

SUMMER TRAINING / INTERNSHIP REPORT



AUGUST – 2017

SUBMITTED BY

SIDDHANT JAIN

60596403115

Department of Information and Technology
MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY
SECTOR-22, PSP AREA, ROHINI, DELHI 110086

ACKNOWLEDGEMENT

It is a great pleasure for me to present this report on my Summer Training and Internship at **Streetingo**, where I gained experience in the field of **Android Development**.

I gained a lot of experience in the field of Android Development and programming skills such as Java, XML and combining these two programming languages in Android Studio to develop fully functional Android Applications.

The internship opportunity I had with **Streetingo** was a great chance for learning and professional development. Therefore, I consider myself as a very lucky individual as I was provided with an opportunity to be a part of it. I am also grateful for having a chance to meet so many wonderful people and professionals who led me through this internship period.

Bearing in mind previous I am using this opportunity to express my deepest gratitude and special thanks to the **Mr. Preyeshu Garg (CTO, Streetingo)** who in spite of being extraordinarily busy with his duties, took time out to hear, guide and keep me on the correct path and allowing me to carry out my project at their esteemed organization and extending during the training.

Also, I express my deepest thanks to my mentor **Mr. Rob Perceival (CEO, Cambridge Certification Authority)** for taking part in useful decisions & giving necessary advices and knowledge for Android Development. I choose this moment to acknowledge his contribution gratefully.

I perceive this opportunity as a big milestone in my career development. I will strive to use theses gained skills and knowledge in the best possible way, and will continue to work on their improvement, in order to attain desired career objectives. Hope to continue cooperation with all of you in the future.

Sincerely,

Siddhant Jain

B.Tech (IT), Third Year

60596403115

MAIT, Delhi

DECLARATION

I hereby declare that the project entitled “**Forex Convertor/Savari NCR**” submitted for “**Summer Training/ Internship Report**” to Maharaja Agrasen Institute of Technology is a record of an original work done by me under the guidance of **Mr. Preyeshu Garg, CTO, Streetingo**, and this project is submitted in the partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Information and Technology.

Siddhant Jain

B.Tech (IT), Third Year

60596403115

MAIT, Delhi

PREFACE

This is the Summer Training/ Internship report of the project “**Forex Convertor/ Savari NCR**” developed by Siddhant Jain, pre-final year student of Bachelor of Technology in Information and Technology Engineering at Maharaja Agrasen Institute Of Technology, Delhi, during his training at **Streetingo** .

The aim of the training is to make the trainee gain professional skills and experience in development of Android Applications.

The aim of the project “**Forex Convertor**” is to present users an application that converts any currency to another currency with the help of real time data. The application offers the following functions:

- Convert Currency to another currency
- Provide real time updated data
- Present users with a representation of historic exchange rates and stock analysis

The aim of the project “**Savari NCR**” is to present the users with a location based cab service application. The application offers the following functions

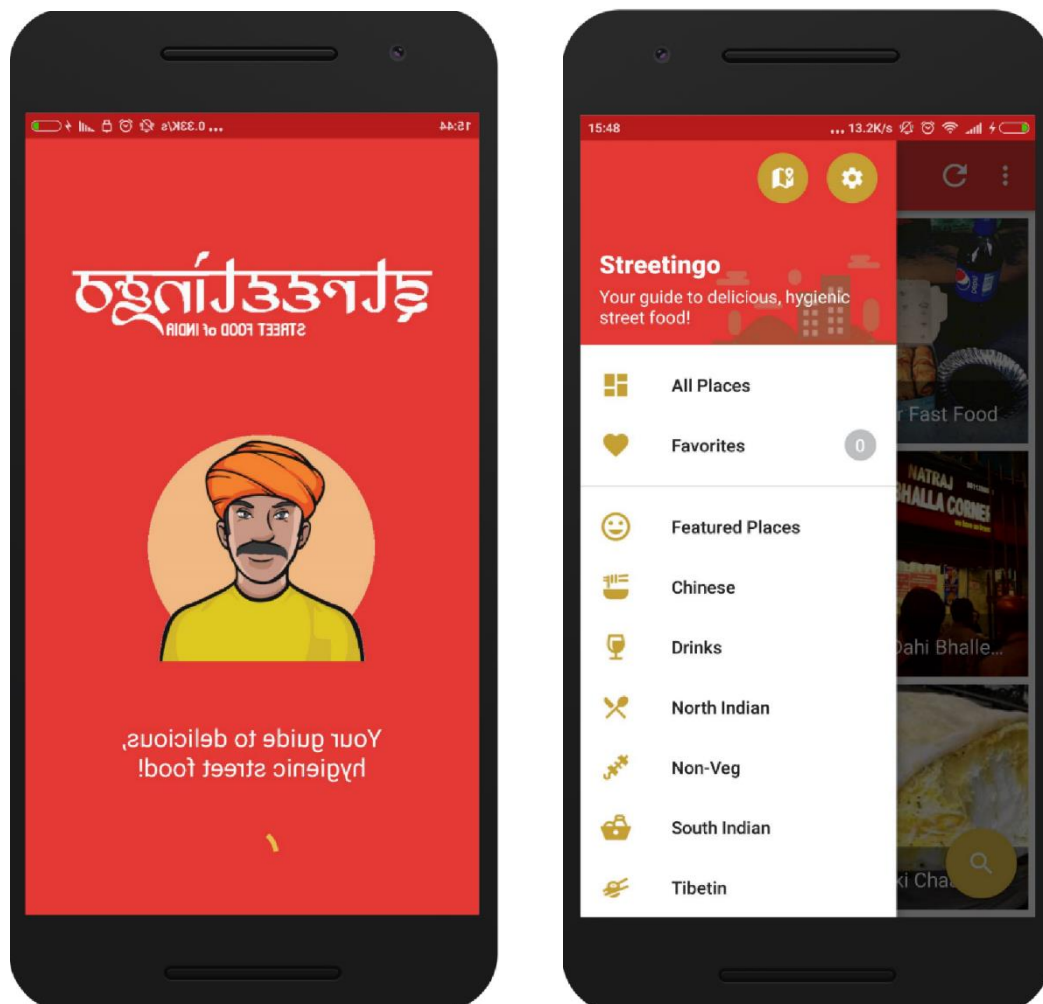
- Separate activities for a rider and a driver
- Uses an Amazon Web Services based Parse Server to store Driver and Rider data
- Allows the user to request for a cab
- Allows the driver to accept the cab request and get the user’s location to offer a cab

Organization Introduction



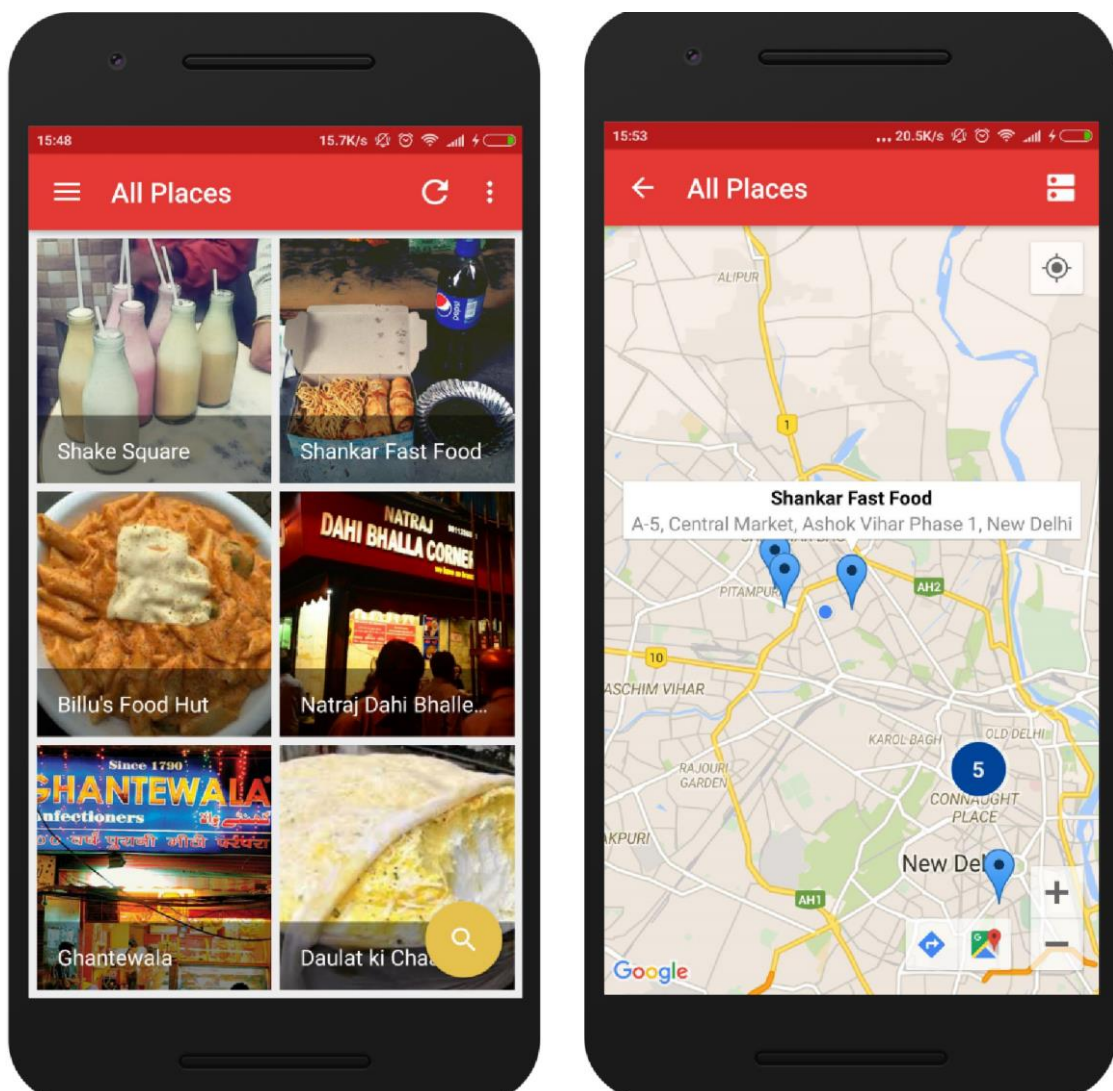
Streetingo is the way to search for hygienic, delicious street food in India. With Streetingo, you can locate the best street food joints near your current location of various categories like North Indian, South Indian, Mexican, Italian, Fast-Food, Chinese etc.

