

# DROPTAXI 2.2

Advanced online taxi software solution



# What's New in Droptaxi 2.2

Thank you for checking out Droptaxi 2.2. This release introduces some new features while also improving on features introduced in previous versions. With this release of Droptaxi our focus is on increasing the chances of riders getting drivers when they request for trips. In the ride hailing business successful trips means more profit, less losses and happy customers.

Droptaxi 2.2 introduces back-to-back trips feature. This feature allows drivers receive and accept trip requests when they are about finishing their current trip. Let's say a driver is on a trip, driving a customer from pickup location point A to destination point B. When the driver is 5KM or less from point B, the driver can receive another trip request. When the driver accepts this request it will be added as an auxiliary trip and the rider will get notified as usual that a driver has been assigned. Droptaxi handles both trips just like it's a single trip. One trip is displayed on the screen at a time. Drivers can switch between trips using the active trip button that appears on the app screen.

Droptaxi 2.2 includes a car pooling or ride share feature. Ride share allows customers share a ride with other random customers. With ride share, drivers handle multiple trips concurrently. This feature is available for vehicles which the ride share option has been enabled. You will find the ride share option under City | Tariff menu in the admin panel. Once enabled, drivers can receive and service multiple trips concurrently. The maximum number of concurrent trips a driver can handle is limited by the number of seats set for the vehicle category. Drivers can switch between active trips using the active trip button.

Droptaxi 2.2 now makes it possible for customers to retry a booking when no driver is found after the first try. In previous versions of Droptaxi, when a rider requests a trip and no driver is found or no driver accepts the trip request the rider had to go through the process of booking the trip again. In Droptaxi 2.2, when no driver is found, the rider will be presented with an option to retry the booking. When the customer retries the booking, the driver search distance radius is increased and the request is sent to up to 5 drivers at the same time instead of in sequence. Drivers that had previously missed or ignored the request will also be sent another request. The first driver to accept the booking gets the trip and the notification disappears on the other drivers' apps.

Droptaxi 2.2 introduces a new improved scheduled trip function. Scheduled trips have been rebuilt from the ground up making it more useful and increasing the chances of riders getting a driver for their future trips. The new scheduled trips feature in Droptaxi 2.2 is vastly improved from the previous versions. First, riders can now book a trip in advance at 20 minutes minimum; down from 1 hour in the previous versions. When a customer books a scheduled trip, a broadcast notification is sent to all drivers in the city irrespective of their distance from the rider. Drivers can view a list of available scheduled trips through the scheduled trips menu on the driver app. A driver can decide on which trip to accept. Once the driver accepts the scheduled trip the customer is notified. 15 minutes to the pickup time, both rider and driver are notified and at the pickup time the driver gets a trip request notification. The driver can accept the request and service the trip. If the driver misses or ignores it, the next driver available is assigned the trip.

Droptaxi 2.2 introduces the alternate / substitute vehicles feature. This allows you setup alternate or substitute vehicle categories in a city which can be allocated trip requests if the main vehicle category isn't available. For instance, assuming you setup a city to have a tricycle and sedan vehicle categories...and you want that if a rider books a trip for a tricycle and its not available he can also get a sedan; you can achieve this using alternate vehicles feature configurable on the admin panel through the City | Tariff menu. Drivers can also enable or disable this feature if they want from their apps through the settings menu. By default it is enabled for all drivers.

Multiple payment gateway option has been added to Droptaxi 2.2. You can setup multiple payment gateways on the admin panel. When customers and drivers want to top up their wallets, all the payment gateways you have setup will be displayed and they can choose from the list. This feature breaks the old webhook URL format so ensure you update your webhook URLs on your payment gateway dashboard where necessary.

Droptaxi 2.2 now prohibits drivers from going online when they have pending documents or when funds in their app wallet has dropped below the minimum you have set. For riders, booking trips are disabled when they fall under these conditions. When a rider tries to book a trip using the wallet as a payment method and there is insufficient fund in the wallet, the app displays the wallet page prompting the

customer to add funds to his wallet. The same applies to driver who have low funds in their wallet and try to go online.

Cancellation reasons preset messages have been added in this release. You can setup a series of message presets that both riders and drivers can choose from when they want to cancel a trip. The message will be displayed in the admin panel under the bookings page and the bookings details page.

We have added the ability to filter customers and drivers records based on their document submission status. This makes it easy to access users whose documents have not been reviewed and take appropriate action in reviewing them.

Transactions can now be viewed from a single page. The Transactions menu item on the admin panel allows you access every transaction on your business. See more details of transactions and take appropriate action.

We have made several other performance improvements to Droptaxi that improves on how the software works in this release. This release fixes a security bug. If you are running on previous versions of Droptaxi It is recommended to upgrade to this latest version.

As always, we are grateful to all Droptaxi customers for their support and suggestions in ways to make Droptaxi the best it can be. We appreciate you all!

Thank you.

Michael Chike  
Lead Software Developer  
Droptaxi

# REQUIREMENTS

This documentation refers to Droptaxi version 2.2.0.

In order to setup and run Droptaxi, you will need the following:

1. A Linux VPS running Ubuntu 22.04 with at least 2GB RAM. You can get a Linux VPS from Digital Ocean for as low as \$15 per month. The more the active drivers the more RAM you will need. As a rule of thumb, you will need 8GB for every 500 online drivers.
2. A Development Laptop or Desktop computer with the latest **Nodejs / NPM, Cordova 12.0.0, JAVA Development Kit (JDK) 17, Gradle 8.7** and Android studio installed and setup. Ensure you include the required SDK tools in Android studio under SDK manager - **Android SDK Platform 14.0 ('UpsideDownCake')** and **Android SDK Build tool 34.0.0 and Google play services**). You will also need to install **Visual Studio Code IDE** for working with the software source files. For iOS you will need a Mac computer running **Xcode 15.1** or later.
3. A Google cloud account with access to Google cloud console to enable you create three Google Map API keys with the following APIs enabled for all keys: Directions API, Distance Matrix API, Geocoding API, Maps Embed API, Maps JavaScript API, Maps SDK for Android, Maps SDK for iOS, Geolocation API and Places API.
4. Droptaxi uses Firebase cloud messaging and Phone Authentication so you'll need to have an account on Google Firebase.
5. For reliable and fast notifications aside from push notifications offered by firebase cloud messaging, Firebase Real-time Database is also used. You will need to create a service account with the appropriate key downloaded and saved as part of a JSON file. This will be uploaded to the server and used for authentication on the server end as well as for sending push notification using FCM.

# UPGRADING FROM 2.1.1

Follow these steps to upgrade your server from Droptaxi 2.1.1 to Droptaxi 2.2

1. Backup your current Droptaxi database. When backing up your database, backup only the data and not the schema or structure.
2. Create a new database and import the Droptaxi 2.2 schema file into it. You will find the Droptaxi 2.2 database setup /schema file in Droptaxi 2.2 package; **drop-files -> install -> database\_setup.sql**. Do not delete your old database.
3. After importing the Droptaxi 2.2 database schema into the newly created database, truncate the tables to empty them.
4. Now import the database you backed up earlier into the new database.
5. Backup all your current Droptaxi files on your server. That is, copy your **drop-files** folder and **public\_html** folder and save them somewhere safe. Let's call this backup "**old\_backup**"
6. Now on your server, delete the drop-files folder and all files inside the **public\_html** folder.
7. Upload the Droptaxi 2.2 server files and copy the drop-files folder to the location where you deleted the old drop-files folder. Copy all the Droptaxi 2.2 files in the public folder into your server **public\_html** folder.
8. Access the "**old\_backup**" and copy the upload folder found inside the **drop-files** folder. Replace the Droptaxi 2.2 uploads folder on your server with it.
9. Open the Droptaxi 2.2 db.php file located on your server at **drop-files -> config -> db.php**. Change DB\_SCHEMA value to the name of the database you created in step 2. Also update DB\_USER and DB\_PASS with information of the database user.
10. Access the "old\_backup" and copy the ride\_imgs folder found in public\_html -> img -> ride\_imgs. Replace the Droptaxi 2.2 ride\_imgs folder with it.
11. Access the "old\_backup" and copy **ajax\_2\_1\_1.php** and **ajaxdriver\_2\_1\_1.php** files found in public\_html folder into the Droptaxi 2.2 public\_html folder on your server.

12. Access the "old\_backup" and copy the license.php file found inside the drop-files -> lib folder. Replace the Droptaxi 2.2 license.php file on your server with it.

13. Access the "old\_backup" and copy all settings files – **settingsdata.php**, **settingsdata2.php**, **settingsdata3.php** and **settingsdata4.php**. You will find these files inside the drop-files -> lib folder. Replace the Droptaxi 2.2 settings data files file on your server with them.

14. Access the new admin panel through your browser.

15. Re-activate Droptaxi so certain files can be updated.

# BACKEND SERVER SETUP

Droptaxi backend is written in core PHP. Ensure you have your VPS setup and running **Apache 2.4, PHP 8.1, MYSQL 5.7** or higher and **Redis 7.2 or higher**. We have integrated Redis in-memory data store into the Droptaxi tech stack to help with fast data access and also free up MySQL database on certain tasks.

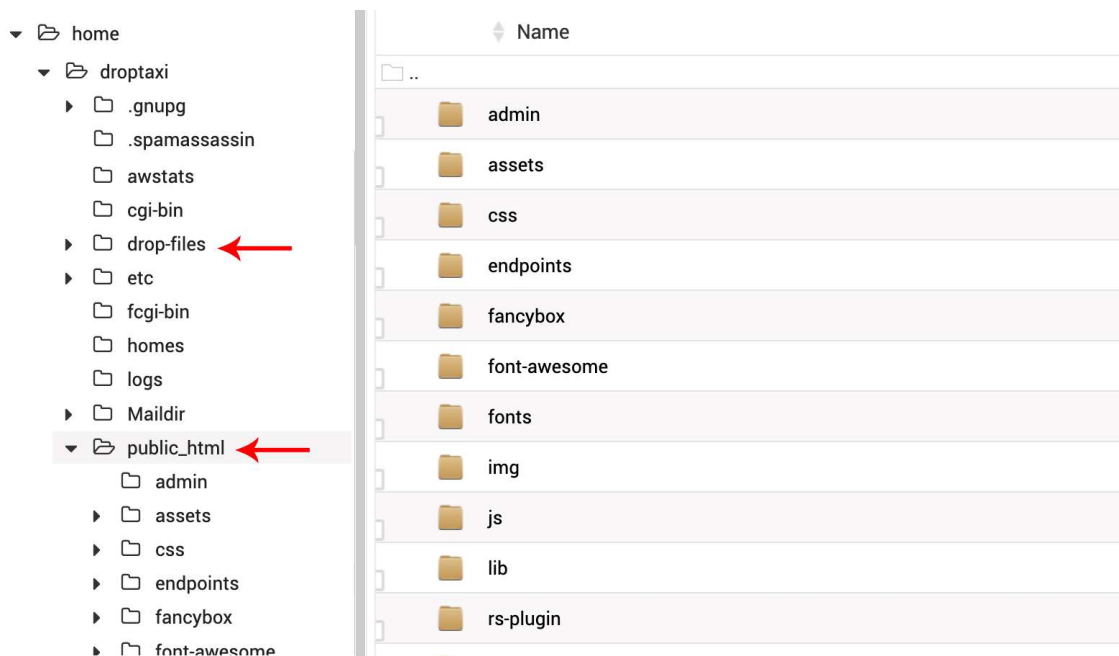
To avoid piracy or resale of the Droptaxi source code we lock every purchase to a single domain hence some parts of the source code are encrypted using ionCube encoder. Ensure you have **ionCube loader 13** or higher running on your server. Also ensure your server has the following PHP extensions installed:

**curl, redis, mbstring, exif, gd and openssl**

If you are running an unmanaged VPS we recommend you install a web server management software such as Webmin / Virtualmin to enable you manage your server easily else you can always use a SSH Terminal software such as Putty or Termius as well as FileZilla for file uploads.

Unzip the package file you received after purchase. You will find three folders – android, iOS and server. Open the server folder to reveal two folders - public and drop-files. Open the folder named public and upload all contents to your public\_html folder on your VPS. Then copy the drop-files folder to your VPS just outside the public\_html folder. The image below shows what the folder structure should look:





## MYSQL DATABASE SETUP

Droptaxi requires MySQL database server for operation. Setting up the database is mandatory. Your VPS server must be running MySQL server 5.7 or higher. To setup the database, using phpMyAdmin or any database management utility, first create a database e.g dreamtaxiDB and assign a USER with full permissions to the database. With the database created you will need to populate it with the required tables. Import the database setup file located at:

**drop-files -> install -> database\_setup.sql**

This file contains the SQL commands required to create the tables and setup the database. Once the database setup is complete, open the database config file located at:

**drop-files -> config -> db.php**

Enter your Database Schema / name (dreamtaxiDB) under **DB\_SCHEMA**, Database Username under **DB\_USER** and your password under **DB\_PASS**. Leave every other thing as it is and save the file. In some SQL versions you will need to ensure that ONLY\_FULL\_GROUP\_BY mode is disabled. Check and ensure it is disabled.

# REDIS SETUP

Redis is a fast in-memory data store. It is used as a cache and in-memory storage to hold drivers' location and bearing data. The driver apps update the server with their location data every 5 seconds. The server receives this data and stores it in memory using Redis. Redis was used due to its speed and flexibility, using MYSQL would be slow in this case. Also certain API calls now use Redis as cache in Droptaxi 2.2. We will be expanding the use of Redis in Droptaxi in subsequent releases.

To Install Redis on your server, follow these steps

Open your server ssh terminal and run these commands.

Note: This Redis setup guide assumes you are running **Ubuntu** on your server. If you are running a different Linux OS, please check online on how to install Redis on your server.

```
curl -fsSL https://packages.redis.io/gpg | sudo gpg --dearmor -o /usr/share/keyrings/redis-archive-keyring.gpg
```

```
echo "deb [signed-by=/usr/share/keyrings/redis-archive-keyring.gpg] https://packages.redis.io/deb $(lsb_release -cs) main" | sudo tee /etc/apt/sources.list.d/redis.list
```

```
sudo apt-get update
```

```
sudo apt-get install redis
```

Start the redis server by entering this command

```
sudo systemctl start redis-server
```

Enable Redis to always start whenever the server starts

## **systemctl enable redis-server**

Restart Redis

## **sudo systemctl restart redis-server**

Redis by default can use the entire server RAM. Open the redis configuration file in your text editor or in the server ssh terminal using the command:

## **sudo nano /etc/redis/redis.conf**

Make the following changes to the redis.conf file:

Enable writes on background save errors

```
453 # permissions, and so forth.  
454 stop-writes-on-bgsave-error no  
455
```

stop-writes-on-bgsave-error no

Uncomment "save 3600 1 300 100 60 10000" by deleting the #. Ignore if it is already uncommented.

```
437 # You can set these explicitly by  
438 #  
439 save 3600 1 300 100 60 10000  
440
```

Set maxmemory value to at least 10% of your server total RAM. Example if your server is 4GB, set maxmemory 400000000 (400 million)

```
1127 maxmemory 400000000  
1128
```

maxmemory 400000000

Save the redis.conf file.

Restart redis so your changes can take effect

## **sudo systemctl restart redis-server**

Test redis to ensure it is working by running this command

**redis-cli**

This will enter redis-cli mode. Then run the command:

**ping**

Redis will respond with "PONG".

Enable php to communicate with Redis server. Run this command to install php redis extension

**sudo apt-get install php8.1-redis**

Restart the server

# CONFIGURATION

Firebase cloud messaging is used for push notifications in Droptaxi. You will need to download a service account JSON file to your server to enable the server send push notifications to the Rider and Driver Apps.

- Visit <https://console.firebase.google.com> and login.
- Click on the 'Create a project' button
- Enter a name for your project, for example **Dreamtaxi**, click the 'Accept the Firebase terms' checkbox and click 'Continue'. In the next page, click 'Continue'. In the last page, accept the Google analytics terms and click on 'Create project' button. Wait while your project is created. When completed, click 'continue'.

