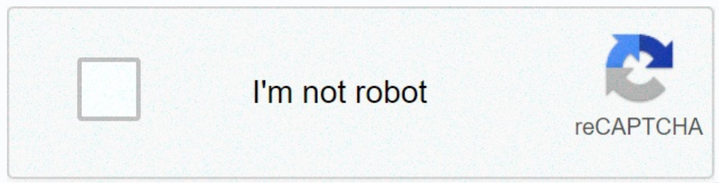

Continue

Intelli studio helpline

Android Video Video Studio is the official Integrated Development Environment (IDE) to develop the Android app, based on IntelliJ Idea. On top of IntelliJ's powerful code editor and developer tool, Android Studio offers even more features that increase your productivity when building Android apps, such as: flexible Grad-based fast simulator system build and unified environment-rich feature where you can apply changes for all Android devices to push code and resource changes to your running app without restarting the template Your application code and GitHub integration to help you build common application features and import extensive sample code develop testing tools and linette tool frameworks to get performance, usability, version compatibility, and other C++ and NDK support problems built-in support for Google's cloud platform, making it easy to integrate Google Cloud Messaging and the program engine this page provides an introduction to the basic features of Android Studio. For a summary of the latest changes, look at android studio release notes. Project structure Figure 1. Project files in android view. Each project in Android Studio includes one or more modules with source code files and resource files. A variety of modules include: Android app modules library modules google app engine modules by default, Android Studio displays your project files in the Android project view, as shown in Figure 1. This view is organized by modules to provide quick access to your project's key source files. All high-level build files are visible under Gradle Scripts, and each application module includes the following folders: manifests: contains the AndroidManifest file.xml. Java: Includes Java source code files, including JUnit test code. res: Includes all non-code sources, such as XML layouts, UI strings, and bitmap images. The structure of the Android project on disk is different from this flattened representation. To see the actual structure of the project file, select the project from the Project drop-down (in Figure 1, it shows as Android). You can also customize the display of project files to focus on certain aspects of your app development. For example, choosing a view of your project problems will display links to source files containing any known coding and syntax errors, such as a missing XML element closing tag in an layout file. Figure 2. Project files in displaying problems, showing a layout file with a problem. For more information, see Project Overview. The interface of the main Android Studio window is made up of several logical regions identified in Figure 3. Figure 3. Main window of Android Studio. Toolbar allows you to perform a wide range of actions, including running your app and launching Android tools. The Navigation Pane helps you navigate through your project and open files for editing. It provides a more compact view of the structure visible in Window. The editor window is where you create and change the code. Depending on the current file type, the editor can change. For example, when viewing an layout file, it displays the Layout Editor. The Tool window bar runs around outside the IDE window and includes buttons that allow you to expand or collapse individual tool windows. Tool windows give you access to specific tasks such as project management, search, version control, and more. You can spread them and collapse them. Displays your project status bar and IDE itself as well as any warnings or messages. You can organize the main window to give yourself more screen space by hiding or moving toolbars and utility windows. You can also use keyboard shortcuts to access most IDE features. A time, you can search across your source code, database, actions, UI elements, and the like, by double-pressing the shift key, or clicking the magnifying glass in the top right corner of the Android Studio window. This can be very useful if, for example, you are trying to find a specific IDE action that you have forgotten how to trigger. Instead of using preset views, Windows Tools is looking for your field and automatically brings up the corresponding tool windows as you work. By default, the most common tool windows used are pinned to the toolbar at the edges of the app window. To expand or collapse the tool window, click the tool name on the tool window bar. You can also drag, pin, open, attach and uncut utility windows. To go back to the current default tool window layout, click the window > restore the default layout, or customize your default layout by clicking the window > Save current layout as default. To show or hide the entire toolbar window, click the window icon in the bottom left corner of the Studio Android window. To find a specific tool window, hover over the window icon and select the Tool window from the menu. You can also use keyboard shortcuts to open utility windows. Table 1 lists shortcuts to the most common windows. Table 1. Keyboard shortcuts are useful for some tool windows. Windows Tools Window and Linux Mac Project Alt+1 Command+1 Control Version Alt+9 Command+9 Run Shift+F10 Control+R Debugging Shift+F9 Control+D Logcat Alt+6 Command+6 Back to Esc Esc Editor Hide All Windows Control Tools+Shift+F12 Command+Shift+F12 If you want to hide all toolbars, Tool window, and editor tabs, click View > Enter Distraction Mode free. This enables a free mode of distraction. To exit distraction mode, click Show > Free Mode exit distraction. You can use search speeds to search and filter in most tool windows in Android Studio. To use Speed Search, select the tools window and then type your search query. For help See keyboard shortcuts. Android Studio's completion code has three types of code completion, code, You can access using the