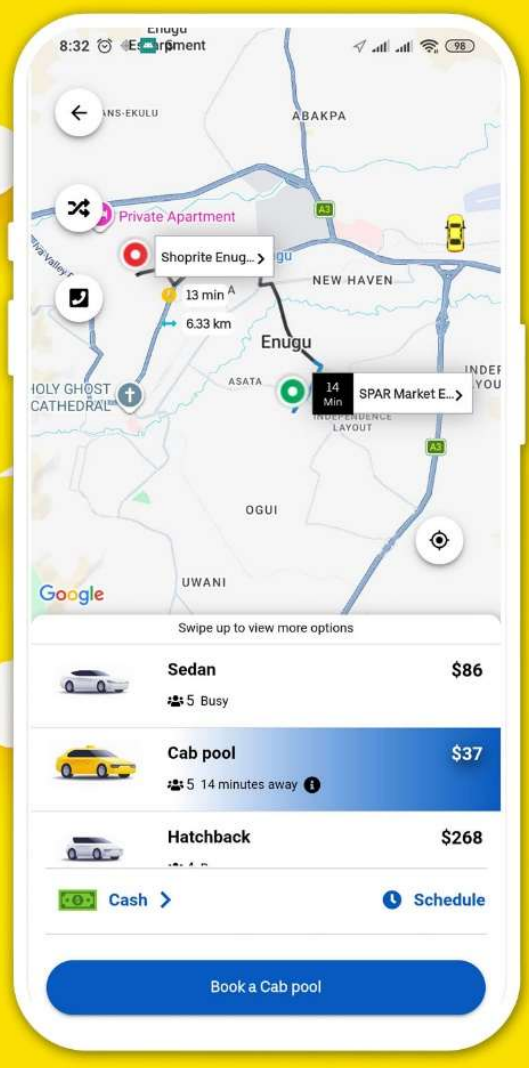
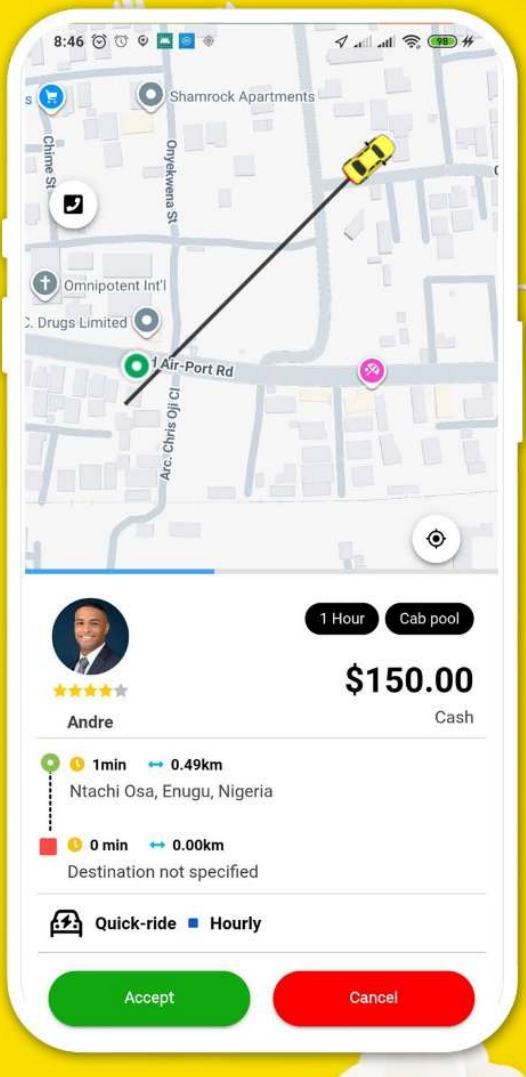




DROPTAXI 2.3

Advanced online taxi software solution

What’s New in Droptaxi 2.3.0?

It is with great joy that we announce the release of the latest version of Droptaxi. We have put in our best to ensure this version of Droptaxi is nothing short of awesome.

Droptaxi is known for its fast and smooth performance, beautiful and fluid UI; this release of Droptaxi builds upon these qualities of Droptaxi. In this release, we have introduced new features, polished the UI and also fixed some bugs. Let’s dive in:

Droptaxi 2.3 for Android now targets Android 15 SDK (API Level 35). You can now update your apps on play store without it being rejected. We have also fixed the 16KB memory page issue in this release of Droptaxi.

Droptaxi 2.3 now includes a new service type. We call it Quick-ride. The Quick-ride service type can be easily accessed from the home screen by tapping the Quick-ride button. Quick-ride allows your customers book trips faster without needing to enter a destination address. Customers need only confirm their pickup location, choose a vehicle and then book the trip. Once a Quick-ride trip is booked, a notification is displayed on the closest driver’s app with details showing that the trip is a Quick-ride trip. Drivers can drive the customer anywhere they want to go. At the end of the trip, the fare is calculated based on the number of hours spent and the cost per hour set.

We have introduced hourly rates in this release of Droptaxi. With hourly rates, you can charge your customers based on the number of hours they spend on a trip. Hourly rates are available through the Quick-ride feature. You can setup hourly rate on the admin panel. Different rates can be set for day time trips as well as night time trips. When hourly rates are disabled for a vehicle category, customers can still book a Quick-ride trip with that vehicle category but they will be billed based on the distance they travel on the trip. This gives you the flexibility to choose which option best suits your business.

Droptaxi 2.3 now includes Wait time feature. Wait time allows you bill customers for the duration of time a driver has to wait for them to start a trip or at a Stop. You can setup Wait times on the admin panel by entering the number of minutes a driver can wait without the customer being charged and the cost-per-minute the customer will

be charge after the wait time elapses. You can setup different wait times for Day time trips as well as night time trips.

In the taxi business, it is highly important to always verify your customers and drivers phone numbers for security purposes. This is usually done using an OTP sent to the user's phone through SMS. Depending on the network conditions, the OTP SMS can be delayed; making it difficult for users to login or signup. This impacts the user's experience negatively. Droptaxi 2.3 now integrates WhatsApp phone number verification as an option for phone number verification. Our implementation of the WhatsApp phone number verification does not cost you a dime and is free to use. When fully setup on the admin panel, and users enter their phone numbers on the rider or driver app, a button is displayed to verify the phone number through Whatsapp. When the user taps this button, a message is sent to a designated Whatsapp phone number. If the phone number entered on the app matches the user's Whatsapp phone number, the user is automatically granted access on the app. It is that simple!

We have improved the way Stops are handled on Droptaxi. Stops are now displayed clearly on the rider and driver app maps during a trip. Customers also get to see information on when a driver arrived a stop, leaves a stop for the next stop and when the normal trip is resumed. Wait time works with Stops too. On arrival of a Stop, the wait time countdown starts, when the trip resumes, the wait time countdown is stopped and is stored. At the end of the trip, the wait time cost is added to the final fare.

Droptaxi 2.3 now has a redesigned scheduled trip page. The new design allows drivers access more information about a scheduled trip before accepting the trip. In Droptaxi 2.3 it is now possible for drivers to view expanded details of a scheduled trip pickup and drop-off locations on a mini map. If Stops are added they will appear on the map. The mini map can be zoomed in / out and panned for a clearer view. The mini map is also available on the trip history page on the rider and driver apps.