Droptaxi 2.2 also uses Firebase real-time database, making it possible for rider app, driver app and server software to communicate in real-time. To setup Firebase real-time database:

- Under firebase project settings, Click the service accounts tab
- Click the 'Create service account' button to create a new service account for your project
- Click the 'Generate new private key' button. On the dialog that appears, click generate key.
- A .json file will be downloaded to your computer.
- Rename this file to **credentials.json** then copy it to drop-files folder and overwrite the already existing **credentials.json** file there.
- Add a firebase web app to your firebase project. Enter any name as the App nickname. Copy and keep aside the API key and storageBucket values. You will enter these in the web admin panel settings.
- On the firebase console left sidebar menu, Click Realtime Database the click the 'Create Database' button.
- In the Set up database dialog, leave the default location and click next.
- Under security rules, ensure Start in locked mode is selected, then click Enable.

- After the database has been created, Copy and keep aside the database url. You will enter the value in the web admin panel settings.
- Click on the Rules tab and rename the .read key value from false to true. Click publish to save the changes.

Droptaxi 2.2 now uses firebase cloud messaging HTTP v1 API. The use of secret key is no longer required. Our implementation requires only the service account credential JSON file information stored in the **credentials.json** file.

Google map plays a major role in Droptaxi. You will need five Google map API keys: one for the server, two for the Android rider and driver apps, and the other two for the rider and driver iOS apps. Follow these steps to create the Google map keys:

- Visit <https://cloud.google.com/> and click the 'Get started' button on the upper right of the screen.
- Create an account if you don't already have an account and also enter your billing information. Google charges for maps usage monthly but the \$200 dollar monthly usage credit can suffice when starting up your business.
- Proceed to create a new project, for example 'Dreamtaxi'.
- With the new project selected, on the left hand pane of the dashboard, select APIs & Services -> Library. This shows a list of library items sorted in categories.
- Expand the map category and select and enable each of the following APIs for your project:
  - Directions API
  - Distance Matrix API
  - Geocoding API
  - Geolocation API
  - Maps Embed API
  - Maps JavaScript API
  - Maps SDK for Android

- Maps SDK for iOS
- Places API
- Next, generate an API key by selecting 'Credentials' on the left pane of the dashboard. Click the '+ create credentials' button at the top side of the screen and select API key from the dropdown list. An API key is generated. Repeat this step to generate the rest of the five required API keys. Copy and keep the API keys.


Now you have all the required keys from third party services integrated in Droptaxi you can now proceed to configure the backend.

## ACTIVATING DROPTAXI

We have implemented a security feature that limits piracy or unauthorised distribution of Droptaxi source code. For every Droptaxi purchase, a license key is required to run Droptaxi. The license key is valid for the domain in which it was originally created. You will not be able to copy Droptaxi and run on a different domain with a single license key. However, you can run up to three instances of Droptaxi on one domain by setting up Droptaxi in different sub-domains.

When you purchase Droptaxi on Codecanyon you will receive a purchase code. You will need this code to activate Droptaxi on your domain.

Simply navigate to your site URL from a web browser. The Droptaxi activation page will be shown if not yet activated.



## Activate Droptaxi

Activation is required for Droptaxi to run on this server. Please fill the form below with the correct information. Ensure you enter the correct Server URL address where Droptaxi will run as every activation is linked to a domain and cannot be changed once activated for that domain. We recommend running Droptaxi on a sub-domain so you can use the main domain for your company's website.

\*Enter your codecanyon purchase code

\*Enter the server URL address where Droptaxi will run. Example: <https://appserver.dreamtaxi.com>

\*Enter your taxi business name. Example: **Dream Taxi**

\*Enter your business tag line. Example: **The Best Taxi Service**

**Activate**

Follow the instructions on the activation page to activate Droptaxi on your domain. Enter your Codecanyon purchase code, enter the server URL where you will be running Droptaxi. Droptaxi tries to automatically detect the URL and you will notice this field is populated already. Simply verify the URL is correct and make changes if necessary. Enter your taxi business name in the next text input field. Finally enter your business tagline. Then click the activate button to activate Droptaxi. A license key will be generated and stored on the server. Once activated, subsequent use of the same purchase code to activate Droptaxi will always regenerate the same license key.

You may want to backup your license key file. The license key file is located at:

**drop-files -> lib -> license.php**



## ADMIN LOGIN

The web admin panel setup is almost complete. Log in to the Admin dashboard by pointing your browser to:

*Yourdomain.com/login.php*

This will show the login page. Enter the default admin login details:

Username: [admin@droptaxi.com.ng](mailto:admin@droptaxi.com.ng)

Password: Droptaxi

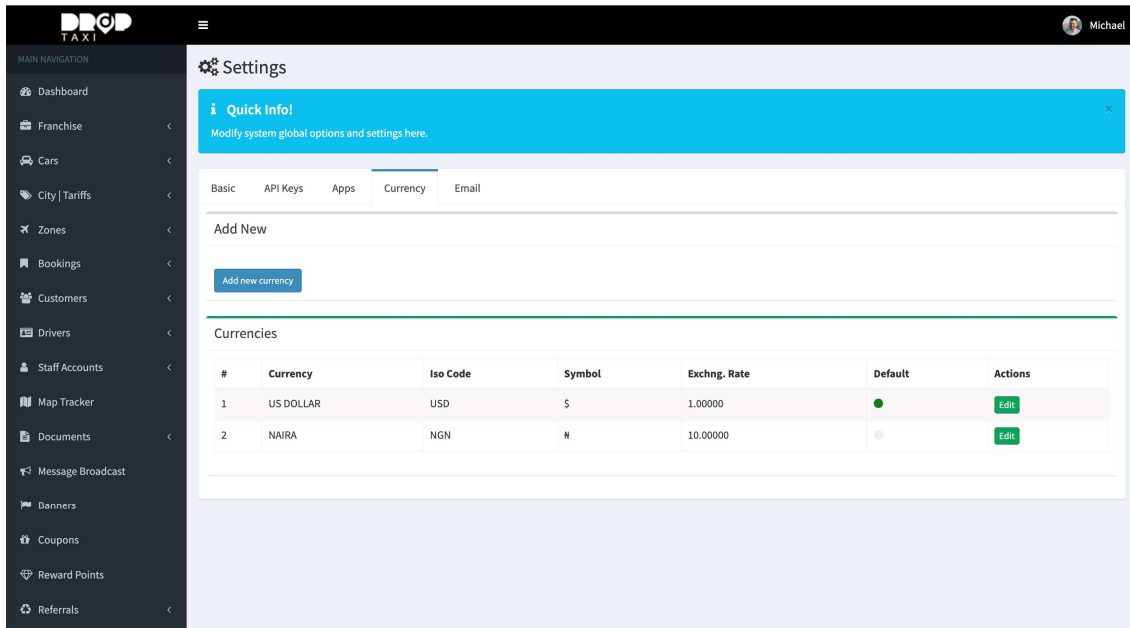
## BACKEND SETTINGS

Once logged in as an Admin, on the left sidebar menu, click on settings to access the settings page. The settings options are grouped into Tabs. Here you can setup parameters of Droptaxi that affects the way it functions. Please feel free to go through each of the options and understand their functions.

## DEFAULT CURRENCY SETUP

All prices and calculations are performed and displayed in your default currency. On the settings page, select the currency tab to setup your default currency. Once setup, we advise you don't change the default currency, changing it at a later time can result in corruption of data.

On the currency settings page click on 'Add new currency' button. The 'Add new currency' dialog is displayed. Select your currency from the dropdown list. Click the 'Mark as default' checkbox and then the 'Add currency' button to add your currency to the list.



You can add other currencies of cities you will be running Droptaxi too. You just follow the same steps as adding your default currency but leave the 'Mark as default' checkbox blank and ensure you add a value for the exchange rate of the currency relative to your default currency. For instance, assuming the currency Naira is set as the default currency and we can to add the Dollar as a currency; using google search, enter "1 Naira to Dollar". This gives 0.0024 thus this is the exchange rate value for Dollar.

## API KEYS SETUP

Select the API keys tab. Here you will setup APIs for your payment gateway, google maps, default OTP SMS service provider and Firebase real-time database.

On the payment gateway section, click the dropdown and select your payment gateway from the list. If your payment gateway is not available, get in touch with us to integrate your payment gateway. Ensure your selected payment gateway supports your set default currency.

Once you select a payment gateway, instructions on how to get the required keys from the payment gateway website are shown. Follow these instructions and fill in the required information to setup the payment gateway.

Next is Google maps API key. Enter one of the three Google maps API keys you generated earlier here.

Due to changes in pricing for Firebase Phone Authentication SMS service, we have added an option to integrate and use a third party bulk SMS service provider. Here you can choose between Firebase and a custom solution. If you would like to use a bulk SMS service provider in your country then you can get in touch with us to assist you with the integration. Please note that this is a paid service.

Next, enter values for your Firebase web API key, Firebase real-time database URL and Firebase storage bucket.

Click the Save button when done.

## Email SETUP

To enable sending emails to customers and drivers, you must configure the Droptaxi email. The email tab in the settings page enables you setup email.

First select the email transport type - System uses the default mail() function of PHP. Use this if your server has email enabled already. If you are using a VPS without email service enable then select custom SMTP to use an external SMTP service. You can use any SMTP service such as Zoho mail. You can also use Gmail. To use Gmail as your SMTP follow these steps:

1. Login to <https://accounts.google.com>
2. On the left sidemenu, click on security
3. Click on 2-Step verification and ensure it is turned on
4. Scroll down to the bottom of the page and click on App passwords
5. Create an app password by entering the name of your app. Example Dreamtaxi
6. Click on the create button
7. Copy and keep the generated app password

8. Return to the settings page of Droptaxi admin and select the Email tab.

9. For Email Transport, select Custom SMTP

\*Under SMTP host enter smtp.gmail.com

\*For SMTP username, use the GMail account username you created and app password for.

\*For SMTP password, Enter the app password you generated earlier. Click save to save your settings. Now emails can be sent using you GMail account through SMTP

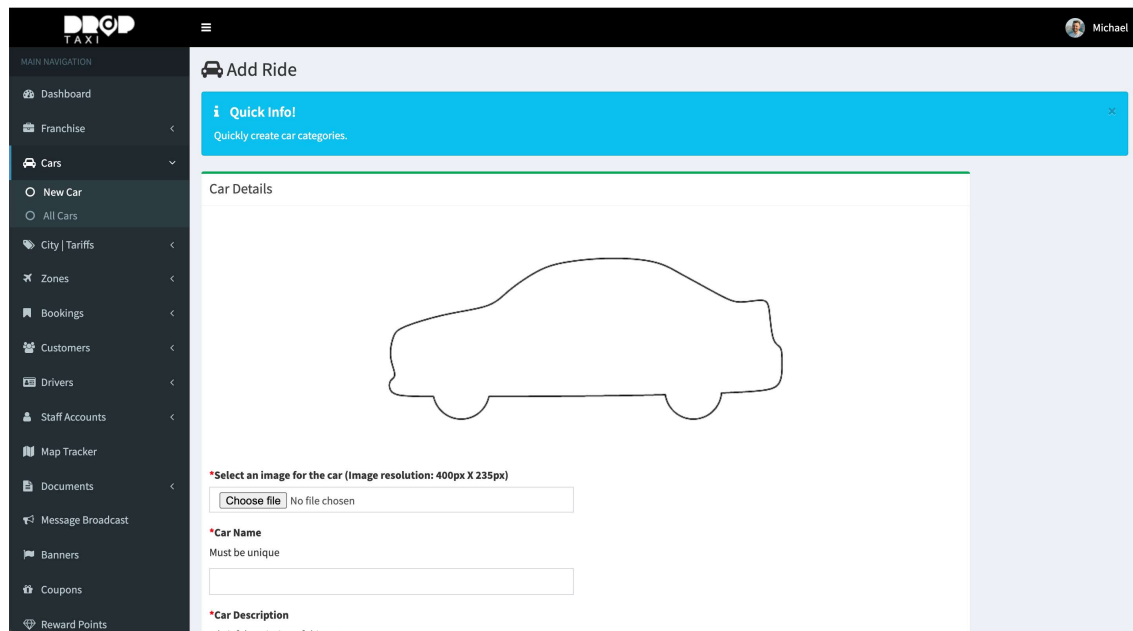
Click the "Test SMTP setting" to test your settings.

Enter a Sender's Email. This will appear as the sender of the email. If you are using Gmail SMTP you may have to use your Gmail username here.

Configure the rest of the email templates. When done click the save button.

# VEHICLES SETUP

Here you'll setup the vehicles available in your taxi service. These are vehicles your customers / riders can choose from when requesting a ride.



The screenshot shows the 'Add Ride' form in the DDD Taxi management system. The interface includes a dark sidebar menu on the left with options like Dashboard, Franchise, Cars, New Car, All Cars, City | Tariffs, Zones, Bookings, Customers, Drivers, Staff Accounts, Map Tracker, Documents, Message Broadcast, Banners, Coupons, and Reward Points. The main content area is titled 'Add Ride' and features a blue 'Quick Info!' banner with the text 'Quickly create car categories.' Below this is the 'Car Details' form, which contains a large empty space for a car image, a 'Choose file' button, and input fields for 'Car Name' (with a 'Must be unique' note) and 'Car Description'.

On the left sidebar menu, click on 'Cars' and select 'New car'. A form is shown where you can enter details of the car. Select a nice picture for the Car. We recommend images of dimensions 400 x 235 pixels. Enter a Car name and description. Select the maximum number of persons the vehicle can carry. This is used to keep track of the number of persons in a vehicle at a time when ride sharing is enabled. Select an icon that will be used to indicate the vehicle on the maps of the rider and driver app. Click the 'available' check box. Click the 'save ride' button to add the new car. Repeat these steps to add other vehicles.

Only 'available' cars show up on the rider app, if you uncheck the 'available' checkbox, the car won't be visible on the rider app.

# CITY | TARIFF SETUP

Here you'll setup cities where your taxi service will be available including the types of vehicles available in the city, city currency, ride fares etc.

**Note: To enable google and Apple reviewers test your apps during review you may skip modifying the default "World" city. You can instead add a new city where you will be running Droptaxi as explained in later part of this section.**

Droptaxi 2.2 comes with a default city titled "World" already created. Edit this default city to match your desired city.

On the left sidebar menu, click on 'City | Tariffs' then select 'All Tariffs'.

You will find the default "world" city. Click on the edit button to edit this city.