keyboard shortcut. Table 2. Keyboard shortcuts to complete the code. Descriptions of Windows and Linux Mac Basics complete the view of basic suggestions for variables, types, methods, phrases, and the like. If you call the base completion twice in a while, you'll see more results, including private members and non-imported static members. Space Control+Intelligent Space Completion displays the corresponding options by context. Intelligent completion is aware of the expected type and flow of data. If you call Smart Completion twice in a while, you'll see more results, including chains. Control+Shift+Space Control+Shift+Space Statement Completes the current statement for you, adding missing parentheses, brackets, braces, formatting, etc. Control+Shift+Enter Shift+Command+Enter You can also perform quick fixes and show intention actions by pressing Alt+Enter. Find sample code The Code Sample Browser in Android Studio helps you find high-quality, Google-provided Android code samples based on the currently highlighted symbol in your project. For more information, look at finding the sample code. Here are some tips to help you move around Android Studios. Switch between recently accessed files using the recent file action. Press Control +E (Command + E on Mac) to bring up the recent file action. By default, the last accessed file is selected. You can also access any utility window through the left column in this action. View the current file structure using the File Structure action. Create the file structure action by pressing Control+F12 (Command+F12 on a Mac). With this action, you can quickly navigate to any part of your current file. Search and navigate to a specific class in your project using the move to class action. Bring up the action by pressing Control+N (Command+O on a Mac). Moving to class supports complex expressions, including camel humps, paths, line-to-line motion, middle name matching, and many more. If you call it twice in a burst, the results will show you outside of the project classes. Navigate to a file or folder using scrolling. Suggest file navigation by pressing Control+Shift+N (Command+Shift+O on a Mac). To search folders instead of files, add one/at the end of your expression. Navigate to method or field by name using move to action symbol. Suggest the action of scrolling to the icon by pressing Ctrl+Shift+Alt+N (command+option+O on a Mac). Find all the components of the class reference code, method, field, parameter, or statement in the current pointer position by pressing Alt +F7 (option +F7 on Mac). Style and formatting as you edit, Android Studio automatically applies the formatting and styles specified in your code style settings. You can customize code style settings with the programming language, including specifying conventions for tabs and densities, spaces, And handcuffs, and empty lines. To customize code style settings, tap File Settings > > Editor > Style Code (Android Studio > Preferences > Editor > Code Style on a Mac.) Although IDE automatically applies formatting as you work, you can also explicitly call Action Code Reformat by pressing Ctrl+Alt+L (Opt+Command+L on Mac), or auto-indent all lines by pressing Ctrl +Alt+ I (Control+Option+I on Mac). Figure 4. Code before formatting. Figure 5. Code after formatting. Basics Control Version Android Studio supports a variety of version control systems (VCS), including Git, GitHub, CVS, Mercurial, Subversion, and Google's cloud source repositories. After importing your app into Android Studio, use the Android VCS Studio menu options to enable VCS support for the desired version control system, create a repository, import new files into version controls, and perform control operations for other versions: from the Android Studio VCS menu, click Enable Integration Control Version. From the drop-down menu, select a version control system to link to the project root, and then click OK. The VCS menu now displays a number of version control options based on the system you selected. Note: You can also use the Control Menu option > > to adjust and change the version control settings. Gradle Build Android Studio system uses Gradle as the foundation of the build system, with more android-specific functionality provided by android plugins for gredel. The build system runs as an integrated tool from the Android Studio menu, and independently of the command line. You can use the build system features to perform the following: customize, configure, and expand the build process. Create multiple APKs for your app, with different features using the same project and module. Reuse code and resources across resources. By applying gradiol flexibility, you can achieve all of this without modifying your program's main source files. Android Studio file making is named build.gradle. They have simple text files that use Groovy syntax to configure build with elements provided by android plugins for gredel. Each project has a high level build file for the entire project and separate file build level modules for each module. When you enter an existing project, Android Studio automatically generates the necessary build files. To learn more about the build system and how to configure, look at your build configuration. Building a species making system can help you create different versions of the same program from a single project. This is useful when you have both a free version and a paid version of your app, or if you want to distribute multiple APKs for different device settings on Google Play. To learn more about configuring build species, look at the configuration of build species. Multi APK Multi APK Support Allows you to efficiently create multiple APKs based on screen density or ABI. For example, you can create separate APKs from an app for hdpi and mdpi screen density, while