With Streetingo you can easily get hygiene rating which is determined by various factors which include in-depth analysis of the kitchen of each vendor, done extensively by the Streetingo Team. We also weigh in factors like using of disposable cutlery by the vendor, use of safety nets, pH of water used for washing and cooking, ingredient analysis etc.



Features

- Category wise listing of Best Street Food available around you.
- Various ratings on Hygiene, Taste, Service and Timing.
- Equipped with GPRS navigation, it helps people in zeroing in on the exact location of street food outlets.
- Information about food taste quality and approximate pricing is also available in the application.



LIST OF TABLES

- **Table 1 :** System Requirements for Android Studio Version 2.x
- **Table 2 :** System Requirements for Android Studio Version 1.x
- **Table 3 :** Android Studio keyboard shortcuts to open tool windows.
- **Table 4 :** Keyboard shortcuts to access types of code completion

LIST OF FIGURES

- **Figure 1** : Android's latest release, Android 8.0 : Oreo, screenshot
- **Figure 2** : Older versions of Android released by Google
- **Figure 3** : Recent in-use versions of Android released by Google
- **Figure 4** : Logo of Android's latest release, Android Oreo
- **Figure 5** : Android Studio homepage screenshot
- **Figure 6** : Android Studio Projects pane
- **Figure 7** : AndroidManifest.xml
- **Figure 8** : Android Studio main window
- **Figure 9** : Code before formatting
- **Figure 10** : Code after formatting
- **Figure 11** : An inline variable value
- **Figure 12** : Result of Lint inspection in Android Studio
- **Figure 13** : Forex Convertor – Application overview
- **Figure 14** : Application Class Diagram for Forex Convertor
- **Figure 15** : Forex Convertor – Home Screen
- **Figure 16** : Spinner drop down items
- **Figure 17** : Buttons
- **Figure 18** : Forex Convertor – Convertor Interface Screen
- **Figure 19** : Forex Convertor – Exchange Rate Screen
- **Figure 20** : Cab Service Application – Savari overview
- **Figure 21** : Switch Button implementation
- **Figure 22** : Savari – Rider's Activity
- **Figure 23** : Savari – Google Maps Activity
- **Figure 24** : Savari – Driver's Activity
- **Figure 25** : Amazon Web Services (AWS) Marketplace
- **Figure 26** : Parse Server Data Home Screen

CONTENTS

1. Android

1.1. Android Operating System

1.2. Features in Android

1.2.1. General

1.2.2. Connectivity

1.2.3. Media

1.2.4. Hardware Support

1.2.5. Java Support

1.2.6. Handset Layouts

1.2.7. Storage

1.3. Version of Android

2. Android Studio

2.1. System Requirements

2.1.1. Version 2.x

2.1.2. Version 1.x

2.2. Android Studio Versions

2.3. Project Structure

2.4. The User Interface

2.5. Tool Windows

2.6. Code Completion

2.7. Find Sample Code

2.8. Navigation

2.9. Style and Formatting

2.10. Version Control Basics

2.11. Gradle Build System

2.11.1. Build Variants

2.11.2. Multiple APK Support

2.11.3. Resource Shrinking

2.11.4. Managing Dependencies

2.12. Debug and Profile Tools

2.12.1. Inline Debugging

2.12.2. Performance Monitors

2.12.3. Heap Dump

2.12.4. Allocation Tracker

- 2.12.5. Data File Access
- 2.12.6. Code Inspections
- 2.12.7. Annotations in Android Studio
- 2.12.8. Log Messages

3. Android Studio Projects

3.1. Forex Convertor (Currency Convertor Application)

- 3.1.1. Introduction
- 3.1.2. Application Objective
- 3.1.3. Requirements
- 3.1.4. Interface Implementation
 - 3.1.4.1. Splash Screen
 - 3.1.4.2. Convertor Screen
- 3.1.5. API
- 3.1.6. Spinner Method
- 3.1.7. Buttons Method
- 3.1.8. Convertor Method
- 3.1.9. JSON Method
- 3.1.10. Convertor Interface
- 3.1.11. Exchange Rate Screen
 - 3.1.11.1. Custom Text View
 - 3.1.11.2. Graphs Implementation

3.2. Savari NCR (Location Based Cab Service Application)

- 3.2.1. Application Overview
 - 3.2.1.1. What is Savari?
- 3.2.2. Application Interface
- 3.2.3. Mai Menu
- 3.2.4. Parse Server
- 3.2.5. Switch Implementation
- 3.2.6. Rider's Activity
- 3.2.7. Google Maps Activity (API)
- 3.2.8. Driver's Activity
- 3.2.9. Amazon Web Services (AWS)
- 3.2.10. AWS Parse Server

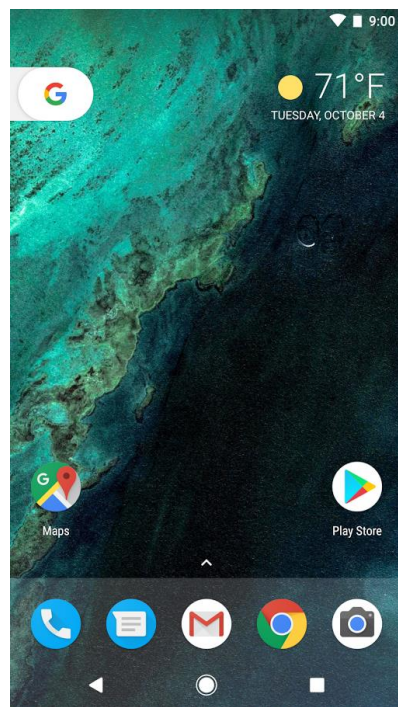
ANDROID

Android Operating System

- Android is a mobile operating system developed by Google, based on the **Linux Kernel** and designed primarily for touchscreen mobile devices such as smartphones and tablets. Android's user interface is mainly based on direct manipulation, using touch gestures that loosely correspond to real-world actions, such as swiping, tapping and pinching, to manipulate on-screen objects, along with a virtual keyboard for text input. In addition to touchscreen devices, Google has further developed Android TV for televisions, Android Auto for cars, and Android Wear for wrist watches, each with a specialized user interface. Variants of Android are also used on game consoles, digital cameras, PCs and other electronics.

- Initially developed by **Android Inc.**, which Google bought in 2005, Android was

Figure 1: Android's latest release – Android 8.0 Oreo.



unveiled in 2007, along with the founding of the Open Handset Alliance – a consortium of hardware, software, and telecommunication companies devoted to advancing open standards for mobile devices. Beginning with the first commercial Android device in September 2008, the operating system has gone through multiple major releases, with the current version being **8.0 "Oreo"**, released in August 2017. Android applications ("apps") can be downloaded from the Google Play store, which features over 2.7 million apps as of February 2017. Android has been the best-selling OS on

tablets since 2013, and runs on the vast majority of smartphones. As of May 2017, Android has **two billion monthly active users**, and it has the largest installed base of any operating system.

- Android's source code is released by Google under an **open source license**, although most Android devices ultimately ship with a combination of free and open

source and proprietary software, including proprietary software required for accessing Google services.

- Android is popular with technology companies that require a ready-made, low-cost and customizable operating system for high-tech devices. Its open nature has encouraged a large community of developers and enthusiasts to use the open-source code as a foundation for community-driven projects, which deliver updates to older devices, add new features for advanced users or bring Android to devices originally shipped with other operating systems. The extensive variation of hardware in Android devices causes significant delays for software upgrades, with new versions of the operating system and security patches typically taking months before reaching consumers, or sometimes not at all. The success of Android has made it a target for patent and copyright litigation between technology companies.

Features in Android

1. General

- **Messaging** : *SMS* and *MMS* are available forms of messaging, including threaded text messaging and Android *Cloud To Device Messaging (C2DM)* and now enhanced version of C2DM, Android *Google Cloud Messaging (GCM)* is also a part of Android Push Messaging services.
- **Web browser**: The web browser available in Android is based on the open-source Blink layout engine, coupled with Chrome's V8 JavaScript engine. The browser is variably known as '*Android Browser*', 'AOSP browser', 'stock browser', 'native browser', and 'default browser'. Starting with Android 4.4 KitKat, Google has mandated that the default browser for Android.
- **Voice-based features**: Google search through voice has been available since initial releases. Voice actions for calling, texting, navigation, etc. are supported on Android 2.2 onwards. As of Android 4.1, Google has expanded Voice Actions with ability to talk back and read answers from *Google's Knowledge Graph* when queried with specific commands.
- **Multi-touch**: Android has native support for multi-touch which was initially made available in handsets such as the HTC Hero. The feature was originally disabled at the kernel level. Google has since released an update for the *Nexus One* and the *Motorola Droid* which enables multi-touch natively.