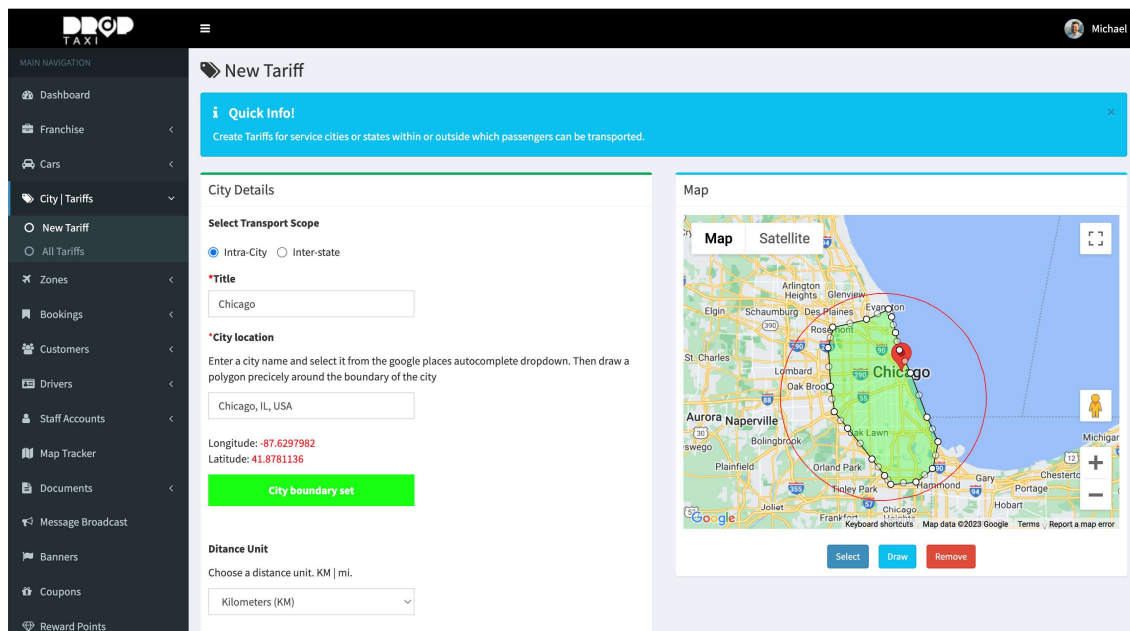
The screenshot displays the 'Edit Tariff' interface in the Droptaxi application. On the left is a dark sidebar with navigation options: Dashboard, Franchise, Cars, City | Tariffs (selected), New Tariff, All Tariffs, Zones, Bookings, Customers, Drivers, Staff Accounts, Map Tracker, Documents, Message Broadcast, Banners, Coupons, and Reward Points. The main content area is titled 'Edit Tariff' and features a blue 'Quick info' banner. Below it, the 'City Details' section includes: 'Select Transport Scope' with 'Intra-City' selected; 'Title' set to 'World'; 'City location' set to 'New York, NY, USA' with coordinates (Longitude: -74.0059728, Latitude: 40.7127753); and 'Distance Unit' set to 'Kilometers (KM)'. A green 'City boundary set' button is visible. To the right is a 'Map' section showing a world map with a red pin and a green polygon boundary around North America. The map includes 'Map' and 'Satellite' tabs, a 'Select' button, and a 'Remove' button.

Droptaxi 2.2 now includes the ability to define a boundary around a city. In previous versions of Droptaxi, the rider has to manually select his city on the rider app. In Droptaxi 2.2, this is done automatically with the help of city boundaries. It makes it possible to precisely detect the current city of a rider automatically. You must always draw a boundary around every city you create. Also there must not be any overlap between city boundaries.

First, on the map displayed, click on the green polygon to select the city boundary polygon. Then click the red button labelled 'Remove' to delete the polygon. You will draw another polygon later.

In the textbox labelled 'Title', enter a title for the city. Next, under 'city location', type the name of the city, as you type, Google maps displays a list of suggestions that match what you are typing, once you find the city name displayed in the Google maps suggestion list, select it. The longitude and latitude of the selected city is shown and the map on the right updates to show the location. Under the map are three buttons. The first button labelled "Select" allows you select a previously drawn polygon on the map. The second button labelled "Draw" allows you draw a polygon on the map. The third button labelled "Remove" allows you remove a selected polygon.

Click the Draw button and draw a city boundary polygon by clicking around the perimeter of the city on the map. As you click on the map, dots will be drawn with a line joining the points. Finish drawing the city boundary polygon by clicking the first dot. The image below shows a city with a boundary drawn around it.



Ensure you draw the boundary around the city and that the red marker is within the boundary. Once you have completed drawing the city boundary it will be indicated

Next, select the city distance measurement unit (Kilometer or Miles).

Under city currency, Click on the select box and select the currency of the city from the list. The currencies on the list are currencies you have added through the currency settings page. If you don't find your desired currency, go back to the currency setting page and ensure you add the city currency.

Under the 'City car tariffs' section, select the vehicles you want available in the city by clicking on the checkbox of the vehicle. As you click on the check box of each vehicle, it expands to reveal more options.

Enter values for pickup cost, drop-off cost, cost-per-minute and cancel cost. Repeat for night-time tariffs as well.

Droptaxi 2.2 allows you set a base distance value in Kilometers. The fare for trips with distances below the set base distance is given by the formula:

**Pickup cost + Drop-off cost**

This is the minimum or base fare. For trips with distances greater than the base distance, the fare is given by the formula:

**Base fare + (Cost-per-minute \* Trip time) + (Cost-per-km \* (Trip distance - Base distance))**

When you set the base distance to zero then the fare is computed using this formula:

**Pickup cost + Drop-off cost + (Cost-per-minute \* Trip time) + (Cost-per-km \* Trip distance)**

You can also setup different pricing for peak periods of the day by clicking the 'peak period charges' check box. Select the days and time you want the peak period pricing to take effect. Also set the charge type (nominal or multiplier) and enter a value for charge based on the charge type selected. When done, Click the save button to finish.

To add additional cities, on the left sidebar menu, click on 'City | Tariffs' then select 'New Tariff'. Follow the above steps to setup new cities.



Droptaxi supports interstate (long distance) tariffs or preset routes. Interstate tariffs are tariffs for rides between cities or fixed locations at a fixed set price. These appear on the rider app when you tap on the routes button located below the menu button.

When customers book an interstate trip, they cannot change the pickup or drop-off locations or the fare as these are already preset. In previous versions of Droptaxi, interstate bookings are dispatched to driver manually from the admin panel. In Droptaxi 2.2, interstate booking are dispatched immediately by the system just like normal bookings.

Setting up an interstate tariff is just like setting up an intra-city tariff. On the New tariff page, click the inter-state radio button. Enter a title for the inter-state tariff as it will appear on the rider app. Example **Lagos -> Abuja**

Under city location, enter the pickup city (Lagos) and the drop-off city (Abuja). As you type these cities, ensure you select them from the Google maps autocomplete list as they appear. Select the pickup city and then enter the pickup location and drop-off location or city. Follow the same steps as explained in setting up intra-city tariff to complete the inter-state tariff setup.

## Zones

Droptaxi 2.2 allows you target any location of interest within a city and set a fare for that location. These areas are called zones. When a rider sets a pickup or drop-off location within a zone, the trip fare is increased by the amount you set for that zone.

To add a zone, open the zone page by clicking the zone menu item on the left sidebar of the admin panel. Select 'New zone' submenu item.

Under zone name, enter a name for the zone. Select a city from the list of cities to set a city where you want to add the zone. The map zooms to the city. With the city boundary displayed around the city.

In the "city quick search" text box, start typing the name of the location you want to add the zone. When it appears in the google maps places autocomplete list, select it. A marker will be place in the location. Zoom into the map and use the drawing tool button to draw a polygon around the span of the location.

Select a zone fare increase type. You can choose between multiplier and additional types. If you choose multiplier type, the trip fare will be multiplied by the value you enter in the zone fare increase value. If you choose additional type, the zone fare increase value will be added to the trip fare. Click the save button when done to save the zone.

**DR**  
**TAXI**

Michael

### Add Zone

**Quick Info!**  
Create zones by drawing a boundary around an area of interest within a city such as airport, estate, park...and set extra fares for the area or zone.

**Zone Details**

**\*Zone Name**  
Enter a name for this zone

**\*Zone City**  
Select the city this zone will be found in

**Zone Map**

City quick search

Map Satellite

Enugu

Select Draw Remove

**\*Zone Fare Increase Type**  
Select how the fare will be increased

**\*Zone Fare Increase Value**

Save Zone

Copyright © 2023 Droptaxi. All rights reserved. Version 2.3.0

# CRON SETUP

Cron allows us automate tasks on Linux. With cron, tasks can be scheduled to run at regular intervals at a particular point in time. Droptaxi uses cron to start the auto-dispatch service and ensure it is always running. The Droptaxi auto-dispatch service handles all the work of dispatching ride requests to available drivers.

Follow these steps to setup cron on your server

**Note:** This assumes you are running **Ubuntu** on your server. If you are running a different Linux OS, please check online on how to setup cron on your server.

Open your server ssh terminal and run this command:

**sudo crontab -e**

A list of options will be displayed. Enter the number that corresponds with using nano as your editor to edit cron.

Inside nano, move your cursor to the bottom of the page and type this command into the nano editor to add a crontab entry for the Droptaxi auto-dispatch service on your server:

```
* * * * * php /path/to/public_html/cron.php
```

Replace **"/path/to/public\_html/"** with the path to your public\_html folder on your server.

Save the file on nano editor.

The above command tells cron to run the cron.php script in the public\_html folder every minute.

Once you setup cron, it will attempt starting the auto-dispatch service every minute. The auto-dispatch service handles the automatic allocation of booking to drivers. If there is a problem with the auto-dispatch service, drivers won't get ride requests.

Once the auto-dispatch service starts running, cron has no control over it. To start, stop or restart the service at any time, use the crondata.txt file. This file can be found inside the public\_html folder. Open the file. The content of the file determines the state of the auto-dispatch service.

The following values written in the crondata.txt file controls the auto-dispatch service.

**0** – A digit 0 stops the auto-dispatch service

**1** – A digit 1 starts the auto-dispatch service

**2** – A digit 2 restarts the auto-dispatch service

To check if the auto-dispatch service is running, enter a digit 2 in the crondata.txt file and save. This should restart the service and update the digit 2 to digit 1 in the crondata.txt file. Open the crondata.txt file and ensure this is the case.

## CORDOVA SETUP

The rider and driver apps are developed with Cordova using plugins written in Java / kotlin for Android and Swift / Objective-C for iOS. Cordova creates a web interface to these natively running plugins, this way you can use HTML, CSS and JavaScript for UI and logic while the core part of the app runs natively. The result is a beautiful, fast and fluid running hybrid App. Hybrid means it can run on both Android and IOS with almost the same code base.

The following explains how to setup cordova on a windows based computer in order to build the android apps.

Download and Install nodejs / npm by visiting <https://nodejs.org/en/download> (as at time of writing the latest version of nodes is 18.18.1 and includes npm 9.8.1).

Install cordova by entering this command in your computer terminal:

```
npm install -g cordova@12.0.0
```

This will install Cordova 12 on your computer.

Download and install Java Development Kit (JDK) 17 through this link:

```
https://www.oracle.com/java/technologies/downloads/#jdk17-windows
```

Ensure you select and download the windows version

Download **Gradle8.7** through this link:

```
https://gradle.org/install/
```

Download the binary-only version. Once downloaded, extract the zip file. Open the extracted folder. You will find another folder inside named 'gradle-8.7'. Copy this folder

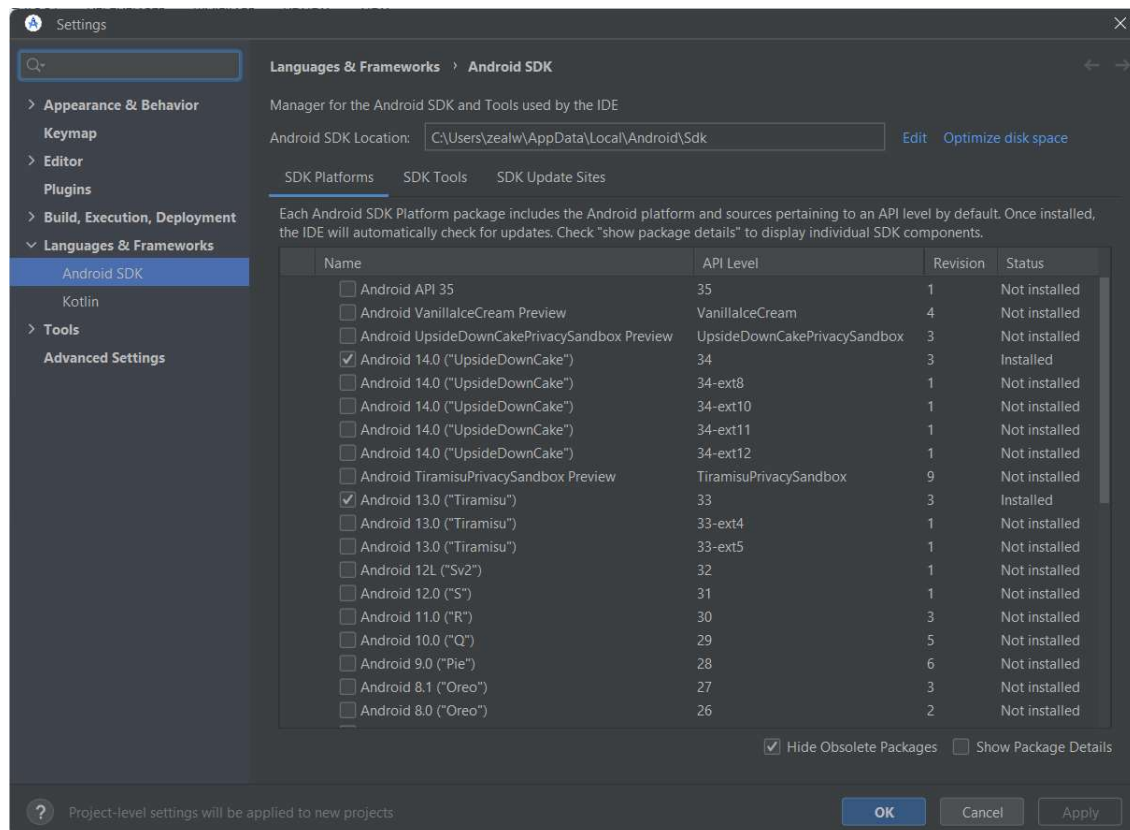
and paste it in the Android studio SDK installation folder. You will find this folder in this path: C:\Users\[xxxxx]\AppData\Local\Android\Sdk where [xxxxx] is your windows user account name. Rename the folder you pasted from **gradle-8.7** to **gradle**. NB: If you have gradle previously installed, delete the **.gradle** folder located at C:\Users\[xxxxx]\

Download and install android studio:

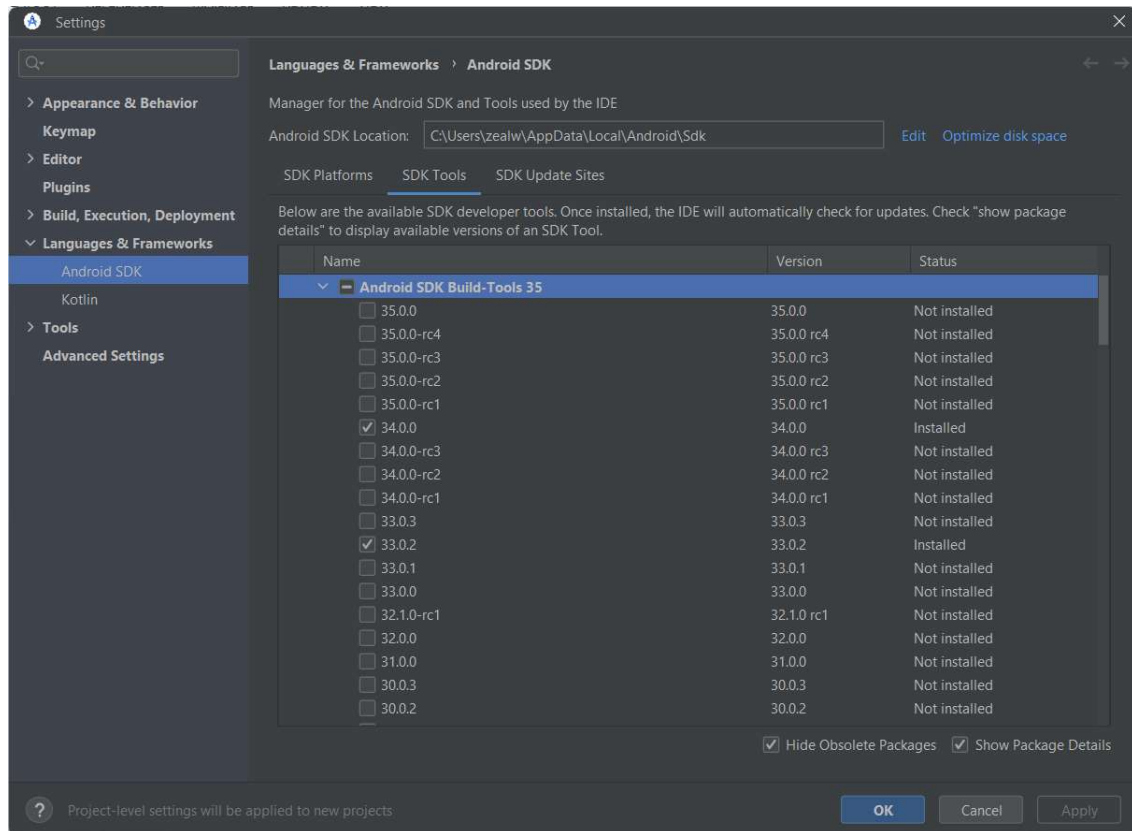
<https://developer.android.com/studio>

After install is completed, run android studio; it will run an additional setup. Once completed, open the SDK Manager dialog. On the dialog check '**Hide Obsolete Packages**' and uncheck '**Show Package Details**' checkbox.