We have redesigned the trip request notification screen on the driver app. Drivers can quickly get details about an incoming trip request at a glance. The trip request notification screen now shows the path of the trip from the driver's current location, to the rider's pickup location and to the rider's drop-off location. Also shown is the

distance and travel time of the trip as well as the distance and travel time of the driver from his current location to the customer's pickup location. Drivers can also see if a trip is a long trip or a short trip. If the trip request is a Quick ride, it is also shown together with the number of hours the customer has set to be driven.

Droptaxi 2.3 now automatically puts the driver offline if the amount of money in his wallet drops below the threshold you have set in the admin panel. Once offline, the driver cannot go online except he funds his app wallet.

In this release of Droptaxi, drivers with a positive balance in their app wallet will be able to accept trips even if the amount of money in the driver's app wallet is lower than the company commission percentage of the trip. Once the driver completes the trip and the company commission is deducted, his wallet balance will go negative. Drivers with a negative wallet balance below the minimum set wallet amount threshold cannot receive trip requests and will be put offline automatically.

We have fixed an issue in alternate vehicles feature where a customer is billed using the tariff structure of the alternate vehicle instead of the initial vehicle the customer booked the trip with. Now when customers book a trip using a vehicle category and that vehicle category isn't available, if alternate vehicles are setup for that vehicle and they are available, the customer will be assigned a driver using the alternate vehicle. At the end of the trip, the fare is calculated using the tariff of the initial vehicle category the customer booked the trip with rather than the alternate vehicle as it is done in previous versions of Droptaxi.

As part of our way of extending the already awesome user experience in Droptaxi, we have added a help tips feature. Help tips are used to display information to the user about a function or feature they are about to access. You can stop a Help tip from being displayed again by checking the "Do not show this again" checkbox on the help tip dialog. We will be using Help tips more in subsequent versions of Droptaxi.

In Droptaxi 2.3 we have added the ability to track the driver location and travel route in near real-time on the view booking details page of the admin panel. When you view details of an active trip, you will now see the current driver vehicle icon animated as the driver travels as well as his travel route.

On the admin panel dashboard, we have added a booking trends bar chart. This chart displays the number of bookings, cancelled and completed over a period of 7 days (1 week).

We have made several performance improvements to the backend server code of Droptaxi in this release to keep Droptaxi running fast and stable. The no-framework / native backend code development ensures we are able to squeeze out more performance from a server compared to other software built using a framework. For us, every microsecond delay in code execution counts. We also continue to leverage the high performance of Redis for a fast and efficient operation of Droptaxi.

So, that's it for Droptaxi 2.3. There are several other tweaks which we have made to Droptaxi; you will notice these as you use Droptaxi. We truly hope you will love this new version of Droptaxi and it helps take your businesses to greater heights.

Once again, we appreciate all Droptaxi customers who have stood by us over all these years and to those who have just discovered Droptaxi, We say thank you.

Michael Chike
Lead Software Developer
Droptaxi

REQUIREMENTS

This documentation refers to Droptaxi version 2.3.0.

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

1. A Linux VPS running Ubuntu 24.04 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 8.13** and **Latest version of Android Studio** installed and setup. Ensure you include the required SDK tools in Android studio under SDK manager - **Android SDK Platform 15.0 ('VanillaIceCream) and Android SDK Build tool 35.0.0 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 16.3** 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, Places API (New), Routes 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 2.2.1

Follow these steps to upgrade your server from Droptaxi 2.2.1 to Droptaxi 2.3

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 2.3 schema file into it. You will find the Droptaxi 2.3 database setup /schema file in Droptaxi 2.3 package; **drop-files -> install -> database_setup.sql**. Do not delete your old database.

3. After importing the Droptaxi 2.3 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 2.3 server files and copy the drop-files folder to the location where you deleted the old drop-files folder. Copy all the Droptaxi 2.3 files in the public folder into your server **public_html** folder.

8. Access the "**old_backup**" and copy the upload folder and **credentials.json** file found inside the **drop-files** folder. Replace the Droptaxi 2.3 uploads folder and credentials.json file on your server with it.

9. Open the Droptaxi 2.3 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 2.3 ride_imgs folder with it.

11. Access the "old_backup" and copy **ajax_2_2_1.php** and **ajaxdriver_2_2_1.php** files found in public_html folder into the Droptaxi 2.3 public_html folder on your server.

12. 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 2.3 settings data files files on your server with them.

13. Access the new admin panel through your browser.

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

15. Once activated, go to settings page, access each of the settings tabs and ensure all your previous settings are available and then click the save button.

BACKEND SERVER SETUP

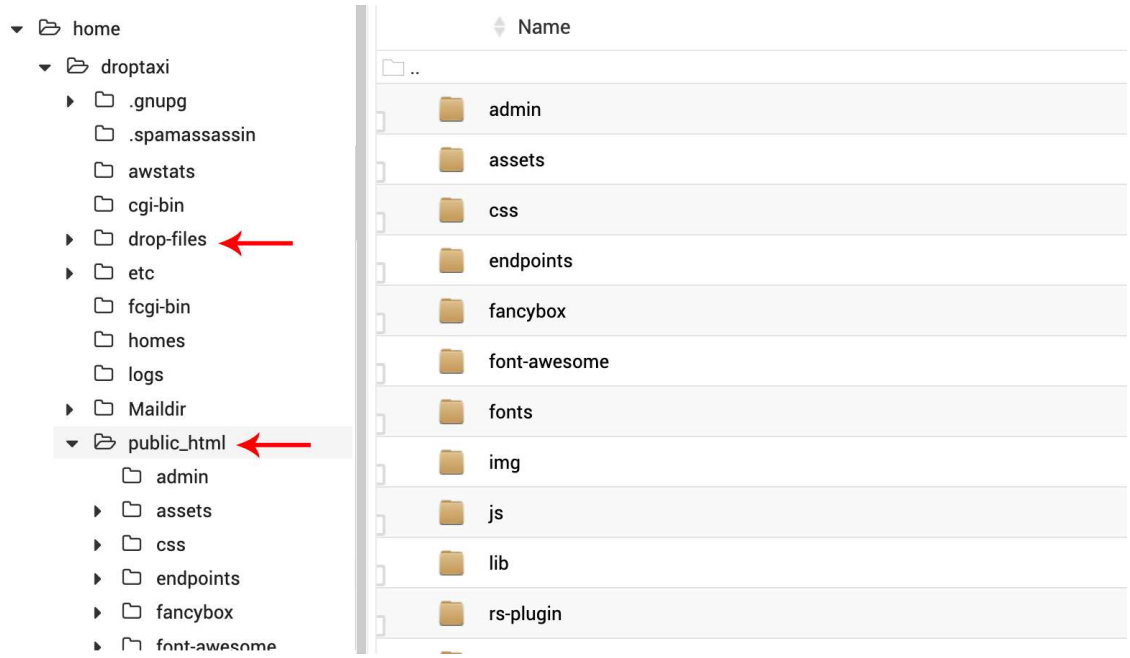
Droptaxi backend is written in core PHP. Ensure you have your VPS setup and running **Apache 2.4, PHP 8.1, MYSQL 8** or higher and **Redis 7.2 or higher**. We have integrated Redis in-memory data store into the Droptaxi tech stack to help with fast data access and also free up MySQL database on certain 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 15** 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 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 8 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. 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

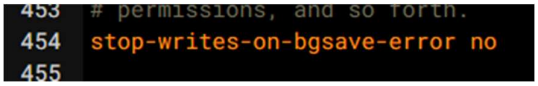
sudo systemctl restart redis-server

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

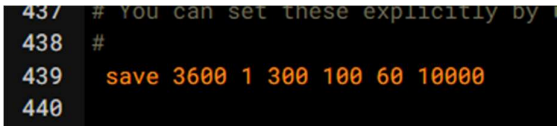
Make the following changes to the redis.conf file:

Enable writes on background save errors



stop-writes-on-bgsave-error no

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



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

maxmemory 400000000

Save the redis.conf file.