- **Multitasking:** Multitasking of applications, with unique handling of memory allocation, is available.
- **Screen capture:** Android supports capturing a screenshot by pressing the power and home-screen buttons at the same time. Prior to Android 4.0, the only methods of capturing a screenshot were through manufacturer and third-party customizations, or otherwise by using a PC connection (*DDMS developer's tool*).
- **TV recording:** Android TV supports capturing video and replaying it.
- **Video calling:** Android does not support native video calling, but some handsets have a customized version of the operating system that supports it, either via the UMTS network (like the Samsung Galaxy S) or over IP. Video calling through *Google Talk* is available in Android 2.3.4 (Gingerbread) and later. Gingerbread allows Nexus S to place Internet calls with a SIP account. This allows for enhanced VoIP dialling to other SIP accounts and even phone numbers. *Skype 2.1* offers video calling in Android 2.3, including front camera support. Users with the Google+ Android app can video chat with other Google+ users through *Hangouts*.
- **Multiple language support:** Android supports multiple languages.
- **Accessibility:** Built-in *text-to-speech* is provided by *TalkBack* for people with low or no vision. Enhancements for people with hearing difficulties are available, as are other aids.

2. Connectivity

- **Bluetooth:** Supports voice dialling and sending contacts between phones, playing music, sending files (OPP), accessing the phone book (PBAP), A2DP and AVRCP. Keyboard, mouse and joystick (HID) support is available in Android 3.1+, and in earlier versions through manufacturer customizations and third-party applications.
- **Tethering:** Android supports tethering, which allows a phone to be used as a wireless/wired *Wi-Fi hotspot*. Before Android 2.2 this was supported by third-party applications or manufacturer customizations.
- Android also supports connectivity technologies including *GSM/EDGE, LTE, CDMA, EV-DO, UMTS, NFC, IDEN, Wi-Fi* and *WiMAX*

3. Media

- **Streaming media support:** RTP/RTSP streaming (3GPP PSS, ISMA), HTML progressive download (HTML5 <video> tag). Adobe Flash Streaming (RTMP) and HTTP Dynamic Streaming are supported by the *Flash plugin*. Apple HTTP Live Streaming is supported by *RealPlayer for Android*, and by the operating system since Android 3.0 (Honeycomb).
- **Media support:** Android supports the following audio/video/still media formats: WebM, H.263, H.264, AAC, HE-AAC (in 3GP or MP4 container), MPEG-4 SP, AMR, AMR-WB, FLAC, WAV, JPEG, PNG, GIF, BMP, and WebP.
- **External storage:** Most Android devices include *microSD* card slots and can read microSD cards formatted with the FAT32, Ext3 or Ext4 file systems. To allow use of external storage media such as USB flash drives and USB HDDs, some Android devices are packaged with *USB-OTG cables*.

4. **Hardware Support:** Android devices can include still/video cameras, touchscreens, GPS, accelerometers, gyroscopes, barometers, magnetometers, dedicated gaming controls, proximity and pressure sensors, thermometers, accelerated 2D and 3D graphics.

5. **Java Support:** Whilst most Android applications are written in Java, there is no Java Virtual Machine in the platform and Java byte code is not executed. A specialized virtual machine provided by third-party applications designed specifically for Android and optimized for battery-powered mobile devices with limited memory and CPU can be used to compile and execute Java classes.

6. **Handset Layouts:** The platform works for various screen sizes from smartphone sizes to tablet sizes, and can potentially connect to an external screen i.e. through HDMI or wirelessly with Miracast or Google's Chromecast. Portrait and landscape orientations are supported and usually switched between by turning the device.

7. **Storage:** SQLite, a lightweight relational database, is used for data storage purposes.

Versions of Android



Figure 2: Older Versions of Android released by Google.

- **Cupcake:** Version 1.5 (API Level 3)
- **Donut:** Version 1.6 (API Level 4)
- **Éclair:** Version 2.0 – 2.1 (API Level 5 – 7)
- **Froyo:** Version 2.2 – 2.2.3 (API Level 8)
- **Gingerbread:** Version 2.3 – 2.3.7 (API Level 9 - 10)
- **Honeycomb:** Version 3.0 – 3.2.6 (API Level 11 - 13)
- **Ice Cream Sandwich:** Version 4.0 – 4.0.4 (API Level 14 - 15)



Figure 3: Recent in-use versions of Android released by Google.

- **Jelly Bean:** Version 4.1 – 4.3.1 (API Level 16 - 18)
- **KitKat:** Version 4.4 – 4.4.4 (API Level 19 - 20)
- **Lollipop:** Version 5.0 – 5.1.1 (API Level 21 - 22)
- **Marshmallow:** Version 6.0 – 6.0.1 (API Level 23)
- **Nougat:** Version 7.0 – 7.1.2 (API Level 24 - 25)

- **Oreo:** Version 8.0 (API Level 26). The latest version of Android, released on August 21, 2017.



Figure 4: Android's latest release, Version 8.0: Oreo.

Android Studio

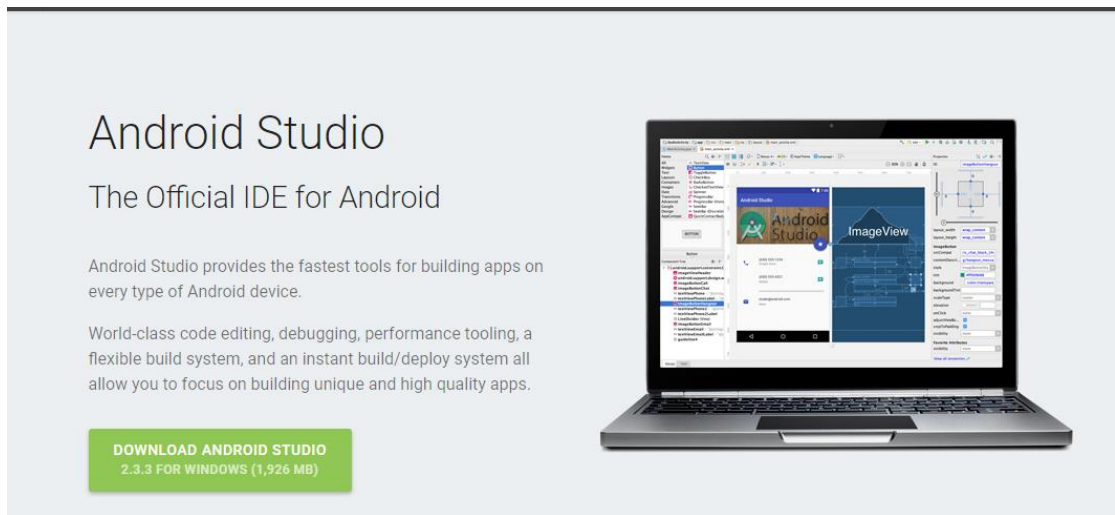


Figure 5: Android Studio homepage screenshot.

Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on **IntelliJ IDEA**. On top of IntelliJ's powerful code editor and developer tools, Android Studio offers even more features that enhance productivity of a developer when building Android apps, such as:

- A flexible Gradle-based build system
- A fast and feature-rich emulator
- A unified environment where you can develop for all Android devices
- Instant Run to push changes to your running app without building a new APK
- Code templates and GitHub integration to help you build common app features and import sample code
- Extensive testing tools and frameworks
- Lint tools to catch performance, usability, version compatibility, and other problems
- C++ and NDK support

- Built-in support for *Google Cloud Platform*, making it easy to integrate Google Cloud Messaging and App Engine

SYSTEM REQUIREMENTS

Version 2.x

Table 1: System Requirements for Android Studio Version 2.x

| Criterion | Description |
|-------------------|---|
| OS Version | Windows 7 or later Mac OS X 10.9.5 or later GNOME or KDE desktop |
| RAM | 3 GB RAM minimum, 8 GB RAM recommended; plus 1 GB for the Android Emulator. |
| Disk Space | 500 MB disk space for Android Studio, at least 1.5 GB for Android SDK, emulator system images, and caches |
| Java Version | Java Development Kit (JDK) 8 |
| Screen Resolution | 1280x800 minimum screen resolution |

Version 1.x

Table 2: System Requirements for Android Studio Version 1.x

| Criterion | Description |
|------------|---|
| OS Version | Windows XP or later Mac OS X 10.8.5 or later |

| | |
|-----------------------|---|
| | GNOME, KDE or Unity desktop on Ubuntu or Fedora or GNU/Linux Debian |
| RAM | 3 GB RAM minimum, 4 GB RAM recommended |
| Disk Space | 500 MB disk space |
| Space for Android SDK | At least 1 GB for Android SDK, emulator system images, and caches |
| Java Version | Java Development Kit (JDK) 8 |
| Screen Resolution | 1280x800 minimum screen resolution |

Android Studio versions

1. 0.8.14 - October 2014
2. 1.0 - December 2014
3. 1.0.1 - December 2014
4. 1.1.0 - February 2015
5. 1.2.1 - May 2015
6. 1.2.2 - June 2015
7. 1.3.0 - July 2015
8. 1.3.1 - August 2015
9. 1.4.0 - September 2015
10. 2.1.0 - August 2016
11. 2.2.0 - September 2016
12. 2.2.3 - December 2016
13. 2.3.0 - March 2017
14. 2.3.1 - April 2017
15. 2.3.2 - April 2017
16. 2.3.3 - June 2017

Google released its latest version of Android Studio on August 21, 2017 i.e. **Android Studio 3.0 Beta**

Project Structure

Each project in Android Studio contains one or more modules with source code files and resource files. Types of modules include:

- Android app modules
- Library modules
- Google App Engine modules

By default, Android Studio displays the project files in the Android project view. This view is organized by modules to provide quick access to the project's key source files.

