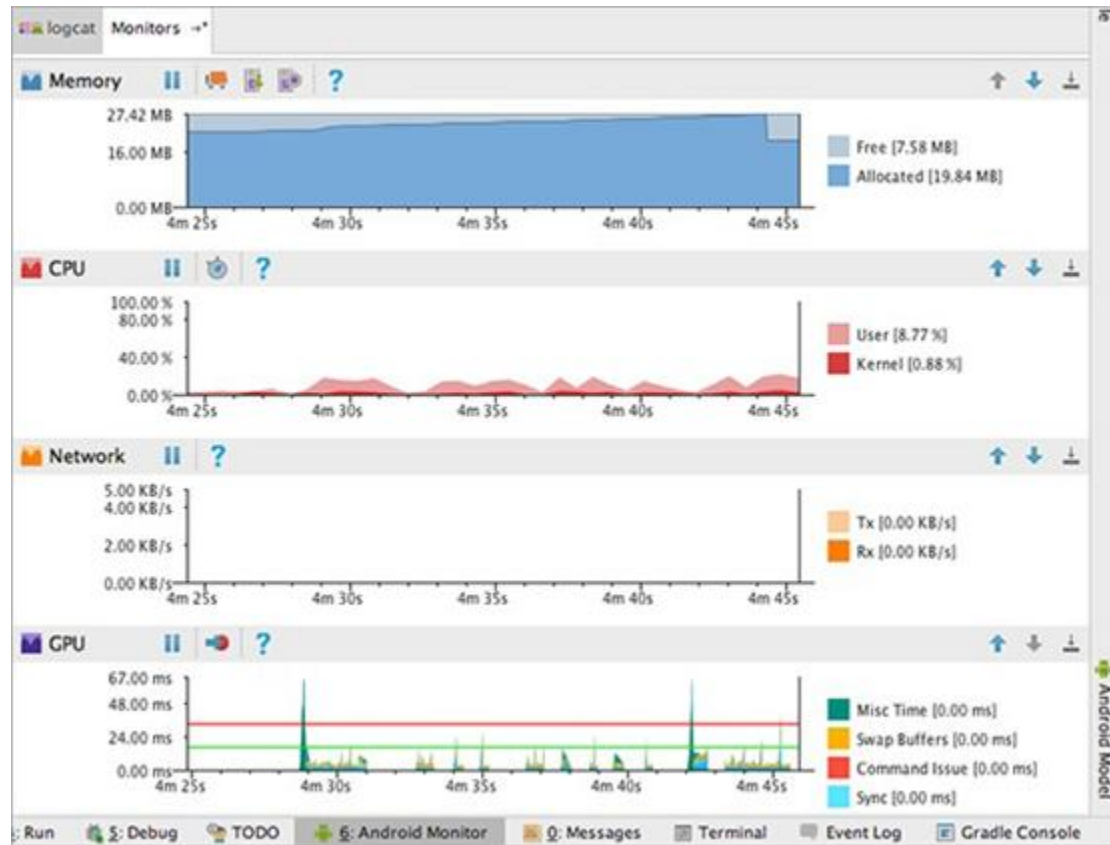
Rendering tool on your Android device.

1. Go to Settings > Developer options.
2. Scroll down to the Monitoring section.
3. Select Profile GPU rendering.
4. Choose On screen as bars in the dialog box.



The monitors are:

1. **Memory monitor**—Reports how your app allocates memory and helps you to visualize the memory your app uses.
2. **CPU monitor**—Lets you monitor the central processing unit (CPU) usage of your app. It displays **CPU usage in real time**.
3. **GPU monitor**—Gives a visual representation of how long the graphical processing unit (GPU) takes to **render frames** to the screen.
4. **Network Monitor**—Shows when your application is making **network requests**. It lets you see how and when your **app transfers data**, and **optimize** the underlying **code** appropriately.



# Security best practices :

## 1. Handling user data

This lesson has already discussed how Android uses permissions to make sure apps cannot access the user's personal data without their permission.