Retart 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 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 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 four Google map API keys: two for the server, one for the Android rider and driver apps, and the other one 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
    - Places API (New)
    - Routes 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 four required API keys.
- It is required you restrict your API keys to avoid theft and unauthorized use. First apply API restrictions to the two keys that will be used for the Android and iOS driver and rider apps. Click the three dots next to each of the API keys then click on Edit API key. On the Edit API key page, under API restrictions section, click the Restrict key radio button. On the Select APIs list enable only Map SDK for Android and Map SDK for iOS APIs. Disable all other APIs.
- Apply IP address restriction for one of the two remaining keys. On the Edit API keys page, under Application restrictions, click the IP addresses radio button. Click the Add+ button and enter your server IP address. This API key will be used by Droptaxi for backend Google maps API calls.
- Finally, Apply domain restriction to the remaining key. On the Edit API keys page, under Application restrictions, click the Websites radio button. Click the Add+ button and enter your domain name. This API key will be used by Droptaxi for rendering maps on the admin panel front end / client area.

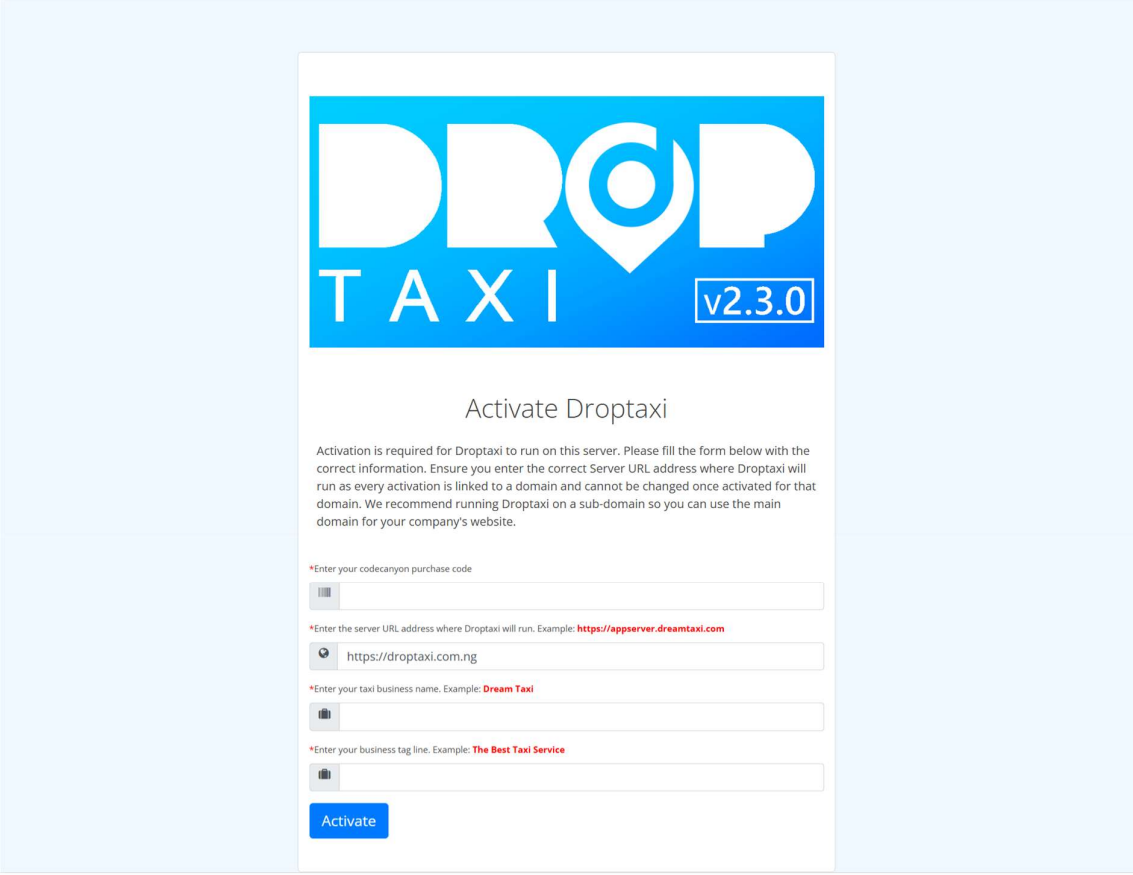
Now you have all the required keys from third party services integrated in Droptaxi you can now proceed to configure the backend.

ACTIVATING DROPTAXI

We have implemented a security feature that limits piracy or unauthorized 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.

When you purchase Droptaxi 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.



Follow the instructions on the activation page to activate Droptaxi on your domain. Enter your 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

Password: Droptaxi

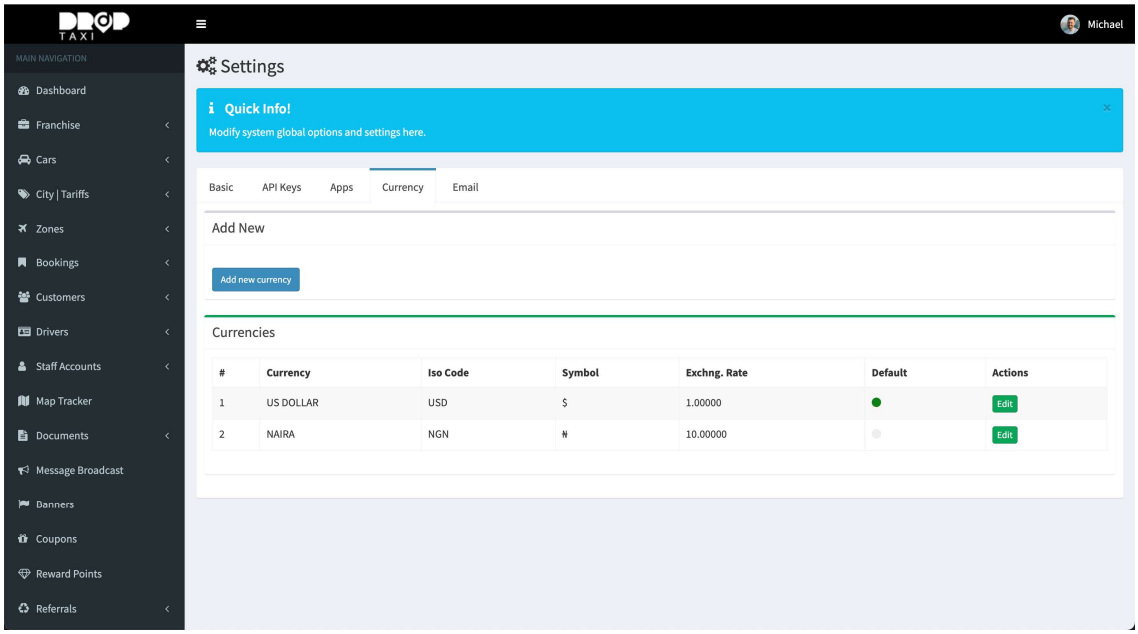
BACKEND SETTINGS

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, Whatsapp, 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 keys. Enter the Google maps API key which you applied domain restrictions earlier to the field labelled **Google Maps API Key (Client)**. In the field labelled **Google Maps API Key (Backend)** enter the Google maps API key which you applied IP restriction.