All the build files are visible at the top level under *Gradle Scripts* and each app module contains the following folders:

- **Manifests:**
Contains the AndroidManifest.xml file.
- **Java:** Contains the Java source code files, including JUnit test code.
- **Res:** Contains all non-code resources, such as XML layouts, UI strings, and bitmap images.

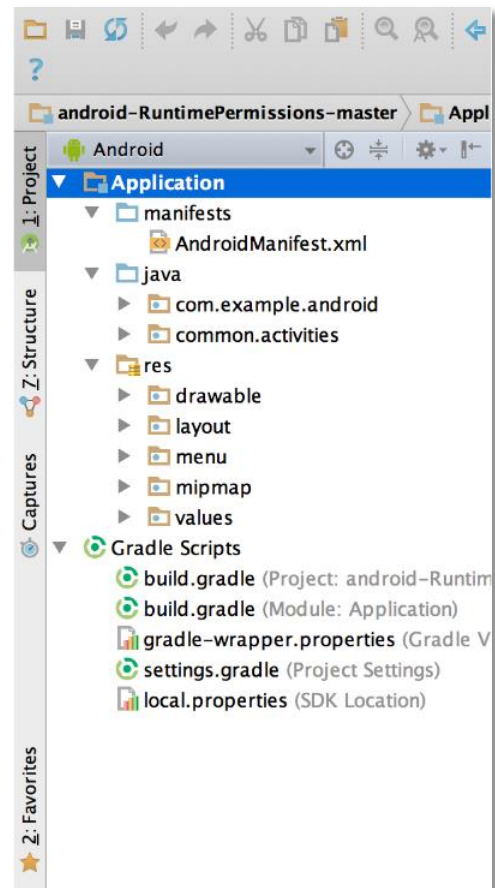


Figure 6: Android Studio Project pane.

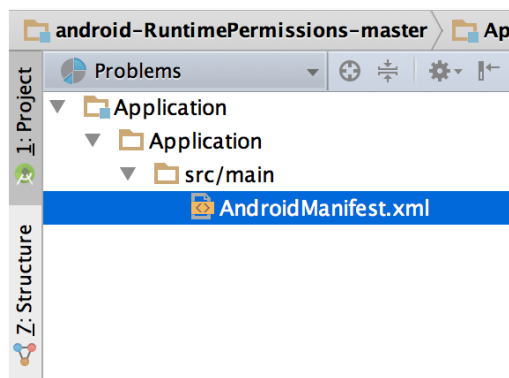


Figure 7: File containing AndroidManifest.xml

The Android project structure on disk differs from this flattened representation. To see the actual file structure of the project, select *Project* from the *Project dropdown*.

The view of the project files can also be customized to focus on specific aspects of your app development.

The User Interface

The Android Studio main window is made up of several logical areas.

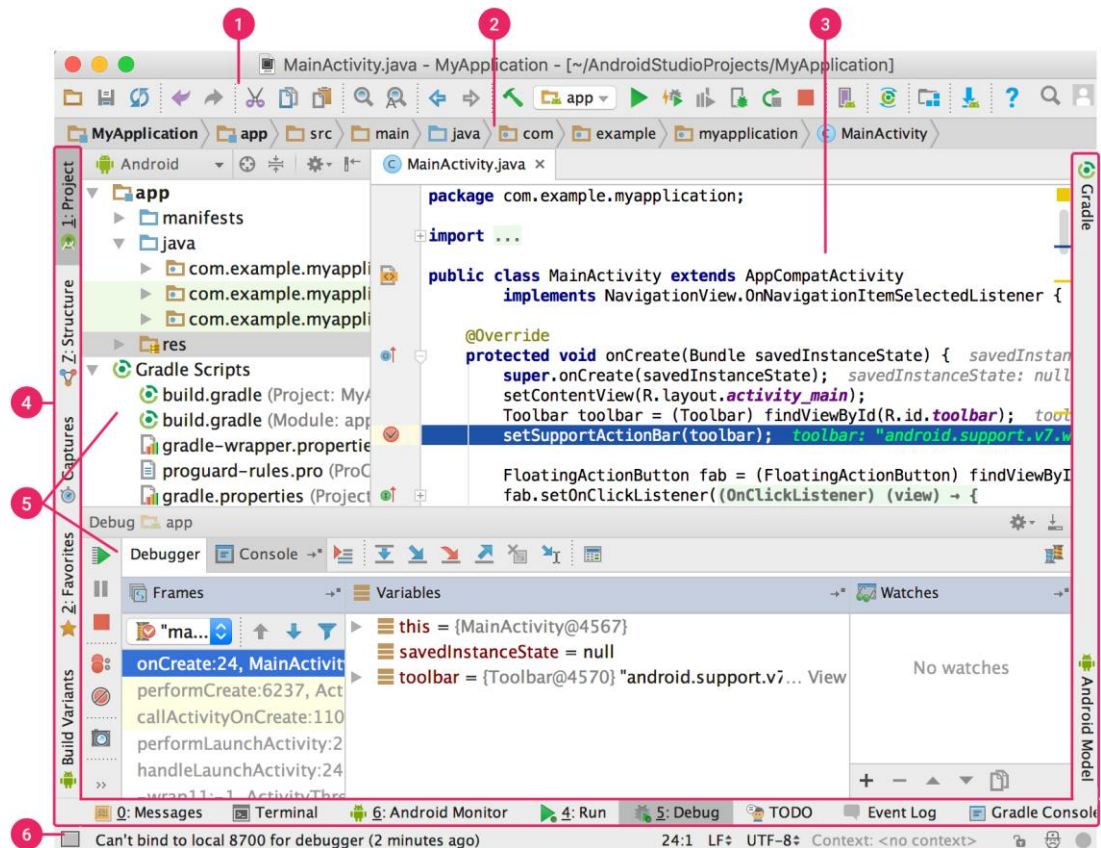


Figure 8: Android Studio main window.

1. The **toolbar** lets a developer carry out a wide range of actions, including running the app and launching Android tools.
2. The **navigation bar** helps the developer navigate through the project and open files for editing. It provides a more compact view of the structure visible in the **Project** window.
3. The **editor window** is where a developer can create and modify code. Depending on the current file type, the editor can change. For example, when viewing a layout file, the editor displays the Layout Editor.
4. The **tool window bar** runs around the outside of the IDE window and contains the buttons that allows the developer to expand or collapse individual tool windows.
5. The **tool windows** gives the developer access to specific tasks like project management, search, version control, and more. You can expand them and collapse them.

6. The **status bar** displays the status of your project and the 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 tool windows. You can also use keyboard shortcuts to access most IDE features.

At any time, you can search across your source code, databases, actions, elements of the user interface, and so on, by double-pressing the Shift key, or clicking the magnifying glass in the upper right-hand corner of the Android Studio window. This can be very useful if, for example, you are trying to locate a particular IDE action that you have forgotten how to trigger.

Tool Windows

Instead of using preset perspectives, Android Studio follows your context and automatically brings up relevant tool windows as you work. By default, the most commonly used tool windows are pinned to the tool window bar at the edges of the application window.


- To expand or collapse a tool window, click the tool's name in the tool window bar. You can also drag, pin, unpin, attach, and detach tool windows.
- To return to the current default tool window layout, click **Window > Restore Default Layout** or customize your default layout by clicking **Window > Store Current Layout as Default**.
- To show or hide the entire tool window bar, click the  window icon in the bottom left-hand corner of the Android Studio window.
- To locate a specific tool window, hover over the window icon and select the tool window from the menu.

Table 3: Android Studio keyboard shortcuts to open tool windows.

| Tool Window | Windows and Linux | Mac |
|-----------------------|--------------------------|--------------------------|
| Project | Alt+1 | Command+1 |
| Version Control | Alt+9 | Command+9 |
| Run | Shift+F10 | Control+R |
| Debug | Shift+F9 | Control+D |
| Android Monitor | Alt+6 | Command+6 |
| Return to Editor | Esc | Esc |
| Hide All Tool Windows | Control+Shift+F12 | Command+Shift+F12 |

If you want to hide all toolbars, tool windows, and editor tabs, click **View > Enter Distraction Free Mode**. This enables **Distraction Free Mode**. To exit Distraction Free Mode, click **View > Exit Distraction Free Mode**.

You can use **Speed Search** to search and filter within most tool windows in Android Studio. To use Speed Search, select the tool window and then type your search query.