But even if the user gives your app permission to access their private data, do not do so unless absolutely necessary.

And if you do, treat the data with integrity and respect. For example, just because the user gives your app permission to update their calendar does not mean you have permission to delete all their calendar entries.

## 2. Public Wi-Fi

Many people use mobile apps over public Wi-Fi. When did you last access the Internet from your mobile phone over the public Wi-Fi at a coffee shop, an airport, or a railway station?

Design your app to protect your user's data when they are connected on public Wi-Fi. Use https rather than http whenever possible to connect to websites. Encrypt any user data that gets transmitted, even data that might seem innocent like their name.

For transmitting sensitive data, implement authenticated, encrypted socket-level communication using the SSLSocket class. This class adds a layer of security protections over the underlying network transport protocol. Those protections include protection against modifications of messages by a wiretapper, enhanced authentication with the server, and increased privacy protection.

### 3. Validating user input

If your app accepts input (and almost every app does!) you need to make sure that the input does not bring anything harmful in with it.

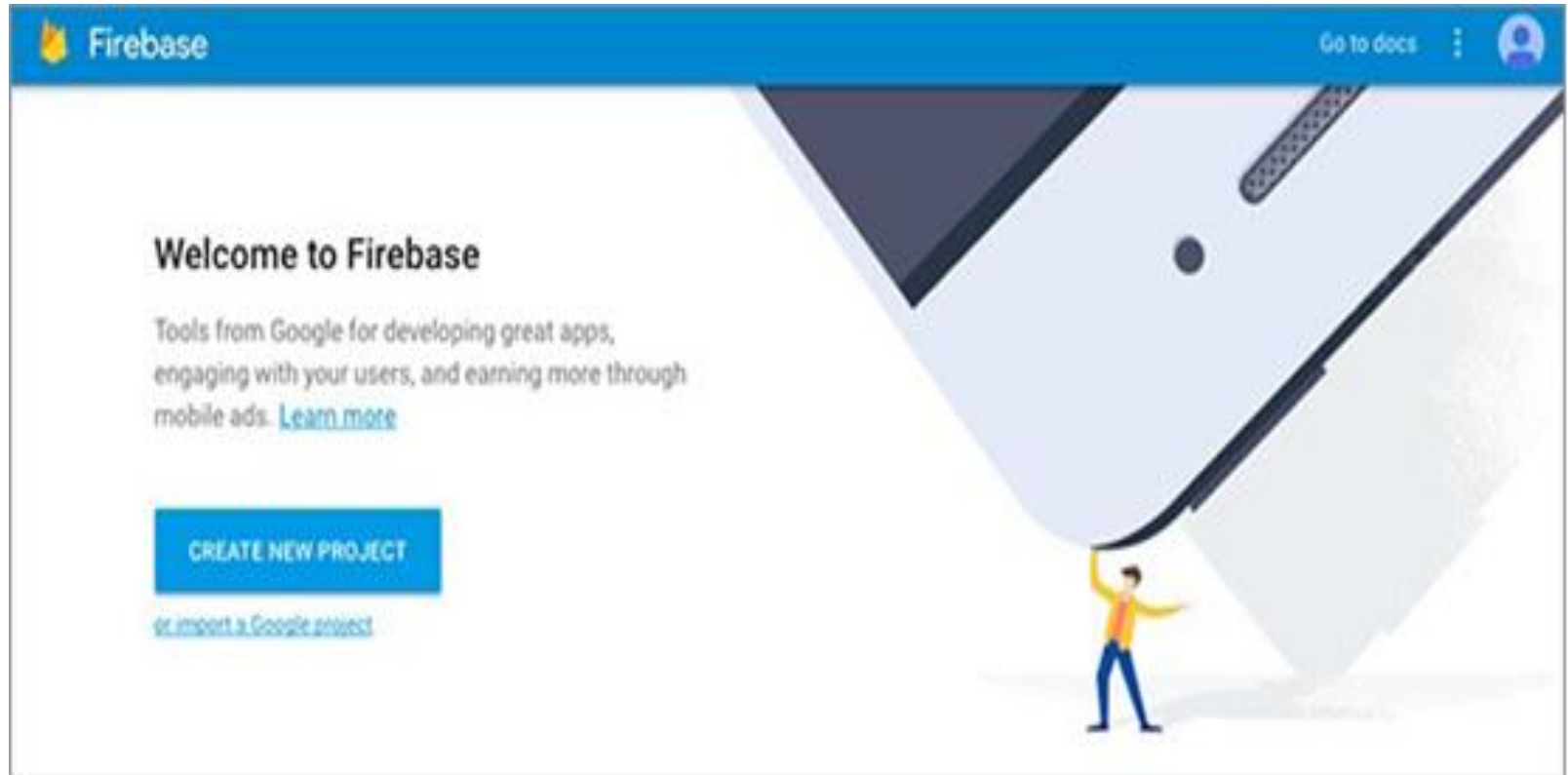
If your app uses native code, reads data from files, receives data over the network, or receives data from any external source, it has the potential to introduce a security issue.