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.

Note: To enable Google and Apple reviewers test your apps during review you should set the Default SMS service to "TestOTP". After review you can change it back.

Whatsapp phone number verification provides and alternative means for users verify their number. This comes in handy when SMS OTP verification isn't available. Follow the instructions in the next section to properly setup Whatsapp phone number verification on your server.

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

Click the Save button when done.

WHATSAPP SETUP

Whatsapp phone number verification in Droptaxi makes it possible for your users to verify their phone numbers fast and easy using the Whatsapp app installed on their phones. Once a phone number is verified on Whatsapp, the user is automatically logged in or redirected to sign up on the app. Setting up Whatsapp to work on your system requires you to have a Whatsapp phone number, Phone number ID, Whatsapp access token. Follow these steps to setup Whatsapp:

1. Login to your Facebook developer account. Register a developer account if you don't have one. Visit https://developers.facebook.com/apps

2. Click the "Create Apps" button.

3. On the page that asks what you want your app to do, Select Other option at the bottom of the page and click next.

3. On the "Select an app type" page. Select Business and click next.

4. On the next page, click create app button to create your Facebook app.

5. On the App dashboard, add Whatsapp to your app by clicking on the setup button. If you don't have a business portfolio, you will be asked to create one.

6. Once added you will see WhatsApp menu item on the sidebar menu. Click on "Start using the API" button. You can also select the "API Setup" menu item on the left sidebar menu.

7. Click the configure webhook button to setup your webhook. On the next page, Enter https://yourdomain/wtsappnotify.php as your callback url and enter 12345678 as your verify token. Click the "Verify and save" button. A list of webhook fields will be displayed. Locate "messages" on the list and click the subscribe switch to subscribe to messages webhook notifications.

8. Click the "Add phone number" button to add your phone number. Ensure you verify the phone number. After verification. **Copy and save the Phone number ID**.

9. Click the "Add payment" button to add your credit card.

10. Generate an access token that will be used to send messages. Visit https://business.facebook.com/settings

11. On the left sidebar menu, click "System users" menu item.

12. On the System users page, click the +Add button. On the create system user dialog, enter a name for the system user such as droptaxiwhatsappsender. In the System user role dropdown menu, select Admin. Click the "Create system user" button to create a system user button.

13. After creating the system user, Click the "Assign assets" button to assign the app you created earlier to the system user.

14. In the "Select and assign permissions" dialog, Select the Apps assets type. Click the checkbox to check the app you created earlier in the list. In the Assign permissions section under Full control section, turn on Manage app. Click the Assign assets button.

15. In the "Select and assign permissions" dialog, Select the WhatsApp accounts assets type. Click the checkbox to check the app you created earlier in the list. In the Assign permissions section under Full control section, turn on Manage WhatsApp business accounts. Click the Assign assets button. You may need to refresh the page for your changes to take effect.

16. Click the "Generate token" button to generate an access token for the system user.

17. In the Generate token dialog, on the select app page, select the app created earlier. Click Next.

18. In the Set Expiration page, select "Never" for token expiration. Click Next

19. In the Assign permissions page, Click the select permissions dropdown and select whatsapp_business_messaging from the list. Click Generate token button to generate the token. **Copy the generated token and save it somewhere**.

20. Click Done to finish.

21. Switch from Development mode to Live mode by clicking on the "App mode" switch at the top of the screen.

22. Return to Droptaxi admin panel settings page, API tab, and enter the Whatsapp token, Phone number ID and your Whatsapp phone number (International format) in the appropriate fields.

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.

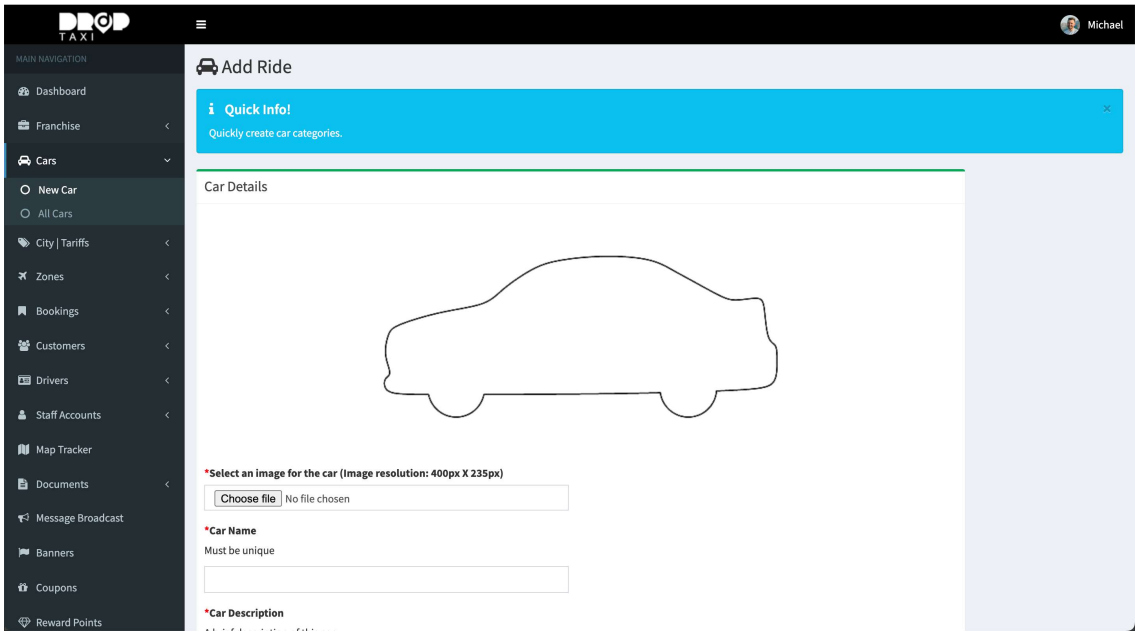
Enter a Sender’s Email. This will appear as the sender of the email. If you are using Gmail SMTP you may have to use your Gmail username here.

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

Go through all the tabs in the settings page and 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.