Code Completion

Android Studio has three types of code completion, which can be accessed using keyboard shortcuts.

| |
|---|
| Table 4: Keyboard shortcuts to access types of code completion |
|---|

| Type | Description | Windows and Linux | Mac |
|------------------|---|----------------------------|----------------------------|
| Basic Completion | Displays basic suggestions for variables, types, methods, expressions, and so on. If you call basic completion twice in a row, you see more results, including private members and non-imported static members. | Control+Space | Control+Space |
| Smart Completion | Displays relevant options based on the context. Smart completion is aware of the expected type and data flows. If you call Smart Completion twice | Control+Shift+Space | Control+Shift+Space |

| | | | |
|----------------------|---|----------------------------|----------------------------|
| | in a row, you see more results, including chains. | | |
| Statement Completion | Completes the current statement for you, adding missing parentheses, brackets, braces, formatting, etc. | Control+Shift+Enter | Shift+Command+Enter |

Find Sample code

The Code Sample Browser in Android Studio helps find high-quality, Google-provided Android code samples based on the currently highlighted symbol in your project.

Navigation

- Switch between your recently accessed files using the *Recent Files* action. Press **Control+E** (**Command+E** on a Mac) to bring up the Recent Files action. By default, the last accessed file is selected. You can also access any tool window through the left column in this action.
- View the structure of the current file using the *File Structure* action. Bring up the File Structure action by pressing **Control+F12** (**Command+F12** on a Mac). Using this action, you can quickly navigate to any part of your current file.
- Search for and navigate to a specific class in your project using the *Navigate to Class* action. Bring up the action by pressing **Control+N** (**Command+O** on a Mac). Navigate to Class supports sophisticated expressions, including camel humps, paths, line navigate to, middle name matching, and many more. If you call it twice in a row, it shows you the results out of the project classes.

- Navigate to a file or folder using the *Navigate to File* action. Bring up the *Navigate to File* action by pressing **Control+Shift+N** (**Command+Shift+O** on a Mac). To search for folders rather than files, add a / at the end of your expression.
- Navigate to a method or field by name using the *Navigate to Symbol* action. Bring up the *Navigate to Symbol* action by pressing **Control+Shift+Alt+N** (**Command+Shift+Alt+O** on a Mac).
- Find all the pieces of code referencing the class, method, field, parameter, or statement at the current cursor position by pressing **Alt+F7**.

Style and Formatting

As you edit, Android Studio automatically applies formatting and styles as specified in your code style settings. You can customize the code style settings by programming language, including specifying conventions for tabs and indents, spaces, wrapping and braces, and blank lines. To customize your code style settings, click **File > Settings > Editor > Code Style** (**Android Studio > Preferences > Editor > Code Style** on a Mac.)

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    mActionBar = getSupportActionBar();
    mActionBar.setDisplayHomeAsUpEnabled(true);
}
```

Figure 9: Code before formatting.

Although the IDE automatically applies formatting as you work, you can also explicitly call the *Reformat Code* action by pressing **Control+Alt+L** (**Opt+Command+L** on a Mac), or auto-indent all lines by pressing **Control+Alt+I** (**Alt+Option+I** on a Mac).

```
} public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    mActionBar = getSupportActionBar();
    mActionBar.setDisplayHomeAsUpEnabled(true);
}