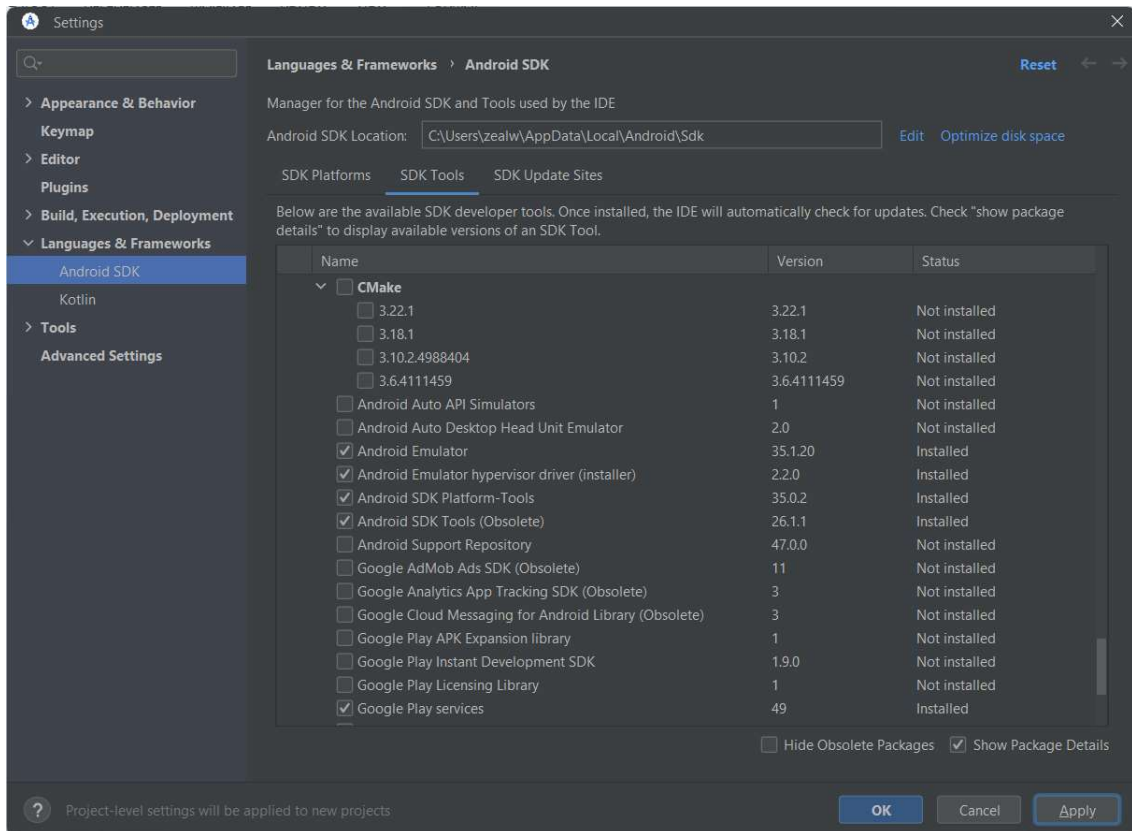
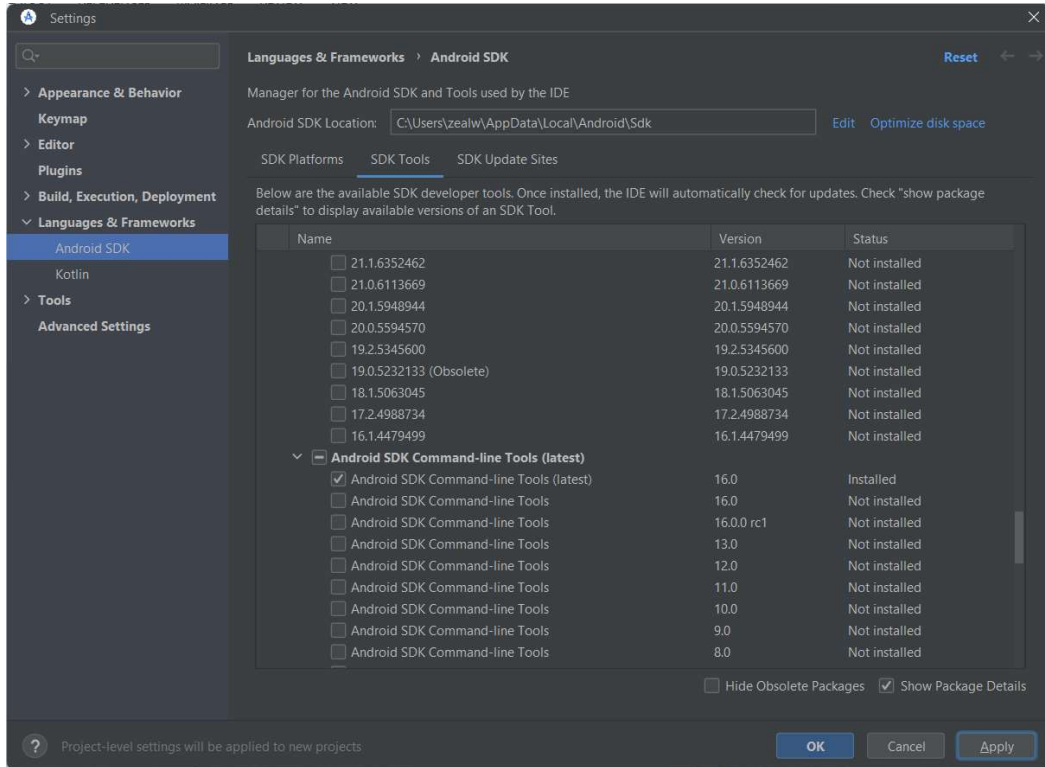
Under SDK platforms tab, from the list, locate and check the android platform target version Android 14.0 ("UpsideDownCake") as shown in the image below:



Select the SDK tools tab. On the dialog uncheck '**Hide Obsolete Packages**' and check '**Show Package Details**' checkbox. From the list, check build tool version 34.0.0 as shown in the image below:



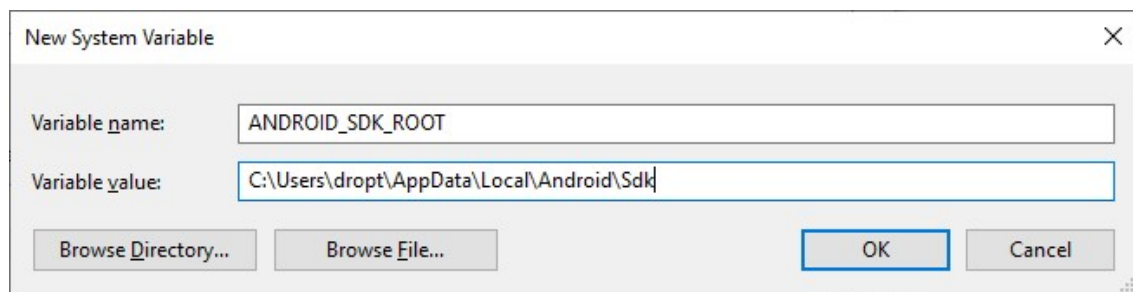
Scroll up and check the 'Android SDK command-line tools (latest)' and 'Android SDK Tools (Obsolete)' checkbox. Also check the 'Google play services' checkbox as shown in the image below:



Click OK. If a Terms and Conditions dialog appears, accept the terms and click OK. Android studio will start downloading your selected items. Once completed, close android studio.

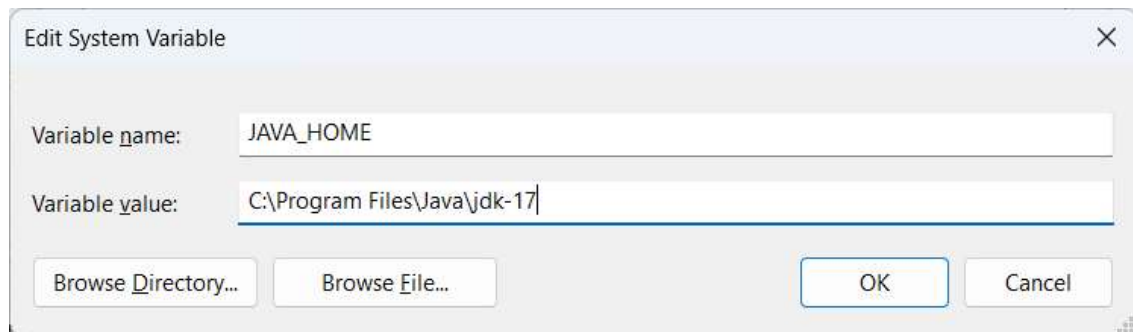
Next we will setup environment variables. On the windows taskbar, click the Start button and type 'environment variable'. On the search result list, click the 'Edit the system variables (control panel)' item. The system properties dialog is displayed. Click on the 'Environment Variables...' button.

On the Environment Variables dialog, Click 'New...' button. In the 'Variable name' text input field enter 'ANDROID\_SDK\_ROOT'. In the 'Variable value' text input field, enter the path to Android SDK installation folder. By default the path is: C:\Users\[xxxxx]\AppData\Local\Android\Sdk



Where [xxxxx] is the Windows user account name. You can use the 'Browse directory' button to navigate to the Android SDK installation folder following the path example. Click OK when done to save your new system Variable.

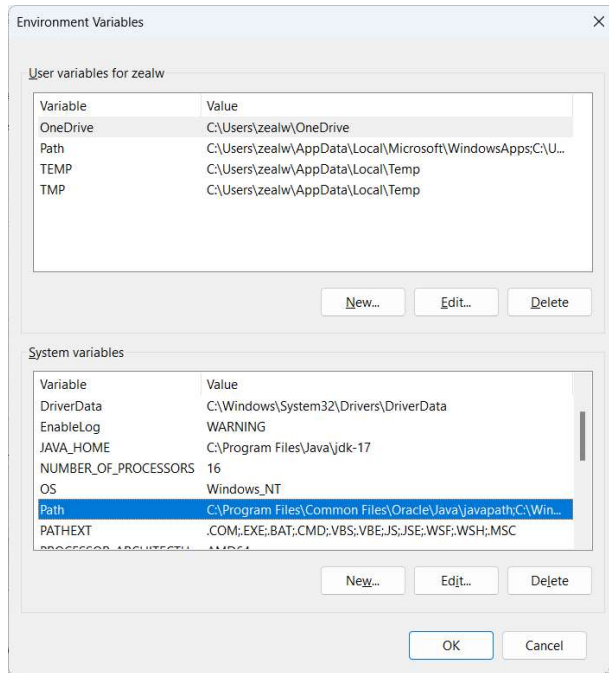
Add another variable with Variable name JAVA\_HOME and Variable value C:\Program Files\Java\jdk-17



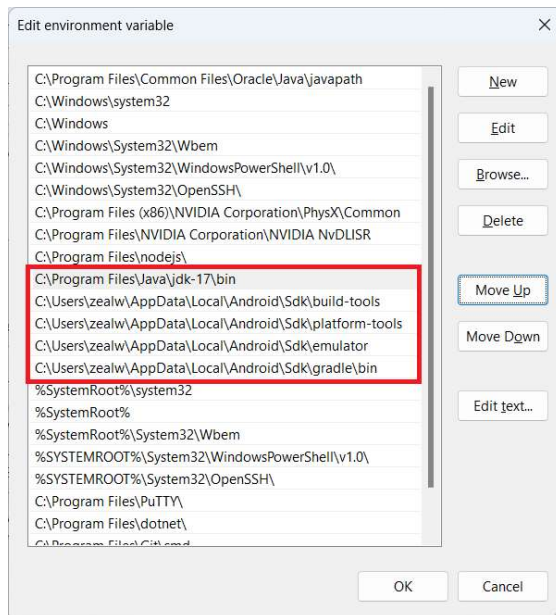
Your java version might be a bit different from this, use the 'Browse directory' button to navigate to the java installation folder. Click OK when done to add the new environment variable.



On the system variables panel, locate the item named 'Path' on the list and double click to edit it.



The 'Edit environment variable' dialog is displayed. Click the 'New' button to create a new entry, enter the paths in the red box in the image below one at a time. Replace 'dropt' in some of the path strings with your windows account name. Click OK when done.



This completes the Cordova setup.

Visual studio code is required to modify some Droptaxi source files as well as running some commands through the integrated terminal. Download and install Visual Studio code through this link:

<https://code.visualstudio.com/Download>

## ANDROID APP SETUP

### RIDER APP

Carefully follow these steps to setup the rider app. Open the android folder in the package you received from Codecanyon. You will find two folders – rider and driver. Open the 'rider' folder in Visual Studio Code. This folder contains all the source files of the Rider App. Open the file 'config.xml'. You will need to make some changes to this file.

```
1 <?xml version='1.0' encoding='utf-8'?>
2 <widget id="com.projectgics.drop" version="2.0.0" xmlns="http://www.w3.org/ns/widgets" xmlns
3 <name>DropTaxi</name>
4 <description>
5     A versatile White-label online taxi booking App.
6 </description>
7 <author email="support@projectgics.com.ng" href="http://projectgics.com.ng">
8     ProjectGICS
9 </author>
10 <content src="index.html" />
11
12 <access origin="*" />
13 <allow-intent href="http://*" />
14 <allow-intent href="https://*" />
15 <allow-intent href="tel:*" />
16 <allow-intent href="sms:*" />
17 <allow-intent href="mailto:*" />
18 <allow-intent href="geo:*" />
19 <platform name="android">
20     <allow-intent href="market:*" />
21     <preference name="LoadUrlTimeoutValue" value="700000" />
22     <splash density="port-hdpi" src="res/screen/android/splash-port-hdpi.png" />
```

Under the 'Widget' tag, in the 'id' attribute enter your package name for the rider app. Let's say you are using the 'dreamtaxi' example business name, you can use **com.dreamtaxi.riderapp** as the package name for the rider app. Under version attribute, set a version number or you can leave the default value and follow our versioning for Droptaxi to avoid confusions with updates. Please note that whatever version number you enter here must correspond with the version number you set in the backend settings page under the Apps tab, else the app will prompt for update. Under the 'name' tag enter the name of the App, E.g Dreamtaxi. Under the 'description' tag enter a brief description of the app. Update the 'author' tag with your email and website url.

Scroll up the page. You will see a series of 'preference' tags, under the preference tag with a name attribute GOOGLE\_MAPS\_ANDROID\_API\_KEY, enter the second Google maps API key from the three keys you generated earlier. Save the config.xml file.

Open the file **index.js** located at:

**android->rider->www->js->index.js**

At the top of the file are the variable declarations. Under APP\_TITLE enter your app name in quotes E.g "Dreamtaxi". Under APP\_VERSION\_ANDROID enter the app version

you entered in config.xml file. Under 'siteurl' variable enter the url of the backend server E.g <https://dreamtaxi.com> (Do not include a forward slash at the end).

Under 'carrier\_country\_code' variable, enter your country 2-letter code. This sets the default country dial code on the login page of the apps.

Save the index.js file.