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

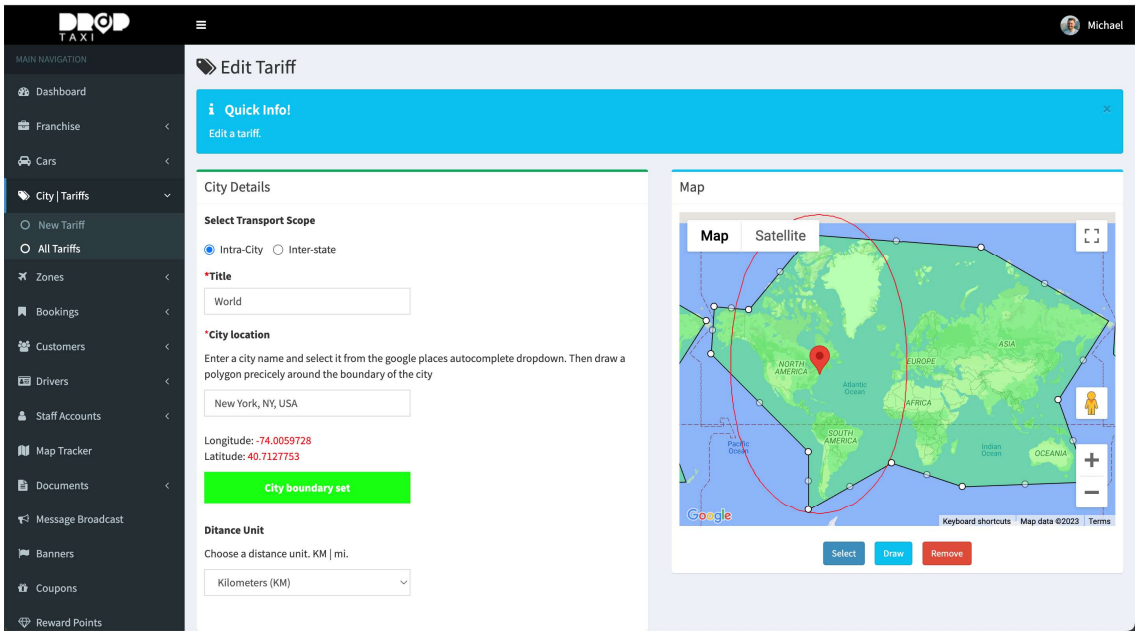
Here you'll setup cities where your taxi service will be available including the types of vehicles available in the city, city currency, ride fares etc.

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 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.

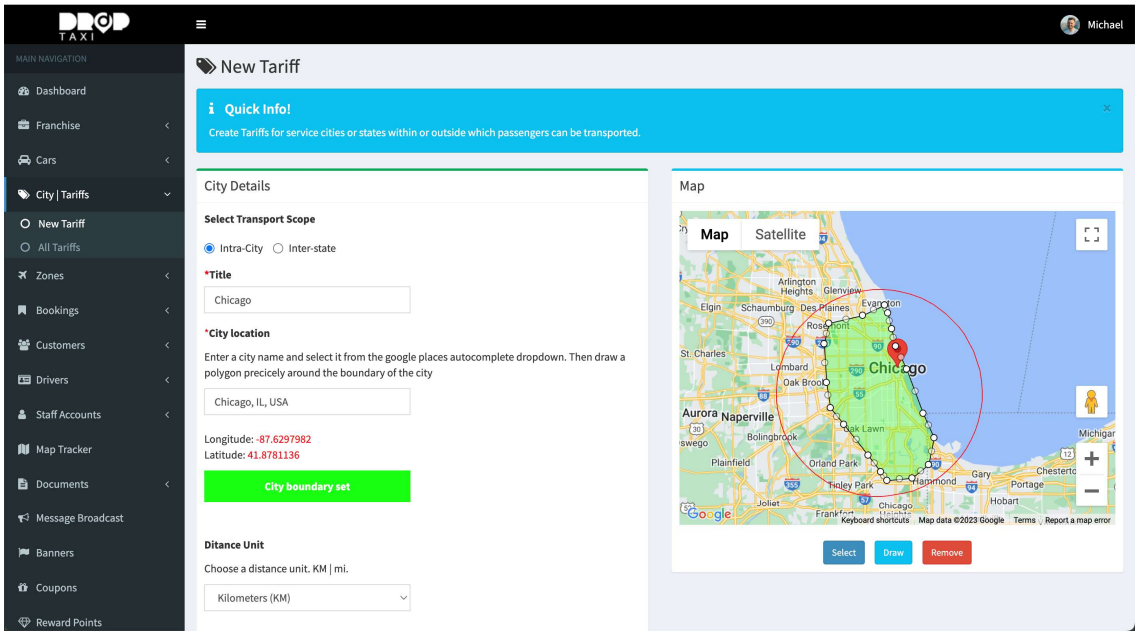


Droptaxi includes the ability to define a boundary around a city. In previous versions of Droptaxi, the rider has to manually select his city on the rider app. In Droptaxi, this is done automatically with the help of city boundaries. It makes it possible to precisely detect the current city of a rider 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:

Compute Fare with Trip Meter: Enabling this option for any vehicle causes Droptaxi to use the actual distance measured by the trip meter on the driver app to calculate the trip fare at the end of a trip. Disable it to use fixed fare pricing calculated using Google maps API.

Enable Ride Sharing: Enabling the option for any vehicle will allow the drivers under the vehicle category to service multiple trips at the same time. Customers will share the vehicle with other customers thus reducing costs.

Enable Hourly Rate: Enabling this option for any vehicle will allow customers to be billed on an hourly basis when they book a **Quick-ride**. Quick-rides are rides where the customer did not specify his destination location. When enabled you will have to enter hourly fares and distance for day time as well as night time. When disabled, customers are billed based on the distance and time they traveled.

Alternate vehicles: When vehicles under the alternate vehicles list are checked; when riders choose the primary vehicle category when booking a trip on the app and the vehicle is not available, drivers using any of the alternate vehicles checked will be allocated the ride if available.

Normal Fare Setup

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

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)

Wait time

Enable Wait Time billing by entering a number value (minutes) greater than zero in the wait time input field. Then enter the Wait time cost value in the wait cost per minute field. When the number of minutes you entered in the wait time field elapses, customers will be billed for the additional time the driver has to wait for them. Wait time countdown is started when the driver arrives the customer pickup location and also when the driver arrives a Stop. The countdown is stopped when the driver begins a trip or when the driver resumes the trip after arriving a Stop.