// Get reference to the drawer layout and set event listener
```

Figure 10: Code after formatting.

Version Control Basics

Android Studio supports a variety of version control systems (VCS's), including *Git*, *GitHub*, *CVS*, *Mercurial*, *Subversion*, and *Google Cloud Source Repositories*.

After importing your app into Android Studio, use the Android Studio VCS menu options to enable VCS support for the desired version control system, create a repository, import the new files into version control, and perform other version control operations:

1. From the Android Studio **VCS** menu, click **Enable Version Control Integration**.
2. From the drop-down menu, select a version control system to associate with the project root, and then click **OK**.

The VCS menu now displays a number of version control options based on the system you selected.

Gradle Build System

Android Studio uses Gradle as the foundation of the build system, with more Android-specific capabilities provided by the *Android plugin for Gradle*. This build system runs as an integrated tool from the Android Studio menu, and independently from the command line. You can use the features of the build system to do the following:

- Customize, configure, and extend the build process.
- Create multiple APKs for your app, with different features using the same project and modules.
- Reuse code and resources across sourcesets.

By employing the flexibility of Gradle, you can achieve all of this without modifying your app's core source files. Android Studio build files are named `build.gradle`. They are plain text files that use *Groovy* syntax to configure the build with elements provided by the Android plugin for Gradle. Each project has one top-level build file for the entire project and separate module-level build files for each module. When you import an existing project, Android Studio automatically generates the necessary build files.

Build Variants

The build system can help you create different versions of the same application 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 configurations on Google Play.

Multiple APK Support

Multiple APK support allows you to efficiently create multiple APKs based on screen density or ABI. For example, you can create separate APKs of an app for the hdpi and mdpi screen densities, while still considering them a single variant and allowing them to share test APK, javac, dx, and ProGuard settings.

Resource Shrinking

Resource shrinking in Android Studio automatically removes unused resources from your packaged app and library dependencies. For example, if your application is using *Google Play services* to access Google Drive functionality, and you are not currently using Google Sign-In, then resource shrinking can remove the various drawable assets for theSignInButton buttons.

Managing Dependencies

Dependencies for your project are specified by name in the build.gradle file. Gradle takes care of finding your dependencies and making them available in your build. You can declare module dependencies, remote binary dependencies, and local binary dependencies in your build.gradle file. Android Studio configures projects to use the Maven Central Repository by default. (This configuration is included in the top-level build file for the project.)

Debug and Profile Tools

Android Studio assists you in debugging and improving the performance of your code, including inline debugging and performance analysis tools.

Inline debugging

Use inline debugging to enhance your code walk-throughs in the debugger view with inline verification of references, expressions, and variable values. Inline debug information includes:

- Inline variable values
- Referring objects that reference a selected object
- Method return values
- Lambda and operator expressions

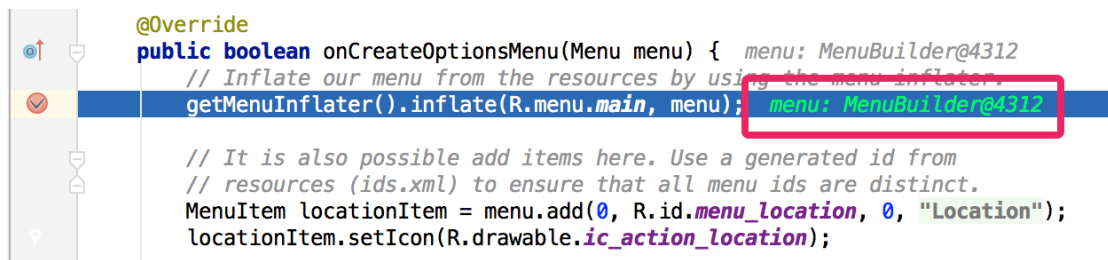


Figure 11: An inline variable value.

To enable inline debugging, in the *Debug* window,  click *Settings* and select the checkbox for *Show Values Inline*.

Performance monitors

Android Studio provides performance monitors so you can more easily track your app's memory and CPU usage, find deallocated objects, locate memory leaks, optimize graphics performance, and analyse network requests. With your app running on a device or emulator, open the *Android Monitor* tool window, and then click the *Monitors* tab.

Heap dump

When you're monitoring memory usage in Android Studio, you can simultaneously initiate garbage collection and dump the Java heap to a heap snapshot in an Android-specific HPROF binary format file. The HPROF viewer displays classes, instances of each class, and a reference tree to help you track memory usage and find memory leaks.

Allocation tracker

Android Studio allows you to track memory allocation as it monitors memory use. Tracking memory allocation allows you to monitor where objects are being allocated when you perform certain actions. Knowing these allocations enables you to optimize your app's performance and memory use by adjusting the method calls related to those actions.

Data file access

The Android SDK tools, such as *Systrace*, *logcat*, and *Traceview*, generate performance and debugging data for detailed app analysis.

To view the available generated data files, open the Captures tool window. In the list of the generated files, double-click a file to view the data. Right-click any .hprof files to convert them to the standard .hprof file format.

Code inspections

Whenever you compile your program, Android Studio automatically runs configured **Lint** and other **IDE inspections** to help you easily identify and correct problems with the structural quality of your code.

The Lint tool checks your Android project source files for potential bugs and optimization improvements for correctness, security, performance, usability, accessibility, and internationalization.

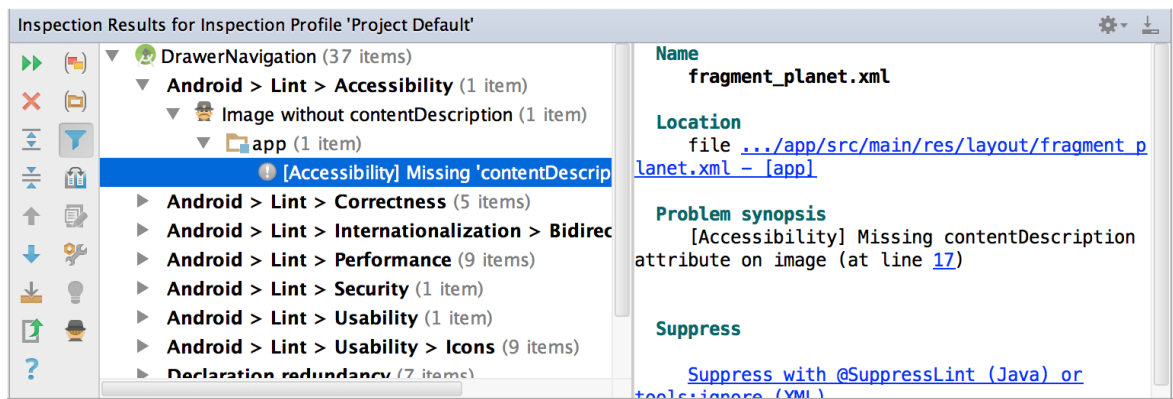


Figure 12: The results of a Lint inspection in Android Studio.

In addition to Lint checks, Android Studio also performs IntelliJ code inspections and validates annotations to streamline your coding workflow.

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. The Android SDK Manager packages the Support-Annotations library in the Android Support Repository for use with Android Studio. Android Studio validates the configured annotations during code inspection.

Log messages

When you build and run your app with Android Studio, you can view adb output and device **log messages (logcat)** by clicking **Android Monitor** at the bottom of the window.

If you want to debug your app with the **Android Device Monitor**, you can launch the Device Monitor by clicking **Tools > Android > Android Device Monitor**.

ANDROID STUDIO PROJECTS

Currency Convertor Application

(FOREX CONVERTOR)



Figure 13: Forex Convertor – Application overview.

➤ Forex Convertor

Introduction