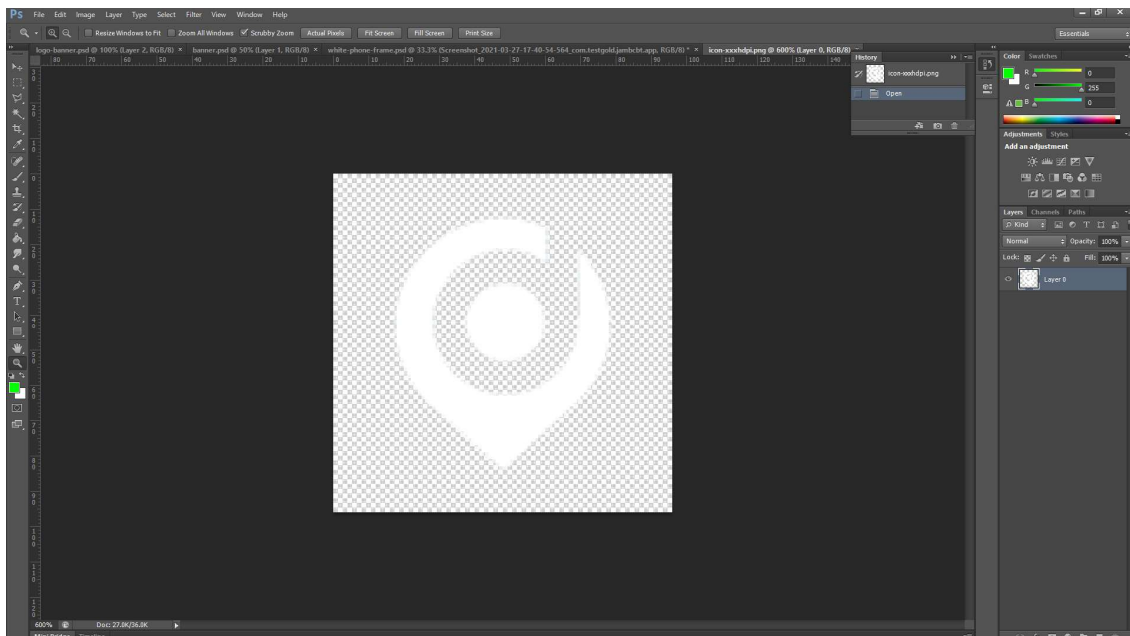
For the splash screen of your app, replace the **splash-icon.png** image file in **android->rider->res->icon->android** folder with your own image. Since Android 12, google has changed the way splash screens work in an app. Your **splash-icon.png** image should be of a resolution 1024 X 1024. The background of the image should be of uniform color with your app icon centered as shown in the image below.



Take note of the background color HEX value you use in your splash-icon image. You will need to enter this value as the background color for the rest of the splash screen so everything blends.

For the icon images, replace the image files in **android->rider->res->icon->android** folder with your own images. Ensure you maintain their respective filenames and resolution.

For notification icons, create an all-white image of your icon image file as shown in the image below. Make the background of the icon image transparent and save it in different resolutions as the images files in **android->rider->res->icon->android-notification**. Ensure you maintain their respective filenames too.



Change the app logo by replacing the logo.png file located at **android->rider->www->img->logo.png** with your own logo file. Use a 200 X 200 pixel resolution logo.

Droptaxi 2.2 supports basic theming. You can change the background and foreground colors of the app to match your brand colors. Open the file located at **android->rider->www->css->styles.css** in visual studio code.

```
main.html profile.html index.js build.json conf
www > css > styles.css > ...
17 under the License.
18 */
19 * {
20   -webkit-tap-highlight-color: rgba(0,0,0,0); /* make trans
21 }
22
23
24
25 :root{
26   --set-foreground-color: #0f5bbf;
27   --set-background-color: #093c82;
28 }
29
30 .setbgcolor{
```

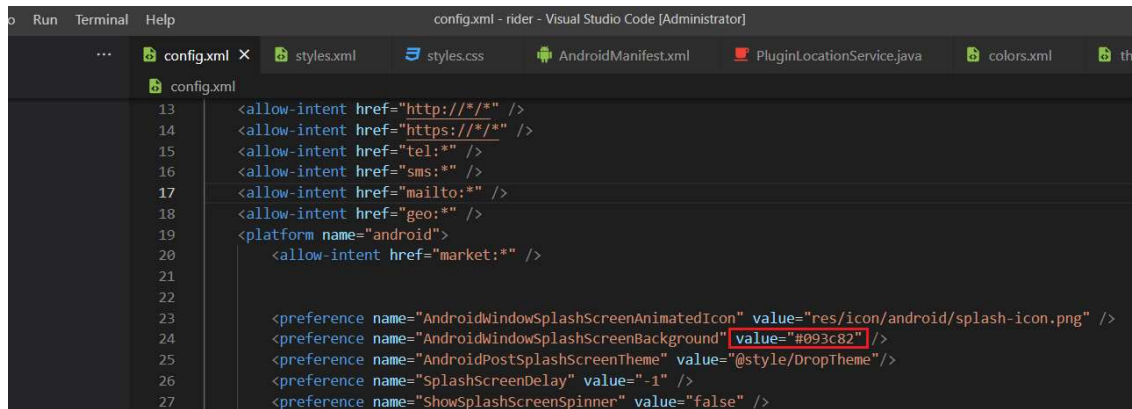
Locate the `:root` css selector and replace the values of the variables `--set-foreground-color` and `--set-background-color` with your desired hex color values. Save the file when done. Hint: always use darker shade colors for the background color and lighter shade colors for the foreground color.

Next open the file located at **android->rider->res->styles.xml** in visual studio code.

```
nal Help styles.xml - rider - Visual Studio Code [Administrator]
main.html profile.html index.js build.json config.xml
res > styles.xml
1 <resources>
2   <color name="primaryColor" #0f5bbf />
3   <color name="primaryColorDark" #093c82 />
4   <style name="DropTheme" parent="Theme.AppCompat.Light.NoActionBar">
5     <!-- on startup the background and app-name is shown, we use the
6     <item name="android:textColorPrimary">@color/primaryColorDark</
```

Change the `primaryColor` and `primaryColorDark` values to match the foreground and background colors you set for `style.css`. Save the file.

Open the `config.xml` file. Locate the `'preference'` tag with name `AndroidWindowSplashScreenBackground` and replace the value property with the background color hex value you used in your splash-icon image. Save the file.



```
13 <allow-intent href="http:///*/*" />
14 <allow-intent href="https:///*/*" />
15 <allow-intent href="tel:*" />
16 <allow-intent href="sms:*" />
17 <allow-intent href="mailto:*" />
18 <allow-intent href="geo:*" />
19 <platform name="android">
20   <allow-intent href="market:*" />
21
22
23   <preference name="AndroidWindowSplashScreenAnimatedIcon" value="res/icon/android/splash-icon.png" />
24   <preference name="AndroidWindowSplashScreenBackground" value="#093c82" />
25   <preference name="AndroidPostSplashScreenTheme" value="@style/DropTheme"/>
26   <preference name="SplashScreenDelay" value="-1" />
27   <preference name="showSplashScreenSpinner" value="false" />
```

Open the file located at **android/rider/platforms/android/android.json**. Look for all occurrences of "**com.dreamtaxi.riderapp**" and rename it to your rider app package name exactly as you entered it in the 'Widget' tag, in the 'id' attribute of the config.xml file.

To be able to upload your app to Play store, you will need to sign it with a key. First you'll need to generate a keystore file. In your computer terminal app, enter the command below to generate a keystore file. Again we are using 'dreamtaxi' as an example. You can replace it with your real business name.

**keytool -genkey -v -keystore dreamtaxi.keystore -alias dreamtaxi-app-key -keyalg RSA -keysize 2048 -validity 10000**

For this command to work, ensure you have setup environment path variable to point to your **android->android studio->java->bin** directory where the keytool exec file is located.

You will be asked a series of questions in the process of generating the keystore file. You will also be required to enter a password. Do not misplace the password. Once the process is completed, a keystore file dreamtaxi.keystore will be generated. Keep this file safe.

We need to get a SHA-1 key of the keystore file as this will be required in firebase. To get the SHA-1 key of the keystore file. Enter the command below in your computer terminal:

**keytool -list -v -alias dreamtaxi-app-key -keystore dreamtaxi.keystore**

Enter your password when requested.

The details of the keystore file are displayed including the SHA-1 key. Copy the SHA-1 key and save it.

Next you'll need to download your google-services.json file from firebase.

Visit <https://console.firebase.google.com> and login. Ensure your project is currently selected.

Droptaxi uses firebase phone SMS authentication on the rider app so enable it. On the left sidebar menu select Click on Authentication. Select 'Sign-in Method' tab. On the list displayed, locate 'Phone' and click the edit icon on the right. Click the switch to enable phone authentication.

Droptaxi supports integration of third party SMS services for use in sending OTP messages. You might want to use this option in place of Firebase SMS authentication if you find Firebase SMS expensive and unreliable to use in your country. We offer SMS gateway integration services. Get in touch with us so we can help integrate your SMS gateway for you.

On the Left sidebar menu, click the gear icon beside 'Project overview'. Click 'Project settings' on the popup menu. In project settings page ensure the 'General' tab is selected. Under 'your apps' section, add a new android app. On the page displayed, enter your android package name. This should be the same with what you entered in the config.xml file (Example **com.dreamtaxi.riderapp**). Enter an App nickname E.g Dreamtaxi-Rider. Then enter the SHA-1 key you saved earlier. Click the 'Register App' button to complete adding your new android app. In the next step – Dowload config file, you can download the google-services.json file for the app.

Copy the google-services.json file to this folder location:

**android->rider->platforms->android->app**

Open the file **MainActivity.java** located at:

**android->rider->platforms->android->app->src->main->java->com->projectgics->drop**

Rename the first line from

```
package com.projectgics.drop;
```



To your package name e.g com.dreamtaxi.riderapp

```
package com.dreamtaxi.riderapp;
```

Save the file. Then rename the folders to reflect the package name as well. From this:

**android->rider->platforms->android->app->src->main->java->com->projectgics->drop**

To this:

**android->rider->platforms->android->app->src->main->java->com->dreamtaxi->riderapp**

Then, run the command:

**cordova clean**

All is set; now we can build our app.

Open build.json in the rider root folder and fill in the details with the corresponding details of your keystore file. Below is a sample using dreamtaxi keystore.

```
1  {
2    "android": {
3      "release": {
4        "keystore": "dreamtaxi.keystore",
5        "storePassword": "12345678",
6        "alias": "dreamtaxi-app-key",
7        "password": "12345678",
8        "keystoreType": "",
9        "packageType": "apk"
10     }
11  }
12 }
```

Under packageType, you can set it to "apk" to generate an apk file you can side-load and test on your phone. Set to "bundle" to generate an aab file for upload to play store. When done, Save the build.json file.

Using your terminal, make sure your working directory is the rider folder.

Enter the command below to build your app:

**cordova build android --release --buildConfig=build.json**

This should build your App and, depending on your setting in build.json, create either a signed release APK file in this location:

**android->rider->platforms->android->build->outputs->apk->release->app-release.apk**

Or

A bundle file (aab) in this location:

**android->rider->platforms->android->build->outputs->bundle->release->app-release.aab**

Note: After building the app, sideload the apk file and test it on your android device. You will notice some Droptaxi references on the app. To change these to your app name you will need to modify the translation files and change the Droptaxi words to your app name. Read the "**Language Translation**" section of this documentation to learn how to modify the Strings shown on the apps.

## DRIVER APP SETUP

To setup the driver app follow the same steps as setting up the rider app but skip the parts where you enable phone authentication on firebase.

Also, when editing the config.xml file of the driver app, you should use a different package name for the driver app example **com.dreamtaxi.driverapp**. Use the last key out of the three google maps API keys you created. Leave the version number to the default to maintain our Droptaxi versioning.

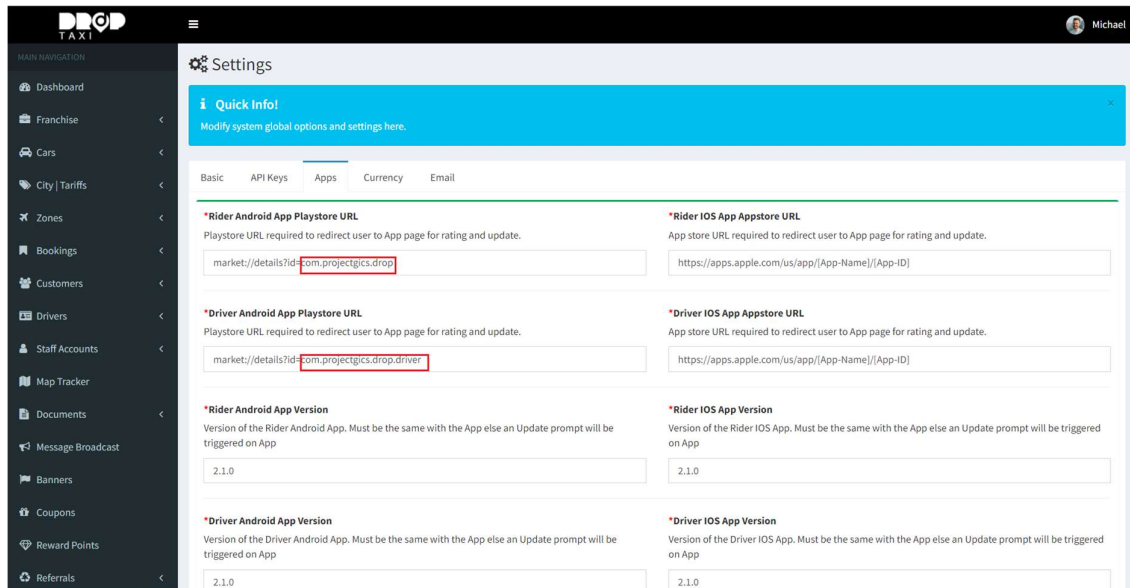
Open the file located at **android/driver/platforms/android/android.json**. Look for all occurrences of "**com.dreamtaxi.driverapp**" and rename it to your driver app package name.

Follow the same steps with the rider app to create and replace the splash screen, icon, logo as well as theme the app colors.

Add a new android app to firebase and enter the package name and SHA-1 key. Register the app and download the required google-services.json file and place in the same folder location as you did in the rider app.

You can use the same keystore file as the rider app to sign the driver app.

After setting up your Android rider and driver apps, access the settings page on the web admin panel. Click the Apps tab to reveal the app settings.



Replace the highlighted sections with your package names for the rider and driver app. Save the changes when done.

## IOS APP SETUP

### RIDER APP

Droptaxi 2.2 ships with an iOS version of the rider app. To setup the iOS rider app, a Macintosh (Mac) computer is required and a paid Apple developer account. Follow these steps to setup your iOS rider app.

Install Xcode. You can download xcode by visiting <https://xcodereleases.com/>

Version 15.1 was used to build Droptaxi 2.2. You may want to download and setup this version or a higher version.

Install Xcode command line tools by running the command below:

**xcode-select --install**

Install Homebrew by running this command below:

**/bin/bash -c "\$(curl -fsSL  
<https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh>)"**

After homebrew install is complete. You will find instructions on the screen on how to add homebrew to your PATH so you can use the "brew" command anywhere from your terminal. The commands look like this:

```
(echo; echo 'eval "$(/opt/homebrew/bin/brew shellenv)'" >>  
/Users/michaelchike/.zprofile
```

```
eval "$(/opt/homebrew/bin/brew shellenv)"
```

Install ios-deploy tools to enable launching IOS app on the IOS device from the command line. Use the command:

**brew install ios-deploy**

Install cocoapods tools needed to build IOS apps. Run the command below:

**brew install cocoapods**

Install Node and NPM. Run the command:

**brew install node**

Install Cordova. Version 12.0.0 is required. Use the command below to install this version.

**sudo npm install -g cordova@12.0.0**

Now you have setup the requirements for building iOS apps with Cordova, Open the package folder you downloaded from Codecanyon. You will see three folders – android, ios and server. Open the ios folder to reveal the rider folder.

Open the rider folder in Visual Studio code.