still considering them a single type and allowing them to share apk, javac, dx, and ProGuard test settings. Read on to learn more about multi-APK support, build a few APKs. Shrinking resource shrinkage resources in Android Studio automatically removes unsealed resources from your packaged app and library dependencies. For example, if your app uses Google Play services to access the Google Drive feature, and you're not currently using Google Sign-In, then shrinking resources can remove the various drawable assets of the TheSignInButton buttons. Note: Shrinking resources works along with code shrinking tools, such as ProGuard. To learn more about downs downselow codes and resources, look at downs downseing your code and resources. Dependency Management for your project is marked with the name in the build.gradle file. Gradelle cares to find your affiliations and make them available in your build. You can declare module dependency, remote binary dependency, and local binary dependencies in your build.gradle file. Android Studio configures projects to use Maven's central repository by default. (This configuration is included in the high-level build file for the project.) For more information about configuring dependencies, read Add Build Dependencies. Android Studio helps you in debugging and improving your code

performance, including inline debugging and performance analysis tools. Debugging line uses debugging in line to enhance your code walking through in viewing debugging by verifying the line of resources, phrases, and variable values. Inline debugging information includes: variable Inline referencing objects that reference object selection method returning Lambda values and Tooltip operator expressions figure 6 values. A lined variable value. To enable debugging in line, in the Debugging window, click Settings and select the check box to show inline values. Android Studio performance profile provides performance profiles so you can more easily track your app memory and use the processor, find traded objects, find memory leaks, optimize graphical performance, and analyze network requests. Open the Android Profiler tab with the app running on a device or emulator. To learn more about performance profiles, look at performance profile tools. Stack dumps when you are profiled using memory in Android Studio, you can simultaneously start collecting garbage and dumping java stacks into snapshot stacks in android-specific HPROF binary format file. HPROF viewer displays classes, instances of each class, and a reference tree to track you memory usage and find memory leaks Slow. To learn more about work Stack dumps, see inspect stacks and allotments. Your memory indexer can use the memory indexer to track memory allocation and watch where objects are assigned when you perform some actions. Knowing these allocations enables you to optimize the performance of the app and the use of your memory by adjusting the calls the method relates to. For information on tracking and allocation analysis, look at stack inspections and allocations. The data file accesses Android TV tools, such as Systrace, and logcat, generating performance and data debugging for detailed app analysis. To view existing generated data files, open the Captures tool window. In the list of generated files, double-click a file to view the data. Right-click on any .hprof file to convert them to the standard check file format using your RAM. Code Inspection Whenever you compile your app, Android Studio automatically runs the Linette configuration and other IDE inspections to help you easily identify and correct problems with the structural quality of your code. The Lynette tool reviews your Android project source files for potential bugs and improves optimization for integrity, security, performance, usability, accessibility, and internationalization. figure 7 . Lynette inspection results in Android Studio. In addition to Check Lynette, Android Studios also conducts IntelliJ code inspections and validation to simplify its coding workflow. For more information, look at improving your code with Lynette checks. Annotations in Android Studio Android Studio supports annotations for variables, parameters, and return values to help you catch bugs, such as null pointer exceptions and resource type conflicts. Android Waiting Manager Support Packages - Annotations Library in android support repository for use with Android Studio. Android Studio validation configured during code inspection. For more information about Android annotations, see Improve Code Inspection with Annotations. When you build and run your app with Android Studio, you can see messages from the ADB output log and device in the Logcat window. Function profile If you want to profile your app's CPU, memory and network performance, open the Android profile by clicking View > Windows Tool > Android Profiler. Logging into your developer account you can sign up to your developer account in Android Studio to access additional tools that require authentication, such as cloud tools for Android studio and app test tool actions. By logging in, it allows those tools to view and manage their data across Google services. After opening a project in Android Studio, you can sign in to your developer account or switch developer accounts, as follows: Click the profile icon at the end of the toolbar, as shown in Figure 8. Figure 8. Click the profile icon at the end of the toolbar to sign In the window that appears, do one of the following: If you haven't registered yet, click Sign In and allow Android Studio access to the services listed. If you're already signed in, click Add account to sign in to another Google Account. Alternatively, you can click Sign Out and repeat the previous steps to sign in to another account. Account.