The most common problems are buffer overflows, dangling pointers, and off-by-one errors. Android provides a number of technologies that reduce the exploitability of these errors, but they do not solve the underlying problem. You can prevent these vulnerabilities by carefully handling pointers and managing buffers.

# Firestore

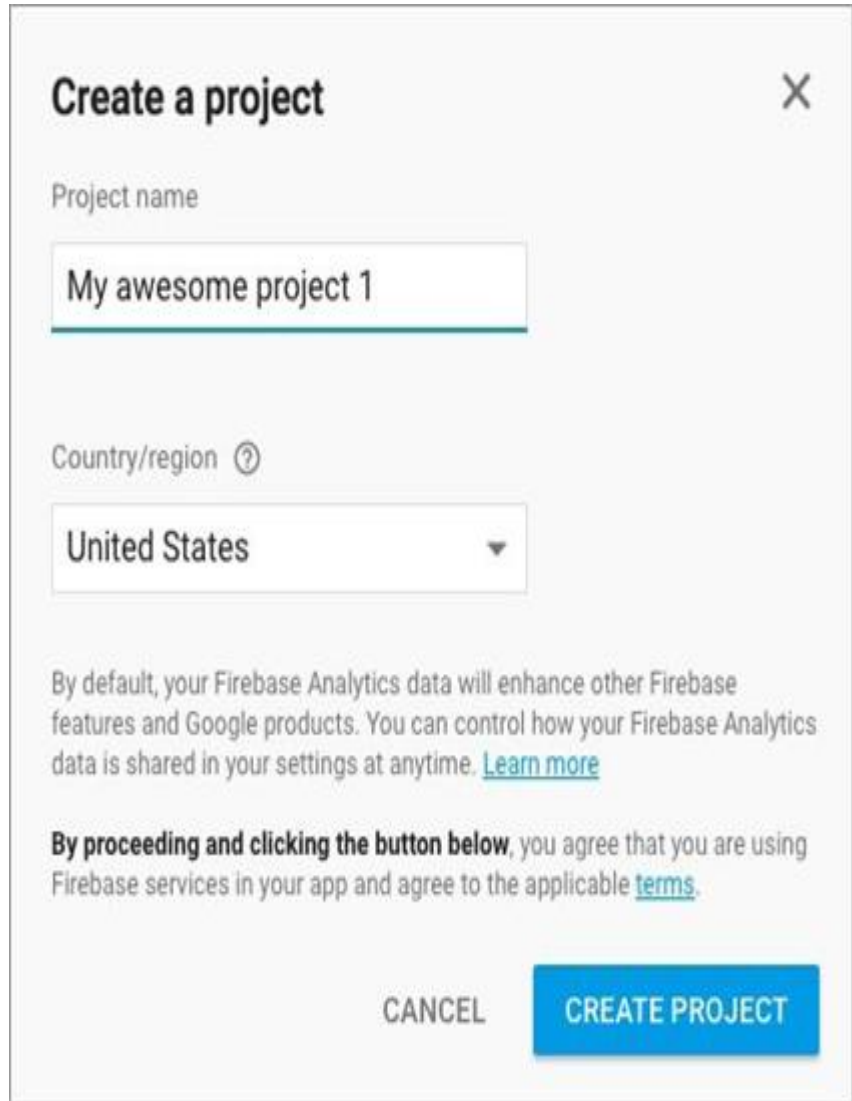
- Firestore is a set of tools for app developers, but not just for Android app developers. It is for iOS app developers and web app developers too. However, since this is a course about Android development, this lesson only talks about how to use
- Firestore with Android apps, an Android developer, you use Android Studio to build your app, but you can use Firestore to add features to your app, get a wider audience for your app, test your app, earn revenue from your app, and get analytics on the usage of your app.