Open config.xml file and change the id from com.projectgics.drop to your app bundle id example com.dreamtaxi.rider . You can use the same package name you used for the android rider app as your bundle ID.

Change the <name> value to your app name example Dreamtaxi. This will be used as the bundle name.

**Note for iOS, there must be no spaces in your bundle names.**

Enter the Google maps API key you used in the Android rider app in the part labelled 'Enter your google maps API key'

Open index.js file located **in ios->rider->www->js** and change the APP\_TITLE variable value to the name of your App. Also change siteurl variable value to the server URL where Droptaxi web admin panel is setup.

Under 'carrier\_country\_code' variable, enter your country 2-letter code. This sets the default country dial code on the login page of the apps.

Save the file.

Setup your app splash screen. Create an image file measuring 2732X2732 with a uniform color matching the background color of your logo. Place your logo at the center of the image and save the file as PNG with a filename of **Default@2x~universal~anyany.png** in the location **ios->rider->res->screen->ios**

Open the Terminal app in the **ios->rider** folder and run the following command:

**cordova plugin save**

After the command has finished executing, run this command

**cordova platform add ios@7.1.0**

This will create an iOS Droptaxi app with the bundle ID and App name that you changed.

Next, login to your apple developer account and copy your Team ID. Open build.json file in **ios -> rider** folder and paste your Team ID in the fields marked XXXXXX. Save the file.

Navigate to firebase console website and add an iOS app to your firebase project.

Enter an Apple bundle ID for your app. Example com.dreamtaxi.riderapp (You can use the same package name you used in for android rider apps)

Enter an App nickname (Rider App) then click on the Register App button to create your app on firebase.

On the firebase console dashboard at the top of the left side menu, click the gear icon beside Project Overview, then select project settings.

Under the general tab, in the apps section, download the GoogleService-info.plist file.

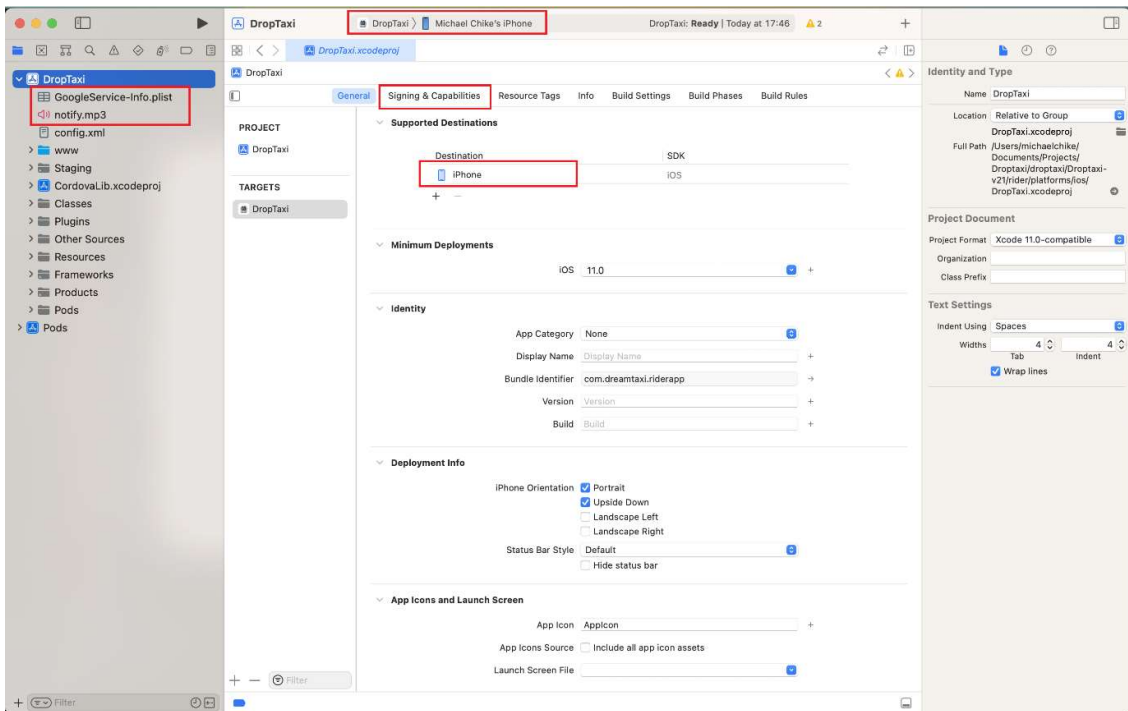
Setup firebase to use Apple push notification service (APNs). In your Apple developer account dashboard, on the left side menu, select certificates, Ids and profiles -> keys.

Click the Create key button. Enter a key name example dreamtaxiapnskey, then click Apple Push Notifications service (APNs) check box.

Click continue, then click register. It will take a little time to process, when done, click the download button to download your key. Also take note of the key ID. You can copy it some place.

Back in firebase project settings, under the cloud messaging tab, scroll to Apple app configuration section. Select your app and click the upload button in APNs Authentication Key section. In the dialog that appears, click browse, locate and select the APNs key file you downloaded. Enter the Key ID and your Team ID and click Upload.

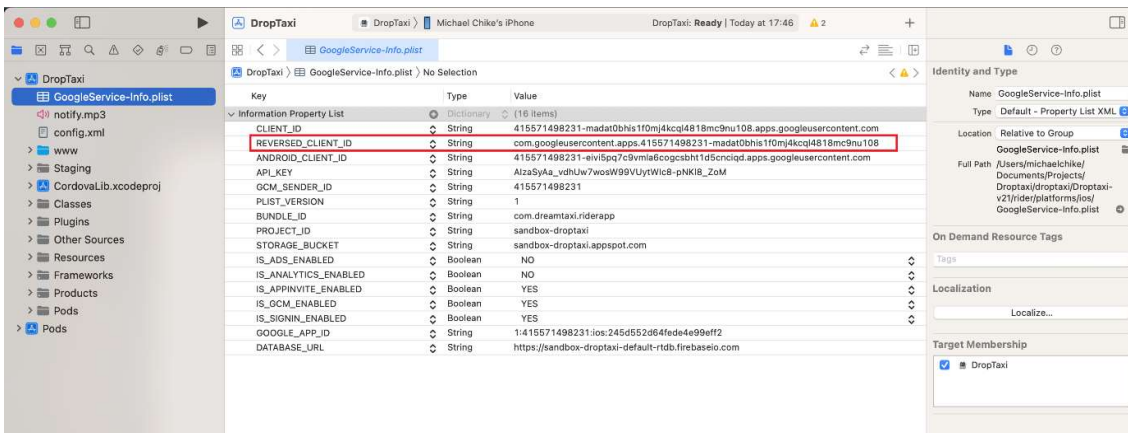
Open the [MyAppName].xcworkspace file located at **ios->rider-platforms->ios** folder in Xcode. On the Navigation panel, drag and drop the GoogleService-info.plist file into the project folder. A popup dialog will appear. Ensure Destination checkbox is checked. Also on Add to targets, ensure the checkbox there is checked.



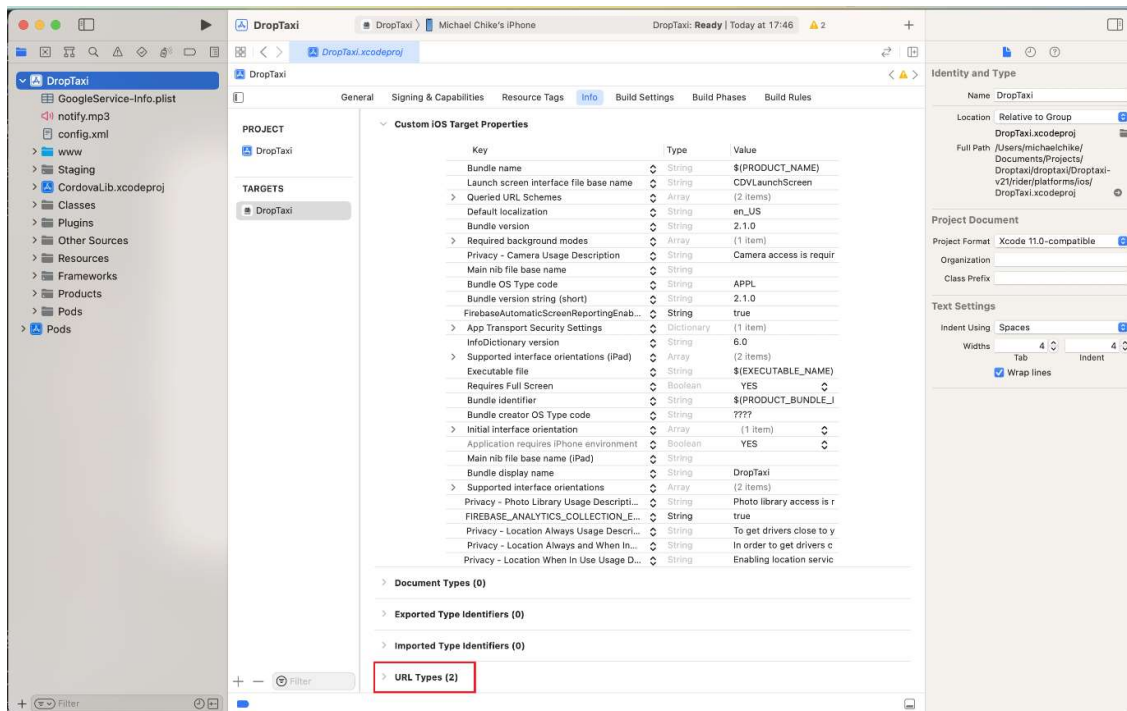
Also drag and drop the notify.mp3 file located at **ios->rider->res->sound**. A popup dialog will appear. Ensure Destination checkbox is checked. Also on Add to targets, ensure the checkbox there is checked.

Click on GoogleService-info.plist file to access it. A table of contents is shown. Locate the row labelled REVERSED\_CLIENT\_ID and copy the string value.

If you don't find the REVERSED\_CLIENT\_ID, return to firebase console, on the sidebar menu click on Authentication. Select the "Sign-in method" tab and click the "Add new provider button" then select Google and add it as a provider. Re-download the GoogleService-info.plist file and then add it to your project as explained earlier.



Click the root project folder in the navigation panel to reveal the project settings. Click the info tab and expand the URL Types section. Click the Plus (+) button and paste the REVERSED\_CLIENT\_ID you copied in the field labelled URL Schemes.



Plug in your iPhone to your Mac and ensure it is detected in Xcode. This is necessary as Xcode will automatically create a provisioning profile in your apple developer account and at least a device UUID is required for this purpose.

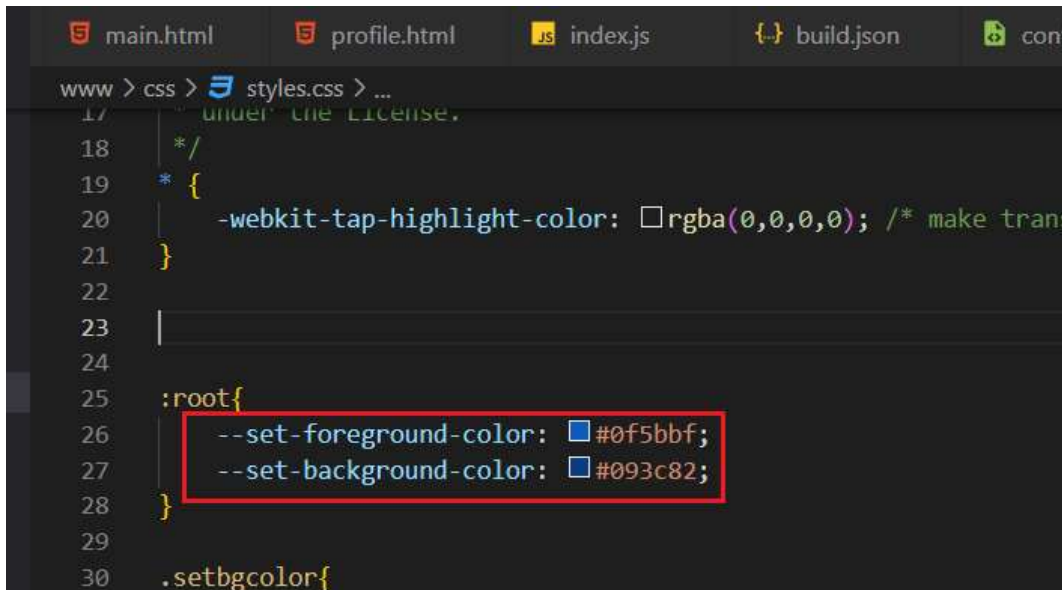
Click the Signing and Capabilities tab and check the automatically manage signing checkbox for both Debug and Release. In Team drop-down, select your Team ID from the list. If you don't see your team ID, select Add an account and login with your Apple ID and try again.

Create and save your icons. Create a single icon file with 1024 X 1024 pixels dimension in PNG image format. Visit <https://appicon.co> online icons generator website. Upload your icon image file and click the generate button. A file Applcons.zip will be downloaded. Extract the zip file and open the extracted folder. You will find another folder named Assets.xcassets. open this folder and copy the Applcon.appiconset folder to **ios->rider->platforms->ios->[MyAppName]->Assets.xcassets** folder and overwrite the **Applcon.appiconset** folder.

Copy the file **CDVViewController.m** located at **ios->rider** folder. Replace it with the file located at **ios->rider->platforms->ios->CordovaLib->Classes->Public->CDVViewController.m**



Basic theming allows you change the background and foreground colors of the app to match your brand colors. Open the file located at **ios->rider->www->css->styles.css** in visual studio code.



```
main.html profile.html index.js build.json conf
www > css > styles.css > ...
17  /* under the license.
18  */
19  * {
20  -webkit-tap-highlight-color: rgba(0,0,0,0); /* make trans
21  }
22
23
24
25  :root{
26  --set-foreground-color: #0f5bbf;
27  --set-background-color: #093c82;
28  }
29
30  .setbgcolor{
```

Locate the `:root` css selector and replace the values of the variables `--set-foreground-color` and `--set-background-color` with your desired hex color values. Save the file when done. Hint: always use darker shade colors for the background color and lighter shade colors for the foreground color.

Run your app on your iPhone to test your Droptaxi customizations as well as update your changes to xcode. Enter the command below to run and test your Droptaxi setup on your iPhone:

### **cordova run ios**

Ensure the Iphone is unlocked while the command is running. Your app will be built and copied to your iPhone.

After testing, switch back to Xcode, click on the project menu item and select Archive from the list. Your App will be archived and a dialog will be presented. If you in anyway close this dialog, you can bring it back by clicking on the window menu item and then click Organizer. In the Archives dialog, select your archive and click the Validate App button. This will validate your app. After it passes validation, click on Distribute App button. On the dialog select App Store connect and click next. In the next page, select Upload and click next. Keep clicking next in all pages shown. Finally click upload. This will upload your file to Apple app connect.

Once your app is published, get the appstore URL to your app page and update the "Rider IOS App Appstore URL" setting in the web admin panel settings page located under the Apps tab.