This document shows how the concepts of object oriented design are implemented for a simple application. This example was developed on the Windows operating system using the Objective Java programming language and the Android Studio. The user interface was developed using the Figma, a software designing tool and the rest of the application was developed in Android Studio.

Application Objective

This application is a currency converter, which takes the currency conversion (exchange) rate and the amount of US dollars to be converted to determine the amount of the other currency. The final (and sole) user interface is shown in Figure 1. The first two text fields are entered manually by the User of the application, who then clicks on the Convert button to have the total Amount in Other Currency calculated.

Requirements

Requirements for this application are minimal.

- Presumably, the data entered into the Exchange Rate and Dollars to Convert fields will be either integer or real (floating point) values.
- Clicking on the Convert button fills the answer in the Amount in Other Currency field.

Application Class Diagram for Currency Converter

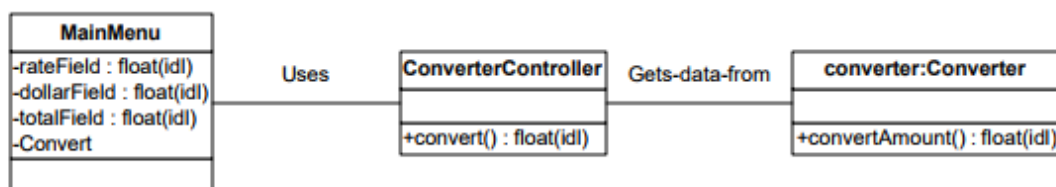


Figure 14: Application Class Diagram for Forex Convertor

Interface Implementation

➤ Splash Screen

The main screen appearing in the application is a simple splash screen which is a simple method to load up data for the application in the background while the user exists over the loading screen. The Splash Screen consists of the following content:

1. On Timer Implementation
2. Intent Implementation
3. Gradle Implementation
4. On Click Implementation



Figure 15: Forex Convertor Home Screen

➤ **Convertor Screen**

The main screen for the application the Forex Convertor is the Convertor Screen. Here the user is met with different options to choose from as the application is connected to an online API which refreshes its database every 10 minutes giving the exact conversion rate for the user to access.

API

API Level is an integer value that uniquely identifies the framework API revision offered by a version of the Android platform. The Android platform provides a framework API that applications can use to interact with the underlying Android system. The framework API consists of: A core set of packages and classes.

Spinner Method

Spinners provide a quick way to select one value from a set. In the default state, a spinner shows its currently selected value. Touching the spinner displays a dropdown menu with all other available values, from which the user can select a new one.

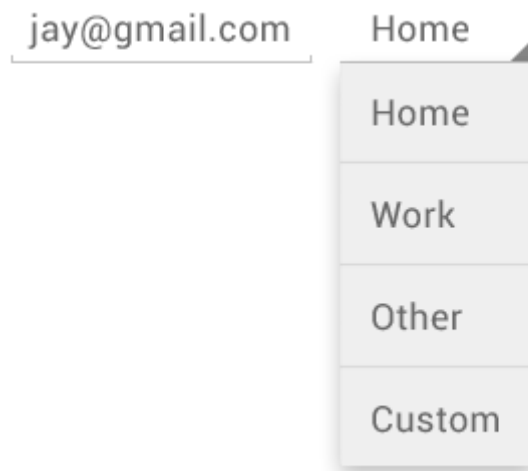


Figure 16: Spinner drop down items

Buttons Method

A button consists of text or an icon (or both text and an icon) that communicates what action occurs when the user touches it.



Figure 17: Buttons

Convertor Method

The user has to enter a desired amount that is to be converted to his / her desired currency. A default currency is set up initially to calculate with the help of API and JSON implementation.

JSON Method

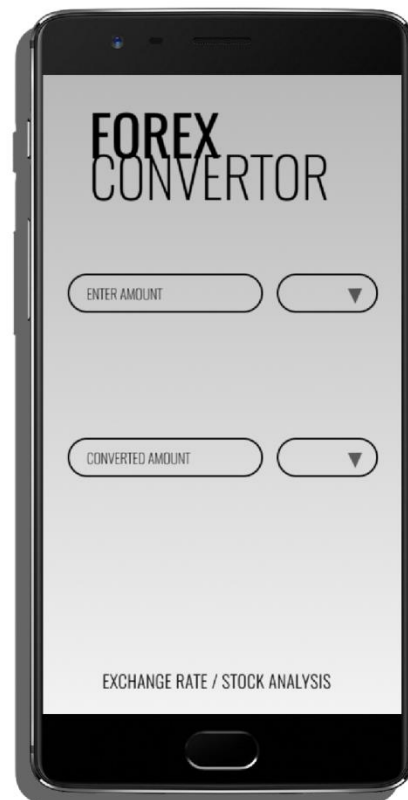
The JSON object contains methods for parsing JavaScript Object Notation (JSON) and converting values to JSON. It can't be called or constructed, and aside from its two method properties it has no interesting functionality of its own. SON is syntax for serializing objects, arrays, numbers, strings, Booleans, and null. It is based upon JavaScript syntax but is distinct from it: some JavaScript is not JSON, and some JSON is not JavaScript. See also JSON: The JavaScript subset that isn't.

Convertor Interface

The final interface that the user uses is as follows which is pretty simple and materialized. It consist of the following elements to make up the screen:

1. Text View
2. Edit Text
3. Spinner View
4. Image View

Figure 18: Forex Convertor
– convertor interface screen.



➤ Exchange Rate Screen

In the exchange rate screen the user is given some more information if needed on the same conversion rate they used in the previous screen. This activity also depicts graphs and inflations in the currency rate for certain amount of time depending on the JSON data it is provided through API of the source.

User can save their default settings and also view data through external links provided in the Activity.

CUSTOM TEXT VIEW

One thing that always bothered me about Android's Text View widget is that you couldn't specify a custom font in the XML layout. It's not a lot of work to load and apply a font in code, but after finding myself pounding out the same few lines of code on every project, I thought there had to be a better way.

GRAPHS IMPLEMENTATION

SciChart for Android is a relative newcomer, but brings extremely fast high performance real-time charting to the Android platform.

SciChart is a commercial control but available under royalty free distribution / per developer licensing. There is also free licensing available for educational use with some conditions.

Figure 19: Forex Converter
– Exchange Rate screen.



LOCATION BASED CAB SERVICE APPLICATION
(SAVARI NCR)



Figure 20: Cab Service App – Savari overview

APPLICATION OVERVIEW

WHAT IS SAVARI?

To customers, SAVARI is fundamentally synonymous with taxis, and to drivers, it's mainly a recommendation service. The Android app associates riders with drivers using their phone's GPS aptitudes, hire both parties know one another's locality and eliminating the question of when the ride will actually reach. In addition, the tech company also practices all payments involved, charging the passenger's credit card, taking a cut for itself (which ranges from 5% to 20%), and direct depositing the outstanding money into the driver's account, all in the circumstantial and totally cashless.

APPLICATION INTERFACE

➤ MAIN MENU

The main activity the user appears to is the main activity which offers a choice to the user of selecting its position as a DRIVER or a RIDER. The complete application is linked up with PARSE server to connect different users / RIDERS to the DRIVERS over the internet.

PARSE SERVER

Parse Server is a new project, separate from the hosted Parse API service. Our intention is to provide and support the growth of an open-source API server, and allow new developers to benefit from the powerful Parse client SDKs regardless of where their application logic and data is stored.

Parse Server can be deployed to any infrastructure that can run Node.js. Parse Server works with the Express web application framework. It can be added to existing web applications, or run by itself.

SWITCH IMPLEMENTATION

A Switch is a two-state toggle switch widget that can select between two options. The user may drag the "thumb" back and forth to choose the selected option, or simply tap to toggle as if it were a checkbox. The text property controls the text displayed in the label for the switch, whereas the off and context controls the text on the thumb. Similarly, the text Appearance and the related set Typeface () methods control the typeface and style of label text, whereas the switchTextAppearance and the related setSwitchTypeface() methods control that of the thumb.