The first time you go to the Firebase you see a welcome screen that includes a button to create a new project.



A **project** is a container for your apps across platforms: Android, iOS, and web. You can name your project anything you want; it does not have to match the **name** of any apps.

1. In the Firebase console, click the **CREATE NEW PROJECT** button.
2. Enter your Firebase project name in the dialog box that appears then click **Create Project**.
3. The **Firebase console** opens. Look in the menu bar for the name of your project.



The screenshot shows a 'Create a project' dialog box with a close button (X) in the top right corner. It contains a 'Project name' text input field with the value 'My awesome project 1'. Below it is a 'Country/region' dropdown menu with a question mark icon and the selected value 'United States'. A paragraph of text explains that Firebase Analytics data will enhance other Firebase features and Google products, with a link to 'Learn more'. Below this is a bold statement: 'By proceeding and clicking the button below, you agree that you are using Firebase services in your app and agree to the applicable terms.' At the bottom right are two buttons: 'CANCEL' and 'CREATE PROJECT'.

**Create a project** X

Project name

My awesome project 1

Country/region ?

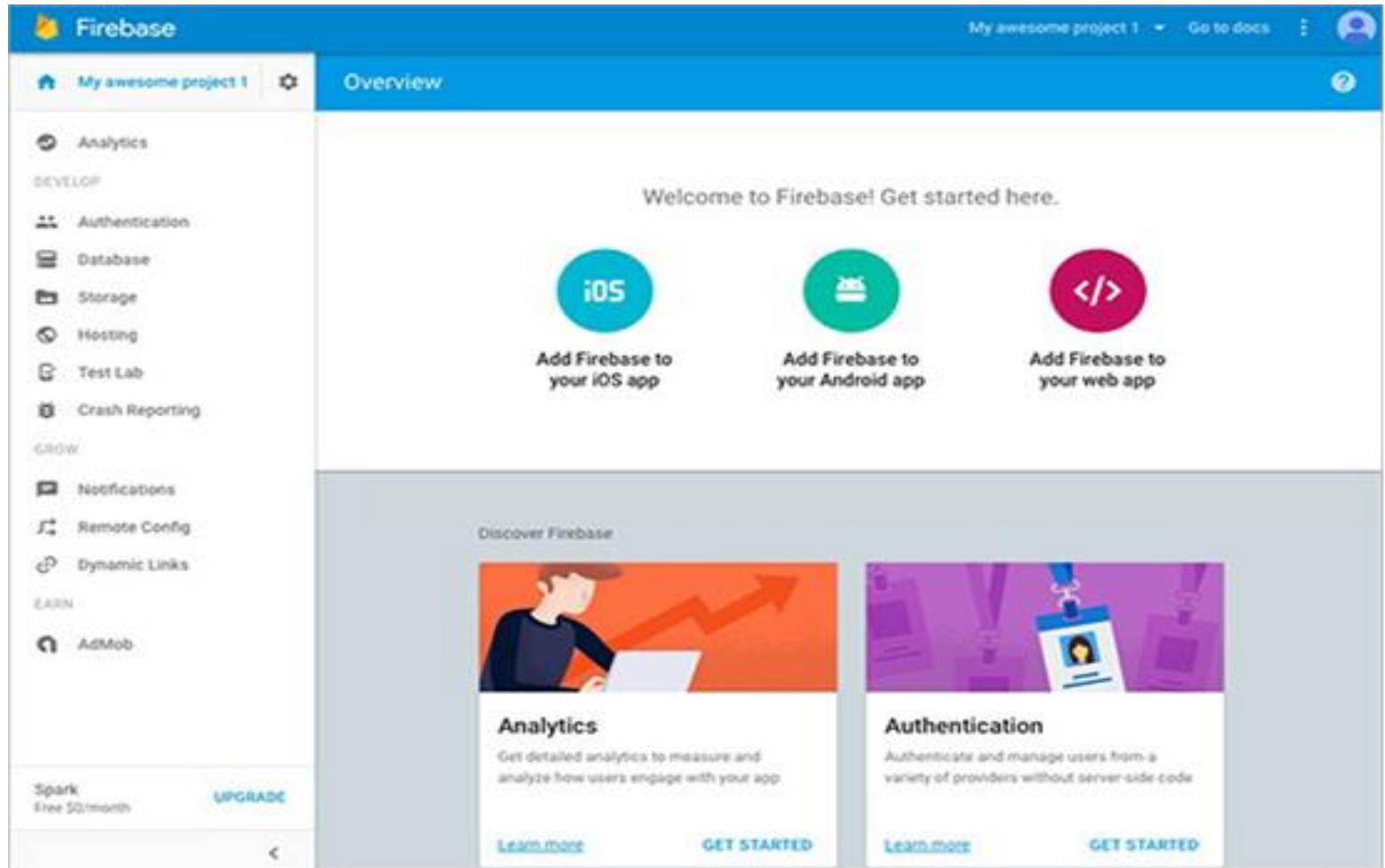
United States ▼

By default, your Firebase Analytics data will enhance other Firebase features and Google products. You can control how your Firebase Analytics data is shared in your settings at anytime. [Learn more](#)

**By proceeding and clicking the button below, you agree that you are using Firebase services in your app and agree to the applicable [terms](#).**

CANCEL **CREATE PROJECT**





## *Add your **Android app** to your **Firebase** project :*

The next step is to associate your Android application with your Firebase project. First though, get the **information** you will need. To **connect** Firebase to your Android app, you need to know the **package name** used in your **app**. It's best to have your app open in Android Studio before you start the process.