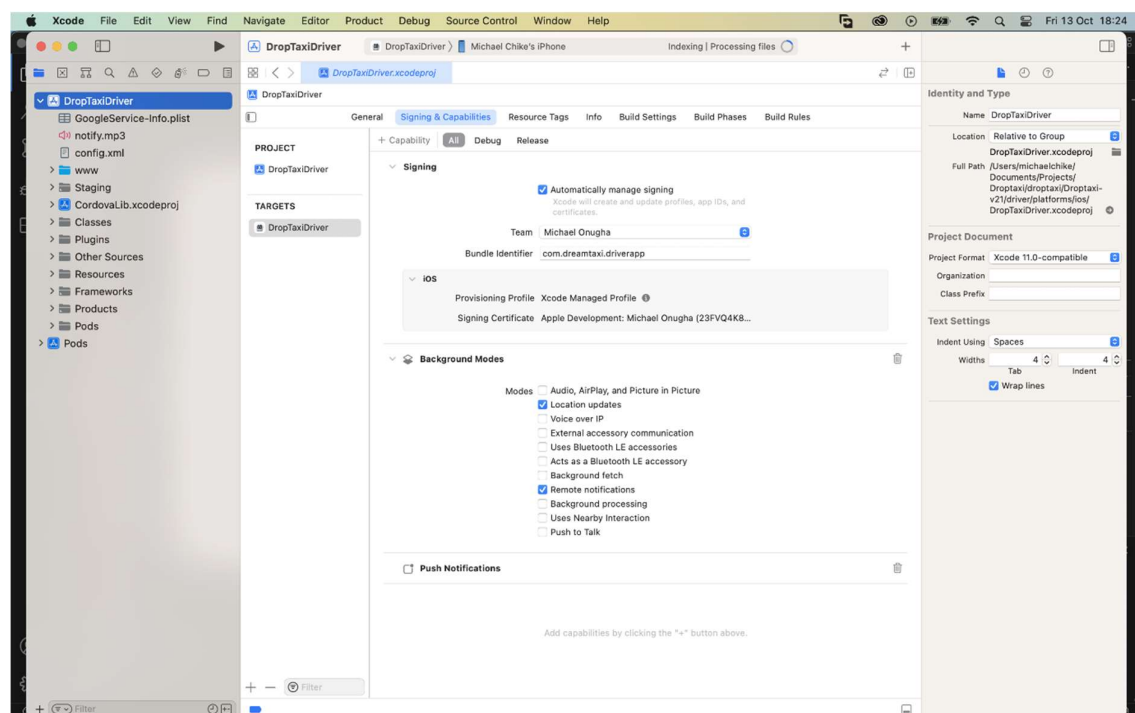
## DRIVER APP

To setup the iOS driver app follow the same steps as setting up the iOS rider app.

When editing the config.xml file of the driver app, you should use a different package name for the driver app example **com.dreamtaxi.driverapp**. Use the last key out of the three google maps API keys you created. Leave the version number to the default to maintain our Droptaxi versioning.

Follow the same steps with the rider app to create and replace the splash screen, icon, logo as well as theme the app colors.

In xcode, under background modes, ensure you select the options shown in the image below. **Location updates and Remote notifications should be the only items selected.** Also ensure you add the notify.mp3 file located at **ios->driver->res->sound** to enable the phone ring when a ride request is received even if the phone is locked.



Navigate to firebase console website and add another iOS app to your firebase project. Enter an Apple bundle ID for your app. Example **com.dreamtaxi.driverapp** (You can use the same package name you used for android driver app)

Enter an App nickname (Driver iOS App) then click on the Register App button to create your app on firebase.

On the firebase console dashboard at the top of the left side menu, click the gear icon beside Project Overview, then select project settings.

Under the general tab, in the apps section, download the GoogleService-info.plist file.

You don't have to bother creating another key file from your Apple developer account. Use the same key file you created for the rider app and setup Apple push notification service for the driver app. In firebase project settings, under the cloud messaging tab, scroll to Apple app configuration section. Select your driver app and click the upload button in APNs Authentication Key section. In the dialog that appears, click browse, locate and select the APNs key file you downloaded. Enter the Key ID and your Team ID and click Upload.

Follow the same steps you did for the rider app to setup your driver app on xcode and upload to Apple app connect.

**Note: Apple requires an organization developer account in order to publish apps that accept driver documents and bank account details. If you are using an individual account, then ensure you disable documents you have added in the admin panel and also, in the settings page, basic tab, delete all **Default Banks and codes**, then save. After your app is approved by apple you can restore these items.**

## LANGUAGE TRANSLATION

Droptaxi 2.2 supports multiple languages. Both LTR and RTL languages are supported. Eight languages are available by default – Arabic, Deutsch, English, French, Hindi, Portuguese, Russian and Spanish.

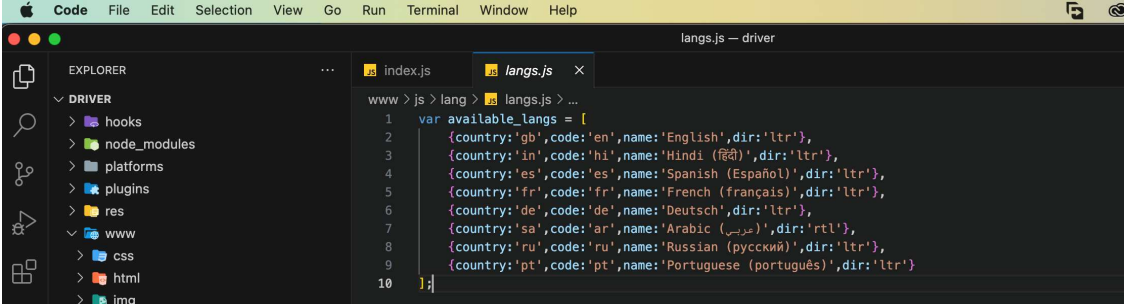
Users can easily switch between the eight languages from the login screen or from the profile page on the rider and driver apps.

Any language can be added to Droptaxi, easily. To add a new language, first get the 2-letter language code for the language. You can use google to search for the language 2-letter codes. Using Italian language as an example, the 2-letter code is **'it'**.

To add the Italian language for example, we will be translating a set of English phrases on the apps and server source code. We will use the rider android app to demonstrate. The procedure is the same for translating all the other apps as well.

Open the language folder located at **android->rider->www->js->lang**

You will find some JavaScript files. Open the file **lang.js** in visual studio code. This is the language descriptor file. All the languages available when the user taps the button to change language are listed here. Also the order they are listed in the file is the same



```
www > js > lang > lang.js > ...
1  var available_langs = [
2    {country:'gb',code:'en',name:'English',dir:'ltr'},
3    {country:'in',code:'hi',name:'Hindi (हिन्दी)',dir:'ltr'},
4    {country:'es',code:'es',name:'Spanish (Español)',dir:'ltr'},
5    {country:'fr',code:'fr',name:'French (français)',dir:'ltr'},
6    {country:'de',code:'de',name:'Deutsch',dir:'ltr'},
7    {country:'sa',code:'ar',name:'Arabic (عربي)',dir:'rtl'},
8    {country:'ru',code:'ru',name:'Russian (русский)',dir:'ltr'},
9    {country:'pt',code:'pt',name:'Portuguese (português)',dir:'ltr'}
10 ];
```

order they appear in the apps.

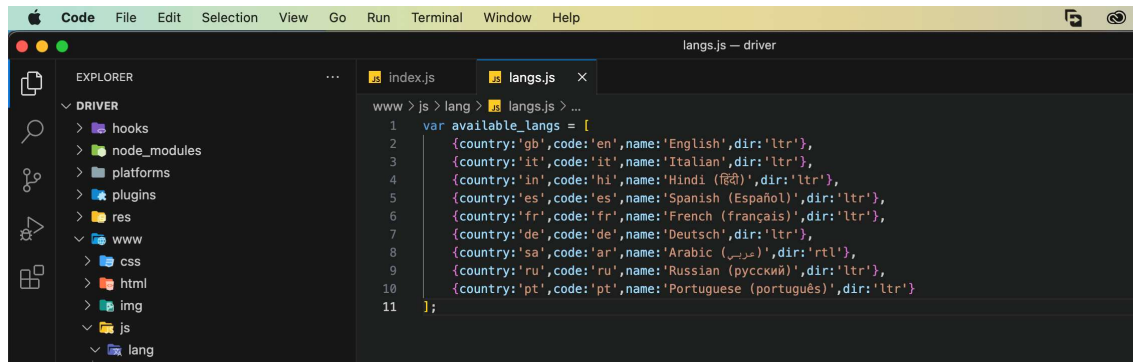
Let's add the Italian language. We will copy the code line (Line 2 as shown in the above image) for English language and paste it just below the same English language code line. Then we will edit the code line from this:

```
{country:'gb',code:'en',name:'English',dir:'ltr'}
```

To this:

```
{country:'it',code:'it',name:'Italian',dir:'ltr'}
```

This is a JavaScript object. The country property holds the 2-letter country code of the country that has the language. The country flag appears next to the language on the language selection page. The **code** property holds the 2-letter code of the language, the **name** property holds the display name of the language. This is what will be shown in the app. The **dir** property holds the direction of the language whether ltr (left-to-right) or rtl (right to left).



When done, the modified lang.js file should look as shown in the image below.

Save the changes you have made to lang.js file.

We have successfully created an entry for the Italian language, let's create the actual language translation file. Duplicate the file **template.js** (you will find it in the same location as lang.js) and rename it to **it.js** (as in our example of adding an Italian language); this is important. The language files must follow the format

### **[2-letter language code].js**

Use lowercase characters just as you did when adding the new language entry in lang.js.

Open the **it.js** file you created by duplicating and renaming the template.js file. You will see the content looking like that shown in the image below:

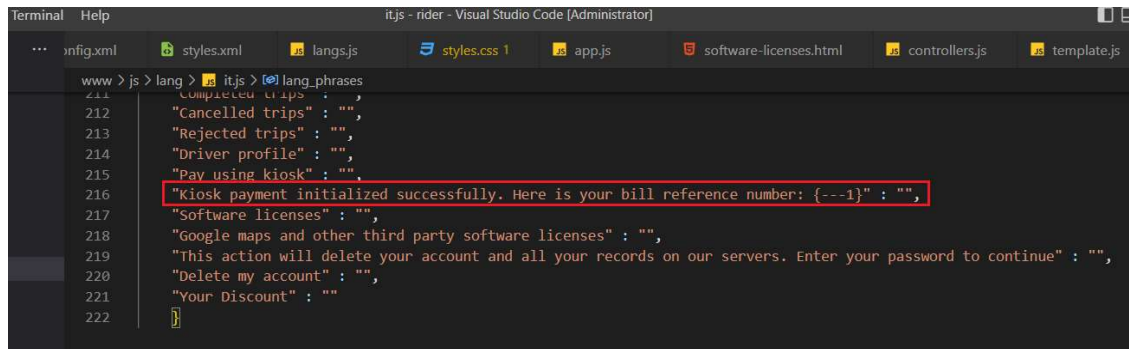
```
1 var lang_phrases = {
2   "Almost done..." : "",
3   "OK" : "",
4   "Connecting you to the nearest driver..." : "",
5   "From" : "",
6   "To" : "",
7   "Call" : "",
8   "SMS" : "",
9   "Cancel" : "",
10  "Inter-State Booking" : "",
11  "Select location..." : "",
12  "Use Location" : "",
13  "Continue" : "",
14  "My profile" : "",
15  "Routes" : "",
16  "Trips" : "",
17  "Wallet" : "",
18  "Logout" : "",
19  "Notifications" : "",
20  "Call for help" : "",
21  "Rate App" : "",
22  "Info" : "",
23  "phone" : "",
24  "password" : "",
25  "Forgot my password" : "",
26  "Sign in" : "",
27  "Don't have an account?" : "",
28  "Sign Up" : "",
29  "Firstname" : "",
30  "Lastname" : "",
31  "Address" : "",
32  "E-mail" : "",
33  "Confirm password" : "",
34  "Got a referral code? Enter it below to qualify for some discount on your next ride" : "",
35  "Referral Code" : "",
36  "Activate Account" : "",
37  "Enter the activation code sent to you via SMS into the field below" : "",
38  "No activation code received? please get in touch with us" : "",
39  "Verify" : "",
40  "A verification code has been sent to your phone via SMS. Please enter the code into the field below" : ""
}
```

The file contains a JavaScript object variable with all the English words and phrases used in the app arranged as properties of the object. The property values will hold corresponding language translations of the English word and phrases properties. You can also look at it this way; each line has a word or phrase in-between double quotes and separated by a colon ":" character. The word or phrase on the left of the colon is in English while the translation on the right should be in whatever language you are translating to, which in this case is Italian. Translating the first ten lines to Italian will look as shown in the image below:

```
1 var lang_phrases = {
2   "Almost done..." : "Quasi fatto...",
3   "OK" : "OK",
4   "Connecting you to the nearest driver..." : "Collegamento con l'autista più vicino...",
5   "From" : "Da",
6   "To" : "Per",
7   "Call" : "Call",
8   "SMS" : "",
9   "Cancel" : "Annulla",
10  "Inter-State Booking" : "Prenotazione interstatale",
11  "Select location..." : "Seleziona località...",
12  "Use Location" : "",
13  "Continue" : "",
14  "My Profile" : "",
15  "Routes" : "",
16  "Trips" : "",
17  "Wallet" : ""
}
```

All translations must be placed inside the provided double quotes on each line. The translations must not contain any double quotes character. Whatever you do, do not modify or change the English object properties. Only add your language translation of the English property of each line.

Some properties will contain variables in the form `{---1}`, your translations must include these variables as well.



```
Terminal Help it.js - rider - Visual Studio Code [Administrator]
... nfig.xml styles.xml langs.js styles.css 1 app.js software-licenses.html controllers.js template.js
www > js > lang > it.js > lang_phrases
211 "Completed trips" : "",
212 "Cancelled trips" : "",
213 "Rejected trips" : "",
214 "Driver profile" : "",
215 "Pay using kiosk" : "",
216 "Kiosk payment initialized successfully. Here is your bill reference number: {---1}" : "",
217 "Software licenses" : "",
218 "Google maps and other third party software licenses" : "",
219 "This action will delete your account and all your records on our servers. Enter your password to continue" : "",
220 "Delete my account" : "",
221 "Your Discount" : ""
222
```

Using the image above as an example, the Italian translation will look like this:

"Il pagamento del chiosco è stato inizializzato correttamente. Ecco il numero di riferimento della fattura: {---1}"

Notice the variable `{---1}` is also included. In this example, a reference code value will be inserted in the variable position in the translation. Sometimes you will encounter phrases with up to three or four variables like this `{---1} {---2} {---3} {---4}`. Try to understand what the phrase is referring to and then create a corresponding translation that includes all the variables in the right order meaningfully.

After translating all the words and phrases, save the file. That's all for the app.

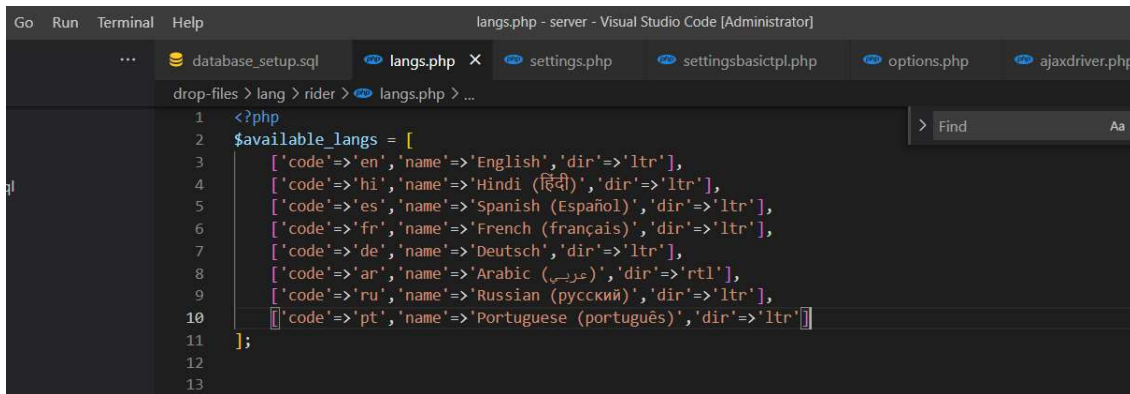
Next is the server source language files translation. The steps are mostly the same with the apps.

Open the language folder located at **server->drop-files->lang**

You will find three folders – `autodispatch`, `rider` and `driver`. The `rider` and `driver` folders each contains the language descriptor file **langs.php** and the language translation files of various languages. The language translation files in the `rider` folder provide language translations for strings used in the rider app server API code while those in the `driver` folder provide language translations for the driver app server API code.

Open the rider folder and then open the **langs.php** file in visual studio code.