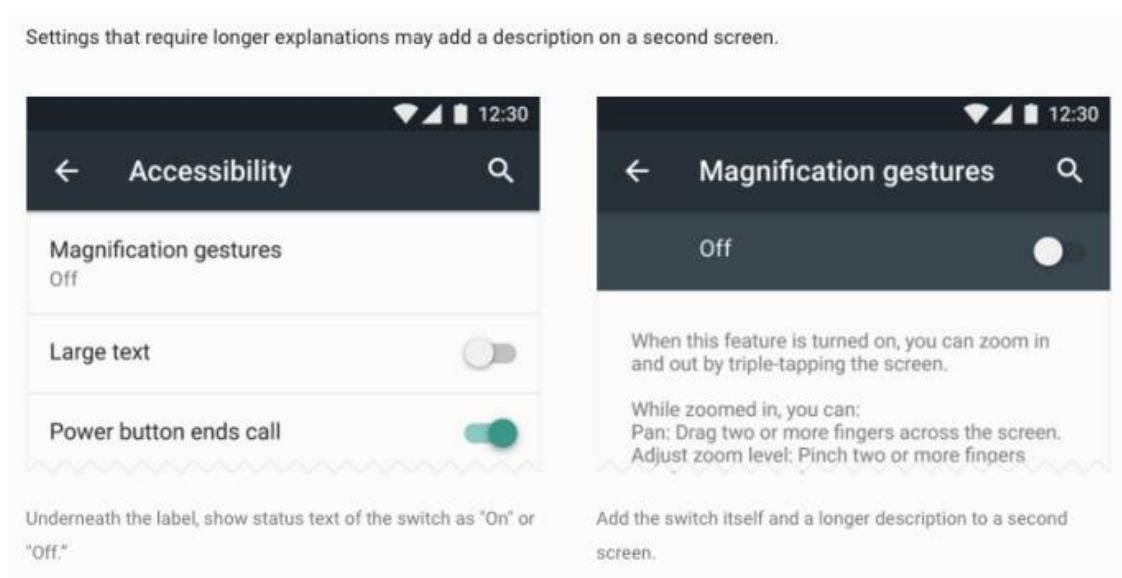


Figure 21: Switch button Implementation

➤ RIDER'S ACTIVITY

In the riders activity choices of logging out and booking a cab are given accordingly. Google Maps Activity is implemented so as to choose desired location for the pick and drop location of the SAVARI.

As the user chooses the option to CALL A SAVARI, the user location stored in the application is sent over the PARSE server and is thereby retrieved by the driver over the internet. If the LOCATION of the user is not specified or denied by the system, the user is asked for permission by the application to access the location from the device.

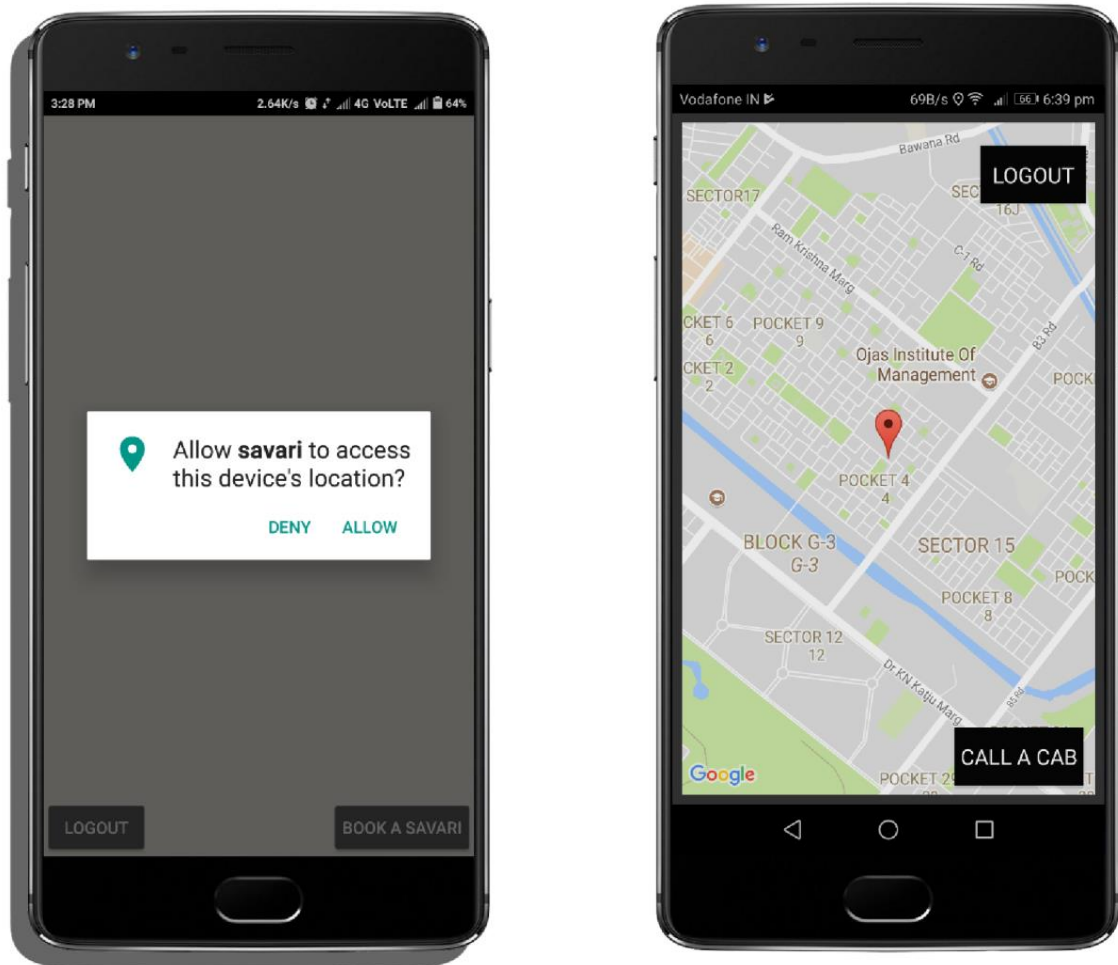


Figure 22: Savari – Rider's Activity

Calling a SAVARI through the application sends the user location and necessary data over the internet to the created PARSE served for the application.

When the user click on the button CALL A CAB, the button changes to CANCEL RIDE, giving the user an option to cancel the ride and exit the application.

GOOGLE MAPS ACTIVITY (API)

Build full-featured Android apps for your users. Google Maps APIs for Android are available via Google Play services so your app can be location-aware, include data-rich maps, find relevant places nearby and more

Google Maps Android API

Add maps to your Android app. Integrate base maps, 3D buildings, indoor floor plans, Street View and Satellite imagery, custom markers and more.

Google Places API for Android

Implement device place detection, auto-complete and add information about millions of locations to your app.

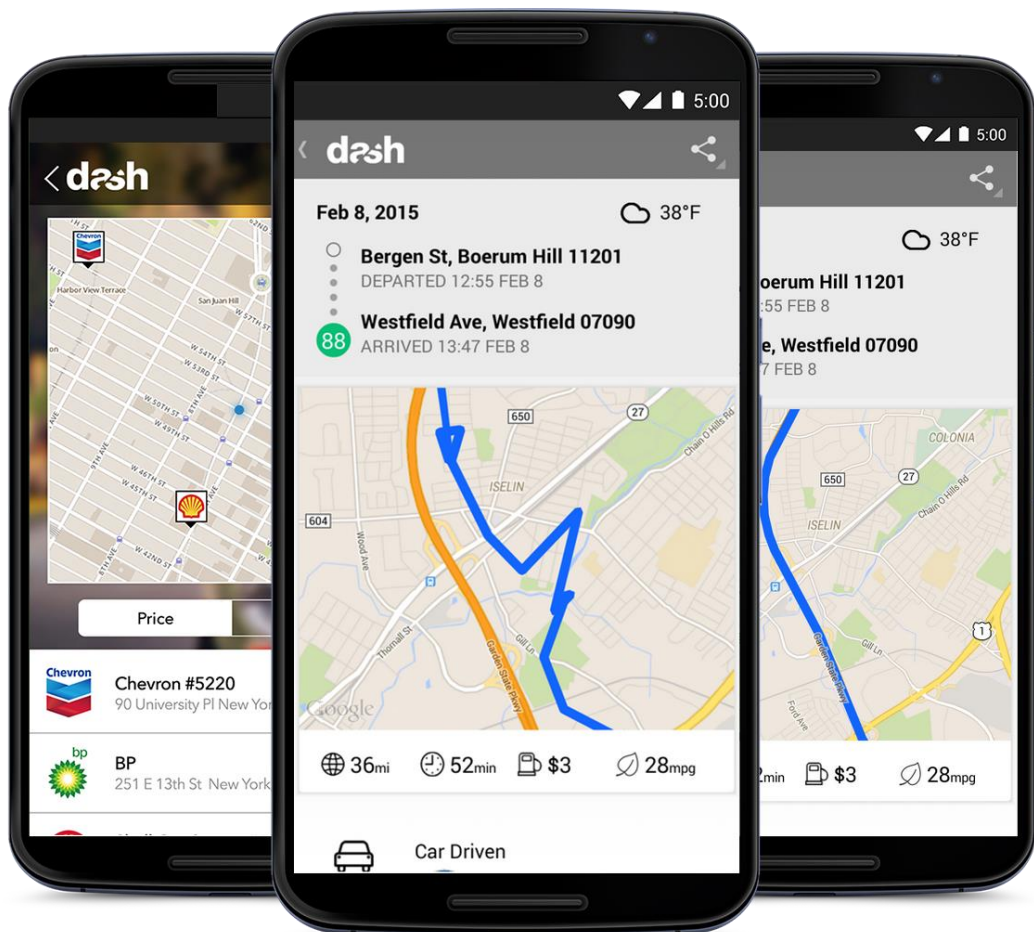


Figure 23: Google Maps Activity

To access the Google Maps API, GOOGLE provides the application with a unique API KEY which is used within the application to access the vast database of Google Maps.

Map View is implemented in Android Studio so as to depict the actual map location over the API. Android allows us to integrate Google maps in our application. You can show any location on the map, or can show different routes on the map etc. You can also customize the map according to your choices.

➤ DRIVER'S ACTIVITY

The DRIVER on the other side of the application is dealing with the same home page to login into the application and registering its data over the PARSE SERVER.

The driver activity consist of a simple LIST VIEW setup which fetches data of the RIDER'S location over the PARSE SERVER and fills up according to the nearest SAVARI to the DRIVER'S location .

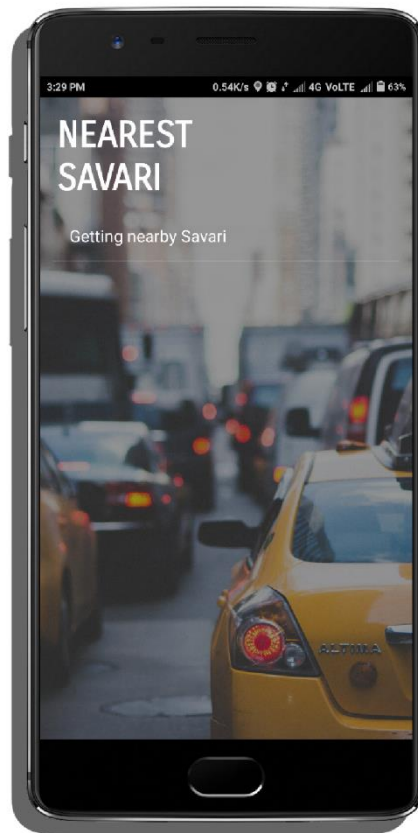


Figure 24: Savari - Driver's Activity

AMAZON WEB SERVICES AWS



Amazon Web Services (AWS) is a subsidiary of Amazon.com that provides on-demand cloud computing platforms to individuals, companies and governments, on a paid subscription basis with a free-tier option available for 12 months.

The technology allows subscribers to have at their disposal a full-fledged virtual cluster of computers, available all the time, through the internet. AWS's version of virtual computers have most of the attributes of a real computer including hardware (CPU(s) & GPU(s) for processing, local/RAM memory, hard-disk/SSD storage); a choice of operating systems; networking; and pre-loaded application software such as web servers, databases, CRM, etc. Each AWS system also virtualizes its console I/O (keyboard, display, and mouse), allowing AWS subscribers to connect to their AWS system using a modern browser. The browser acts as a window into the virtual computer, letting subscribers log-in, configure and use their virtual systems just as they would a real physical computer. They can choose to deploy their AWS systems to provide internet-based services for their own and their customers' benefit.

Bitnami has partnered with Amazon to make Parse Server available in the Amazon Web Services. Launch Parse Server with one click from the Bitnami Cloud Launch pad for Amazon Web Services.

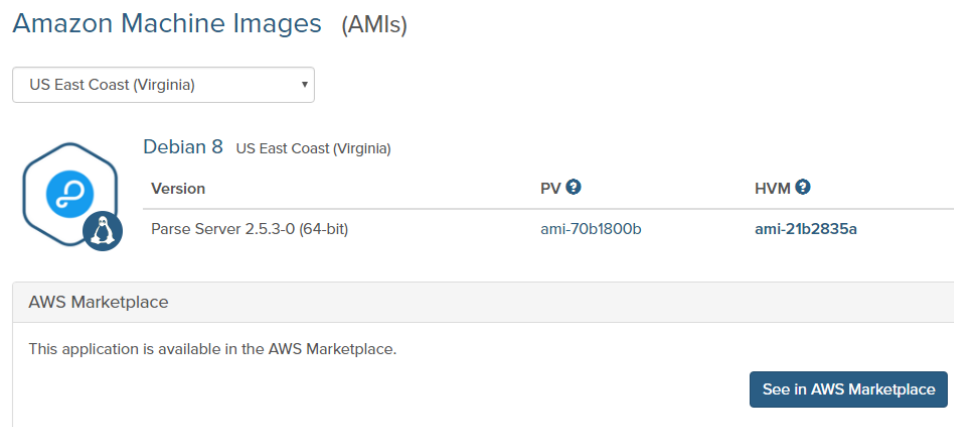


Figure 25: Amazon Web Services (AWS) Marketplace

In addition to providing free, ready to deploy Cloud Images, we also offer Bitnami Cloud Hosting, a subscription service that simplifies the process of deploying and managing open source applications to the Amazon EC2 Cloud. Bitnami has partnered with Google to make Parse Server available in the Google Cloud Platform.

PARSE SERVER

Parse Server is a new project, separate from the hosted Parse API service. Our intention is to provide and support the growth of an open-source API server, and allow new developers to benefit from the powerful Parse client SDKs regardless of where their application logic and data is stored.

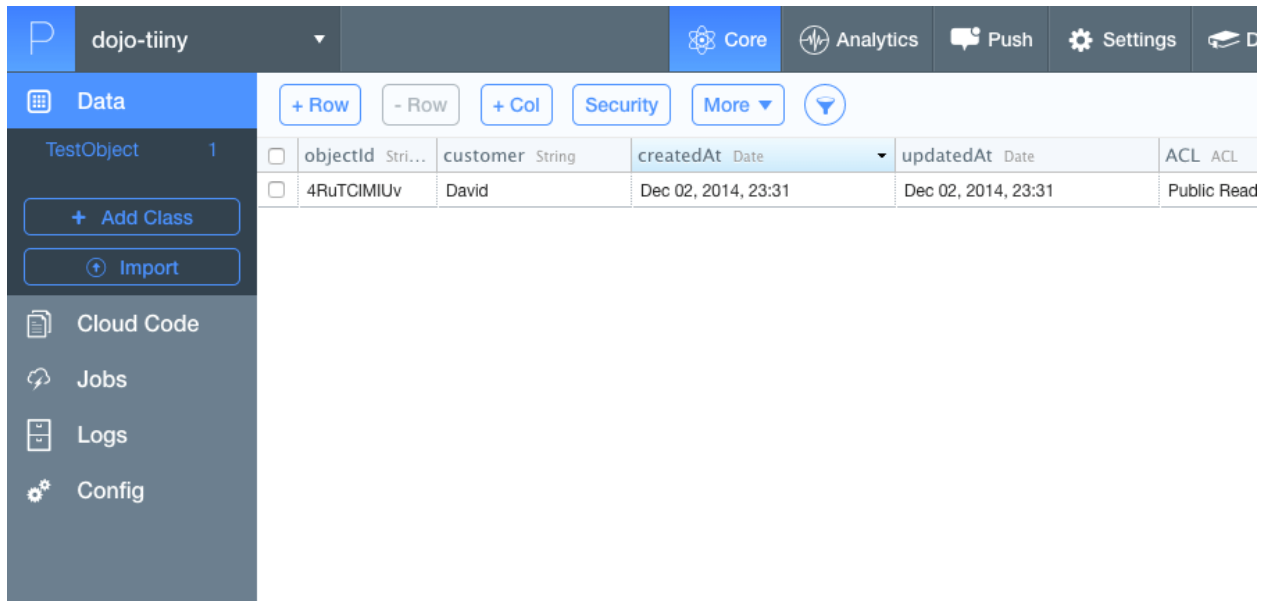


Figure 26: Parse Server Data home screen

Parse Server is an open source version of the Parse backend that can be deployed to any infrastructure that can run Node.js. You can find the source on the GitHub repo.

- Parse Server is not dependent on the hosted Parse backend.
- Parse Server uses MongoDB directly, and is not dependent on the Parse hosted database.
- You can migrate an existing app to your own infrastructure.
- You can develop and test your app locally using Node.

PREREQUISITES

- Node 4.3 ,MongoDB version 2.6.X, 3.0.X or 3.2.X
- Python 2.x (For Windows users, 2.7.1 is the required version)
- For deployment, an infrastructure provider like Heroku or AWS

COMPATIBILITY WITH HOSTED PARSE

There are a few areas where Parse Server does not provide compatibility with the Parse hosted backend. If you're migrating a hosted Parse.com app to Parse Server, please take some time to carefully read through the list of compatibility issues.

The fastest and easiest way to get started is to run MongoDB and Parse Server locally. Use the bootstrap script to set up Parse Server in the current directory.

CONCLUSION

- In a nutshell, this internship has been an excellent and rewarding experience. I can honestly say that my time spent interning with STREETINGO resulted in one of the best utilization of my time.
- Along my training period, I realized that observation is the main element to find out the root cause of a problem. Needless to say, my work isn't flawless both in technical and nontechnical aspect and could be improved with enough time.
- The flexible atmosphere was always welcoming which made me feel right at home. Additionally, I felt like I was able to contribute to the company by assisting and working on projects throughout my internship period.
- Due to my time spent within the organization, I was able to understand more and more about the industry and prepare myself to become a part of the industry in the future. It was an overwhelming experience to be a part of the company.

REFERENCES / BIBLIOGRAPHY

- <https://www.engineersgarage.com/articles/what-is-android/>
- <https://developer.android.com/studio/intro/index.html/>
- <http://androiddevcourse.com/>
- <https://www.udemy.com/complete-android-n-developer-course/learn/v4/content/>