To connect your Firebase project and your Android app to each other:

1. In Android Studio, open your Android app.
2. Make sure you have the latest **Google Play services** installed.  
Tools > Android SDK Manager > SDK Tools tab > Google Play services.
3. Note the **package** of the **source code** in your Android app.
4. In the Firebase console, Click **Add Firebase to your Android app**.
5. Enter the package name of your app in In the dialog box that appears.

# Firestore Analytics

You can enable Firestore Analytics in your app to see data about **how** and **where** your app is used. You will be able to see

data such as **how many people use your app** over time and **where in the world** they use it. All the data that your app sends to Firestore is **anonymized**, so you **never see the actual identity** of who used your app.

To get usage data about your app, add the **firebase-core library** to your app as follows:

1. Make sure you have added your app to your Firestore project.
2. In Android Studio, make sure you have the latest Google Play services installed.

Tools > Android SDK Manager > SDK Tools tab > Google Play services

3. Add the **dependency for firebase-core** to your **app-level build.gradle (Module: app)** file:

```
compile 'com.google.firebase:firebase-core:x.x.x'
```

After you add the firebase-core library to your app, it will **automatically send usage data** when people use your app.

Without having to add any more code, you will get a default set of data about **how**, **when**, and **where** people use your app.

Not only do you not have to add any code to generate default usage data, you **don't even** have to **publish** your app or do

anything else to it other than use it.

When someone does something in your app, such as **click a button** or **go to another activity**, the app will log an Analytics event. This means that the app will **package up the data about the event** that happened, and put it in the **queue** to send to Firestore.

# References/Bibliography

Text Book :

1. Google developer training android developer fundamentals course