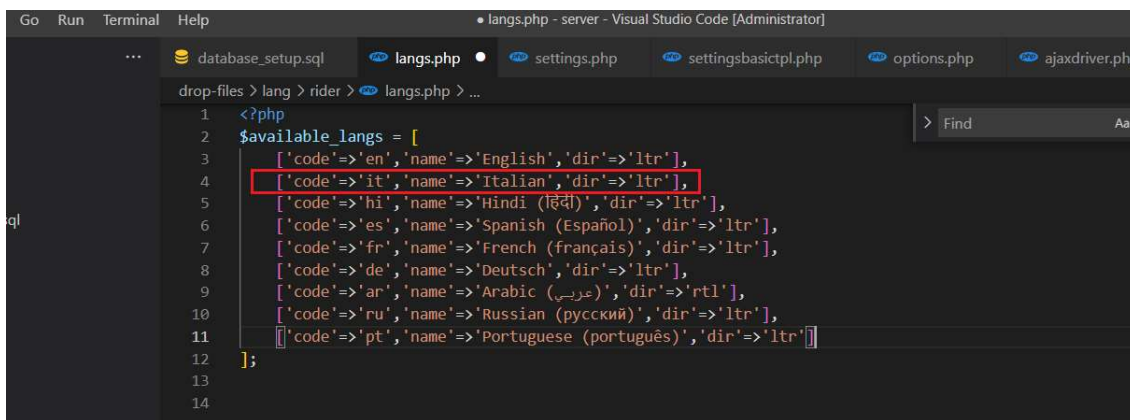




```
1 <?php
2 $available_langs = [
3     ['code'=>'en', 'name'=>'English', 'dir'=>'ltr'],
4     ['code'=>'hi', 'name'=>'Hindi (हिंदी)', 'dir'=>'ltr'],
5     ['code'=>'es', 'name'=>'Spanish (Español)', 'dir'=>'ltr'],
6     ['code'=>'fr', 'name'=>'French (français)', 'dir'=>'ltr'],
7     ['code'=>'de', 'name'=>'Deutsch', 'dir'=>'ltr'],
8     ['code'=>'ar', 'name'=>'Arabic (عربي)', 'dir'=>'rtl'],
9     ['code'=>'ru', 'name'=>'Russian (русский)', 'dir'=>'ltr'],
10    ['code'=>'pt', 'name'=>'Portuguese (português)', 'dir'=>'ltr']]
11 ];
12
13
```

All the languages available on the apps are listed here. The order in which they are listed doesn't matter.

Copy the English code line (Line 3) and paste it just below the English code line. Modify it to match the language you are adding. In the case of the Italian language example, the modified langs.php file will look like this:



```
1 <?php
2 $available_langs = [
3     ['code'=>'en', 'name'=>'English', 'dir'=>'ltr'],
4     ['code'=>'it', 'name'=>'Italian', 'dir'=>'ltr'],
5     ['code'=>'hi', 'name'=>'Hindi (हिंदी)', 'dir'=>'ltr'],
6     ['code'=>'es', 'name'=>'Spanish (Español)', 'dir'=>'ltr'],
7     ['code'=>'fr', 'name'=>'French (français)', 'dir'=>'ltr'],
8     ['code'=>'de', 'name'=>'Deutsch', 'dir'=>'ltr'],
9     ['code'=>'ar', 'name'=>'Arabic (عربي)', 'dir'=>'rtl'],
10    ['code'=>'ru', 'name'=>'Russian (русский)', 'dir'=>'ltr'],
11    ['code'=>'pt', 'name'=>'Portuguese (português)', 'dir'=>'ltr']]
12 ];
13
14
```

When done, save the file.

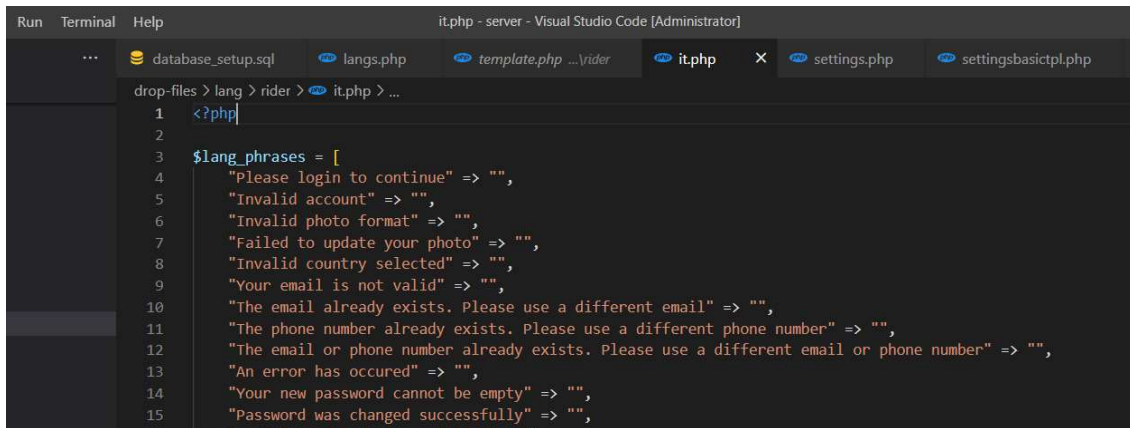
Duplicate the template.php file and rename the duplicate copy using this format

### **[2-letter language code].php**

In our Italian language example, this will be **it.php**

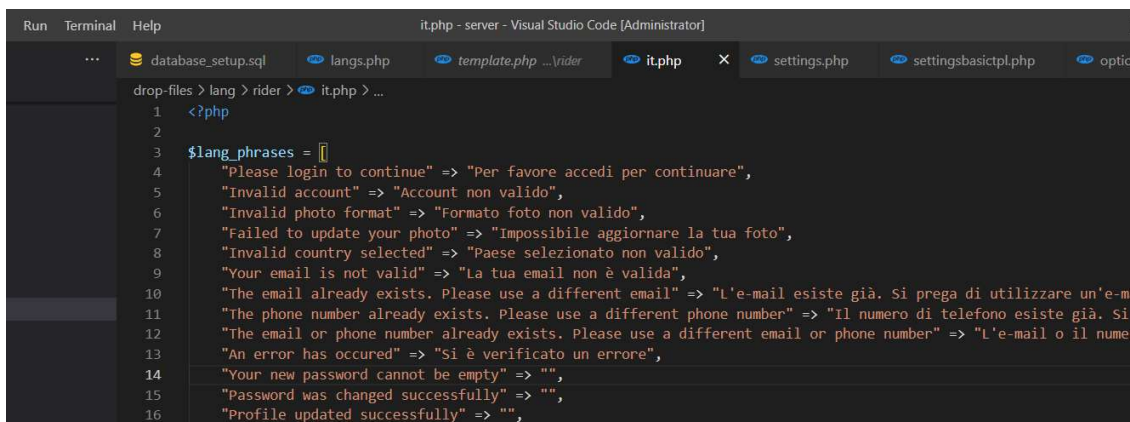
Open the **it.php** file. You will realize the content structure looks like the JavaScript version only this time the colon ":" character has been replaced with this symbol "=>". This is the syntax used in denoting associative arrays in PHP. They are a bit like objects in JavaScript.





```
Run Terminal Help
it.php - server - Visual Studio Code [Administrator]
database_setup.sql langs.php template.php .../rider it.php settings.php settingsbasicpl.php
drop-files > lang > rider > it.php > ...
1 <?php
2
3 $lang_phrases = [
4     "Please login to continue" => "",
5     "Invalid account" => "",
6     "Invalid photo format" => "",
7     "Failed to update your photo" => "",
8     "Invalid country selected" => "",
9     "Your email is not valid" => "",
10    "The email already exists. Please use a different email" => "",
11    "The phone number already exists. Please use a different phone number" => "",
12    "The email or phone number already exists. Please use a different email or phone number" => "",
13    "An error has occurred" => "",
14    "Your new password cannot be empty" => "",
15    "Password was changed successfully" => "",
```

The strings before the “=>” symbol are the keys of the associative array and are in English. They should never be modified. The two double quotes after the “=>” symbol on each line is where the translation of the keys should be inserted. The translations must not contain any double quotes character. The image below shows the first ten lines translated in Italian as per our example.



```
Run Terminal Help
it.php - server - Visual Studio Code [Administrator]
database_setup.sql langs.php template.php .../rider it.php settings.php settingsbasicpl.php options.php
drop-files > lang > rider > it.php > ...
1 <?php
2
3 $lang_phrases = [
4     "Please login to continue" => "Per favore accedi per continuare",
5     "Invalid account" => "Account non valido",
6     "Invalid photo format" => "Formato foto non valido",
7     "Failed to update your photo" => "Impossibile aggiornare la tua foto",
8     "Invalid country selected" => "Paese selezionato non valido",
9     "Your email is not valid" => "La tua email non è valida",
10    "The email already exists. Please use a different email" => "L'e-mail esiste già. Si prega di utilizzare un'e-mail diversa",
11    "The phone number already exists. Please use a different phone number" => "Il numero di telefono esiste già. Si prega di utilizzare un numero diverso",
12    "The email or phone number already exists. Please use a different email or phone number" => "L'e-mail o il numero di telefono esiste già. Si prega di utilizzare un'e-mail o un numero di telefono diversi",
13    "An error has occurred" => "Si è verificato un errore",
14    "Your new password cannot be empty" => "",
15    "Password was changed successfully" => "",
16    "Profile updated successfully" => "",
```

When you are done translating all the keys English strings to your desired language. Save the file.

Some key strings will contain one or more variables that look like this {---1} {---2} {---3} or {---4}. Always ensure the translated string contains the same number of variables as the key.

Next, add a language translation file for the driver server code. Open the driver folder located at:

### **server->drop-files->lang->driver**

Follow the same steps detailed above to add a new language entry in the **langs.php** language descriptor file; duplicate and rename the **template.php** file and then translate the English strings keys of the associative array to your desired language as explained above.

The autodispatch folder contains the translation file used by the Droptaxi auto-dispatch service script. Open the folder to reveal the **dict.php** file. Open dict.php in visual studio code. The content is as shown in the image below:

```

1 <?php
2 $dict_phrases = [
3     'en' => [
4         "New Booking" => "New Booking",
5         "You have a new booking with ID({--1}). Please respond to this booking immediately as your customer is waiting." => "You have a new booking with ID({--1}). Please respond to this booking immediately as your customer is waiting.",
6         "Booking Cancelled" => "Booking Cancelled",
7         "We couldn't locate a driver close to your set pickup location for booking ID({--1}). Please try again later." => "We couldn't locate a driver close to your set pickup location for booking ID({--1}). Please try again later.",
8         "Your Booking with booking ID({--1}) has been cancelled. No driver is available within your set pickup location. Please try again later." => "Your Booking with booking ID({--1}) has been cancelled. No driver is available within your set pickup location. Please try again later."
9     ],
10    'hi' => [
11        "New Booking" => "नई बुकिंग",
12        "You have a new booking with ID({--1}). Please respond to this booking immediately as your customer is waiting." => "आपके पास आईडी ({--1}) का नया बुकिंग है। कृपया तुरंत इस बुकिंग का जवाब दें।",
13        "Booking Cancelled" => "बुकिंग रद्द",
14        "We couldn't locate a driver close to your set pickup location for booking ID({--1}). Please try again later." => "हम बुकिंग आईडी ({--1}) के लिए ड्राइवर नहीं पा सके। कृपया पुनः प्रयास करें।",
15        "Your Booking with booking ID({--1}) has been cancelled. No driver is available within your set pickup location. Please try again later." => "आपका बुकिंग आईडी ({--1}) रद्द हो गया है। ड्राइवर आपके सेट किए गए पिकअप स्थान के भीतर उपलब्ध नहीं है। कृपया पुनः प्रयास करें।"
16    ],
17    'es' => [
18        "New Booking" => "Nueva reserva",
19        "You have a new booking with ID({--1}). Please respond to this booking immediately as your customer is waiting." => "Tienes una nueva reserva con ID({--1}). Responde a esta reserva inmediatamente ya que tu cliente está esperando.",
20        "Booking Cancelled" => "Reserva cancelada",
21        "We couldn't locate a driver close to your set pickup location for booking ID({--1}). Please try again later." => "No pudimos encontrar un conductor cerca de tu ubicación de recogida establecida para el ID de reserva ({--1}). Intenta de nuevo más tarde.",
22        "Your Booking with booking ID({--1}) has been cancelled. No driver is available within your set pickup location. Please try again later." => "Tu reserva con el ID de reserva ({--1}) ha sido cancelada. No hay conductor disponible en tu ubicación de recogida establecida. Intenta de nuevo más tarde."
23    ],
24    'fr' => [

```

Add a new language entry by duplicating the 'en' associative array. Rename key of the duplicate to your language 2-letter language code. For the Italian language example. This will be **'it'**. Replace all strings on each line after the symbol "=>" with the translations string of their respective keys. Ensure lines where there is a variable in the key string, there is also equal number of variables in the translation string. The image below shows how the final file should look:

```

1 t_phrases = [
2     'en' => [
3         "New Booking" => "New Booking",
4         "You have a new booking with ID({--1}). Please respond to this booking immediately as your customer is waiting." => "You have a new booking with ID({--1}). Please respond to this booking immediately as your customer is waiting.",
5         "Booking Cancelled" => "Booking Cancelled",
6         "We couldn't locate a driver close to your set pickup location for booking ID({--1}). Please try again later." => "We couldn't locate a driver close to your set pickup location for booking ID({--1}). Please try again later.",
7         "Your Booking with booking ID({--1}) has been cancelled. No driver is available within your set pickup location. Please try again later." => "Your Booking with booking ID({--1}) has been cancelled. No driver is available within your set pickup location. Please try again later."
8     ],
9     'it' => [
10        "New Booking" => "Nuova prenotazione",
11        "You have a new booking with ID({--1}). Please respond to this booking immediately as your customer is waiting." => "Hai una nuova prenotazione con ID({--1}). Rispondi a questa prenotazione immediatamente poiché il tuo cliente sta aspettando.",
12        "Booking Cancelled" => "Prenotazione annullata",
13        "We couldn't locate a driver close to your set pickup location for booking ID({--1}). Please try again later." => "Non siamo riusciti a trovare un conducente vicino alla tua ubicazione di prelievo stabilita per l'ID della prenotazione ({--1}). Riprova di nuovo più tardi.",
14        "Your Booking with booking ID({--1}) has been cancelled. No driver is available within your set pickup location. Please try again later." => "La tua prenotazione con l'ID della prenotazione ({--1}) è stata cancellata. Non c'è nessun conducente disponibile nella tua ubicazione di prelievo stabilita. Riprova di nuovo più tardi."
15    ],
16    'hi' => [
17        "New Booking" => "नई बुकिंग",
18        "You have a new booking with ID({--1}). Please respond to this booking immediately as your customer is waiting." => "आपके पास आईडी ({--1}) का नया बुकिंग है। कृपया तुरंत इस बुकिंग का जवाब दें।",
19        "Booking Cancelled" => "बुकिंग रद्द",
20        "We couldn't locate a driver close to your set pickup location for booking ID({--1}). Please try again later." => "हम बुकिंग आईडी ({--1}) के लिए ड्राइवर नहीं पा सके। कृपया पुनः प्रयास करें।",
21        "Your Booking with booking ID({--1}) has been cancelled. No driver is available within your set pickup location. Please try again later." => "आपका बुकिंग आईडी ({--1}) रद्द हो गया है। ड्राइवर आपके सेट किए गए पिकअप स्थान के भीतर उपलब्ध नहीं है। कृपया पुनः प्रयास करें।"
22    ],
23 ],

```

Save the file when you are done. Then restart the auto-dispatch service by writing '2' to crondata.txt file as explained earlier.

That's all for setting up Droptaxi 2.2.0!

**We offer complete installation and setup service of Droptaxi software. Customization service is also available. You can get in touch with us through these channels:**

Email: [droptaxisoftware@gmail.com](mailto:droptaxisoftware@gmail.com)

Whatsapp: +2348035631219

Thank you