Peak / Surge Pricing

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, interstate bookings 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

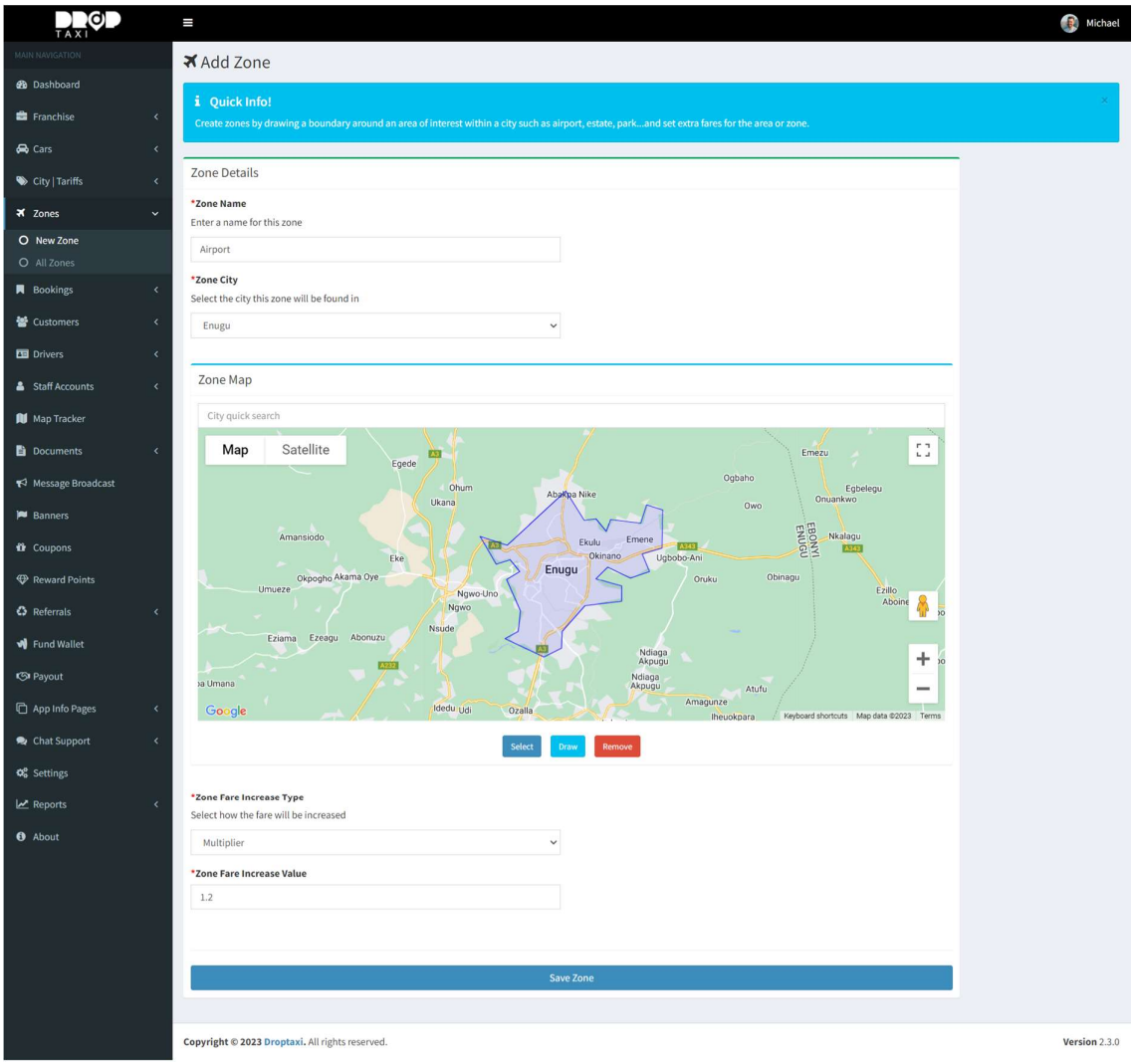
Droptaxi allows you target any location of interest within a city and set a fare for that location. These areas are called zones. When a rider sets a pickup or drop-off location within a zone, the trip fare is increased by the amount you set for that zone.

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.



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 22.19 and includes npm 10.9.3).

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 **Gradle8.13** 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-8.13'. 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-8.13** to

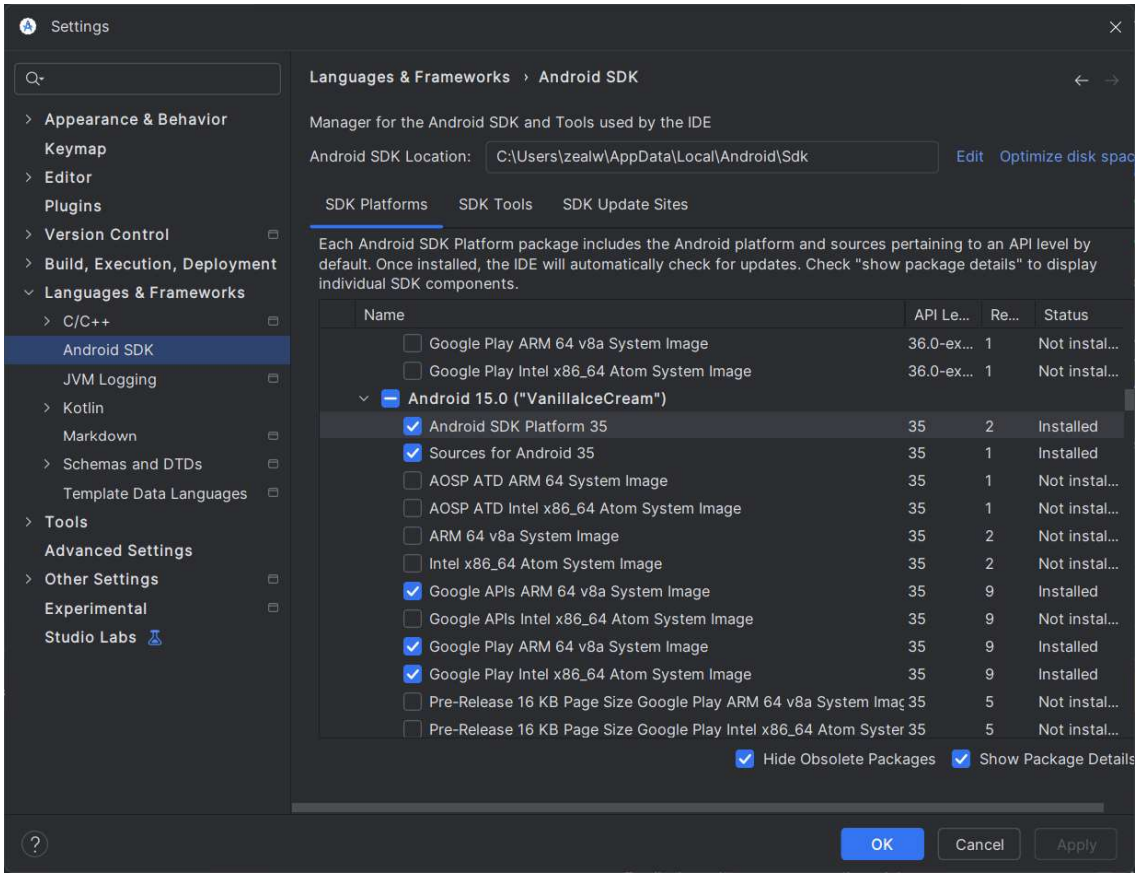
gradle. NB: If you have gradle previously installed, delete the **.gradle** folder located at C:\Users\[xxxxx]\

Download and install android studio:

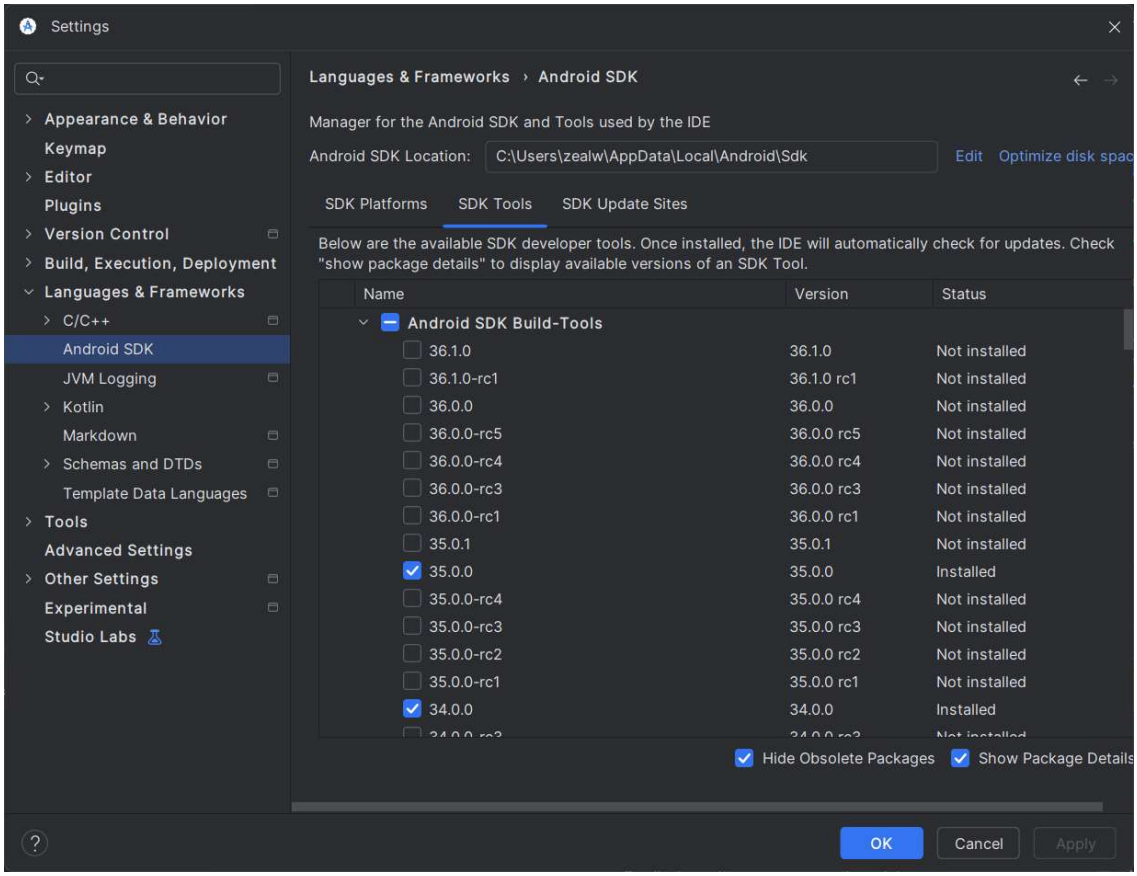
https://developer.android.com/studio

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.

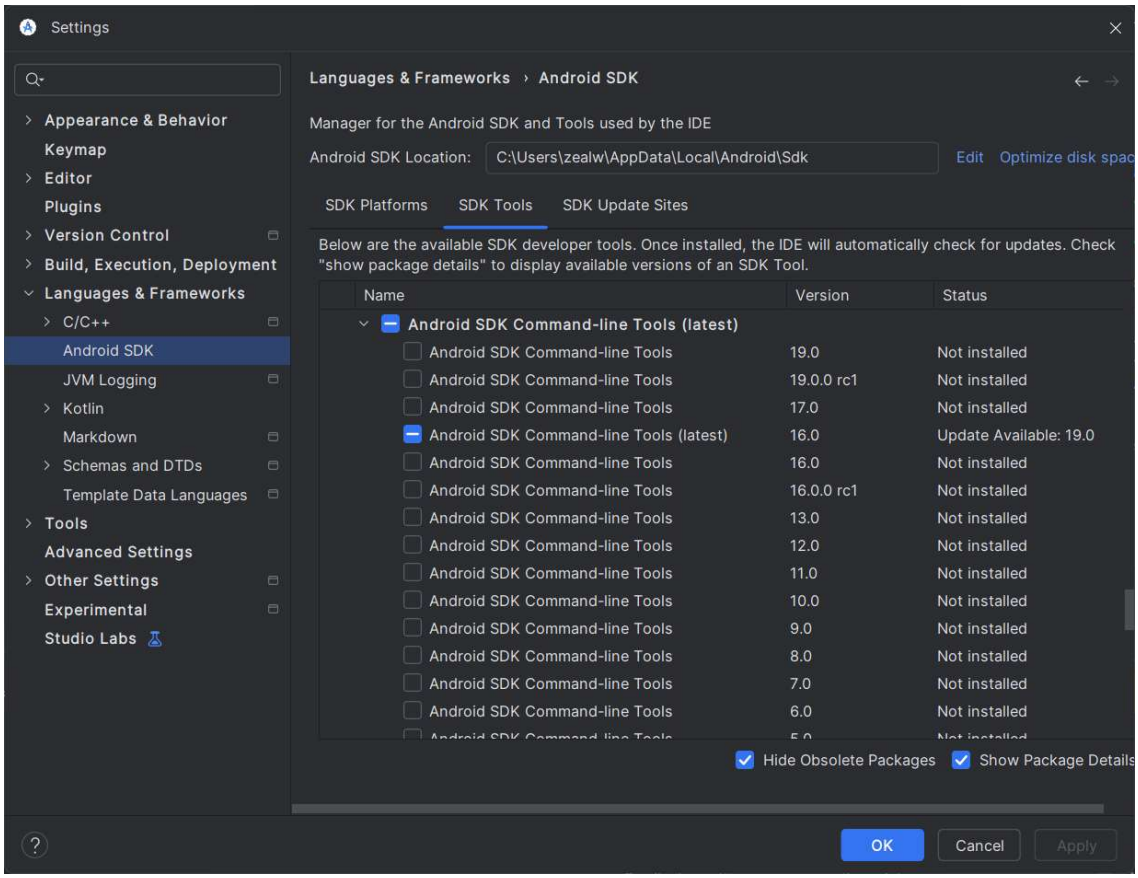
Under SDK platforms tab, from the list, locate and check the android platform target version **Android 15.0 ("VanillaIceCream")** as shown in the image below:

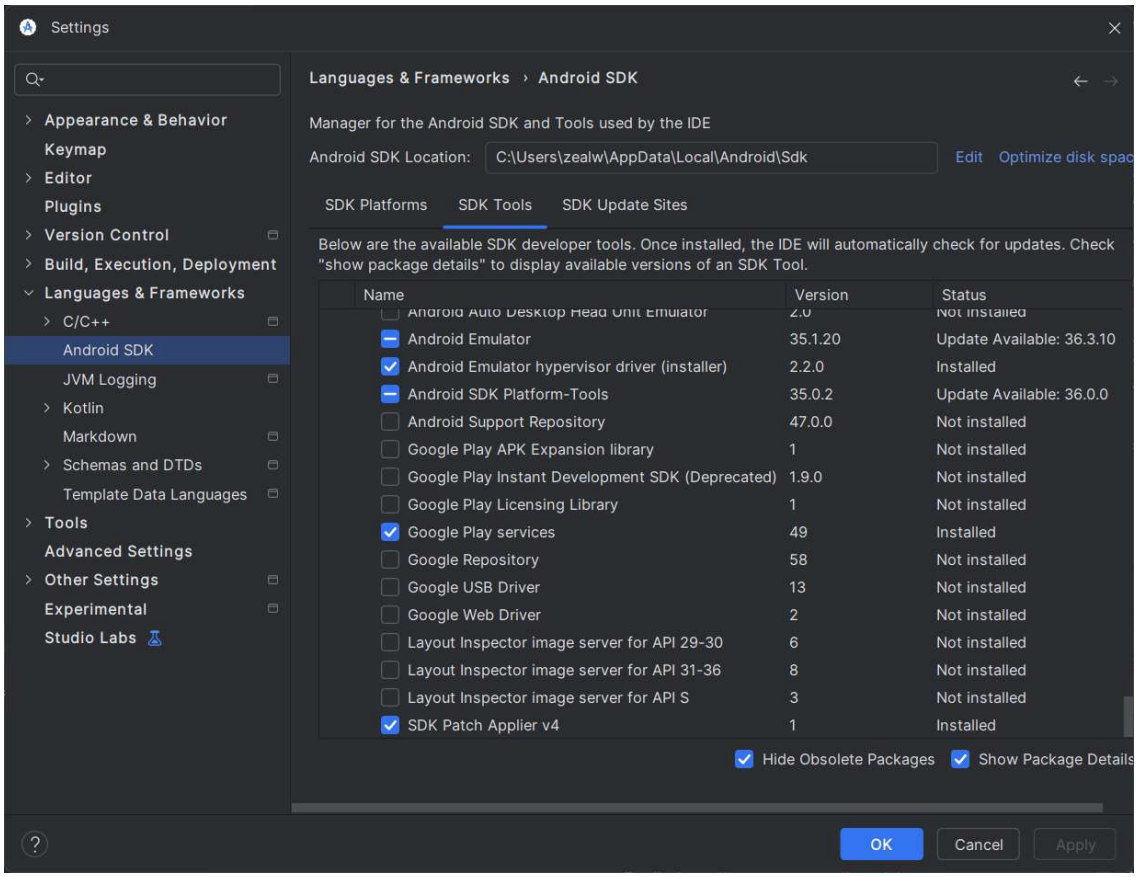


Select the SDK tools tab. On the dialog uncheck '**Hide Obsolete Packages**' and check '**Show Package Details**' checkbox. From the list, check build tool version 35.0.0 as shown in the image below:



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

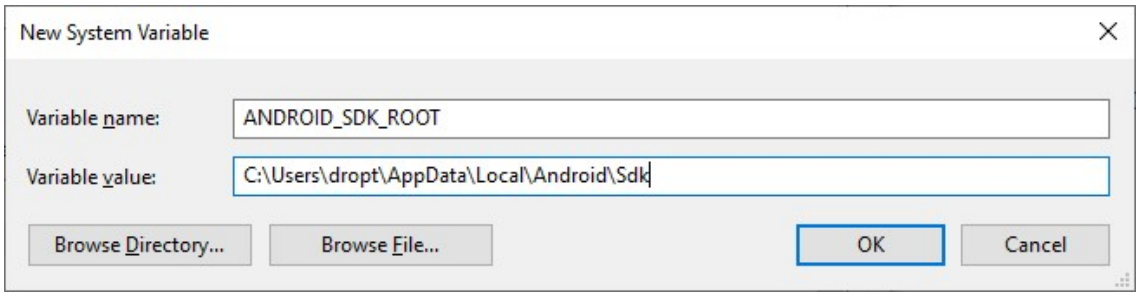

Settings
Languages & Frameworks › Android SDK
Appearance & Behavior
Keymap
Editor
Plugins
Version Control
Build, Execution, Deployment
Languages & Frameworks
C/C++
Android SDK
JVM Logging
Kotlin
Markdown
Schemas and DTDs
Template Data Languages
Tools
Advanced Settings
Other Settings
Experimental
Studio Labs
Manager for the Android SDK and Tools used by the IDE
Android SDK Location: C:\Users\zealw\AppData\Local\Android\Sdk
Edit
Optimize disk spac
SDK Platforms
SDK Tools
SDK Update Sites
Below are the available SDK developer tools. Once installed, the IDE will automatically check for updates. Check "show package details" to display available versions of an SDK Tool.
Name
Version
Status
Android SDK Command-line Tools (latest)
Android SDK Command-line Tools 19.0 Not installed
Android SDK Command-line Tools 19.0.0 rc1 Not installed
Android SDK Command-line Tools 17.0 Not installed
Android SDK Command-line Tools (latest) 16.0 Update Available: 19.0
Android SDK Command-line Tools 16.0 Not installed
Android SDK Command-line Tools 16.0.0 rc1 Not installed
Android SDK Command-line Tools 13.0 Not installed
Android SDK Command-line Tools 12.0 Not installed
Android SDK Command-line Tools 11.0 Not installed
Android SDK Command-line Tools 10.0 Not installed
Android SDK Command-line Tools 9.0 Not installed
Android SDK Command-line Tools 8.0 Not installed
Android SDK Command-line Tools 7.0 Not installed
Android SDK Command-line Tools 6.0 Not installed
Hide Obsolete Packages
Show Package Details
OK
Cancel
Apply



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.

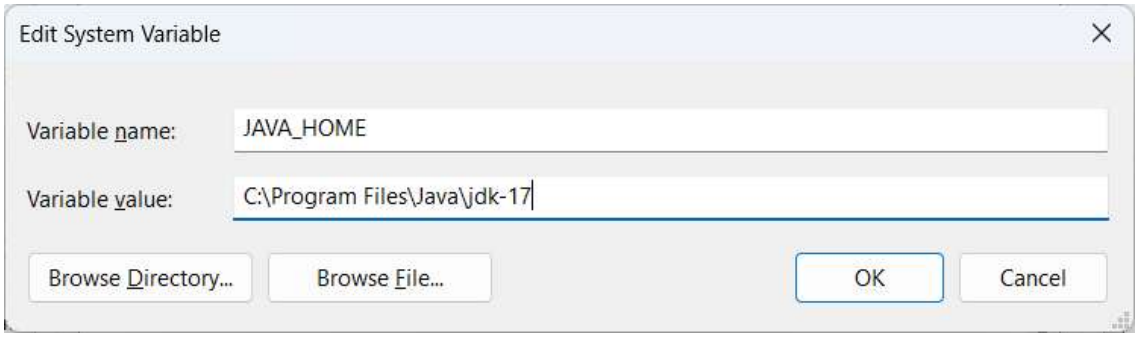
Next we will setup environment variables. On the windows taskbar, click the Start button and type 'environment variable'. On the search result list, click the 'Edit the system variables (control panel)' item. The system properties dialog is displayed. Click on the 'Environment Variables...' button.

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



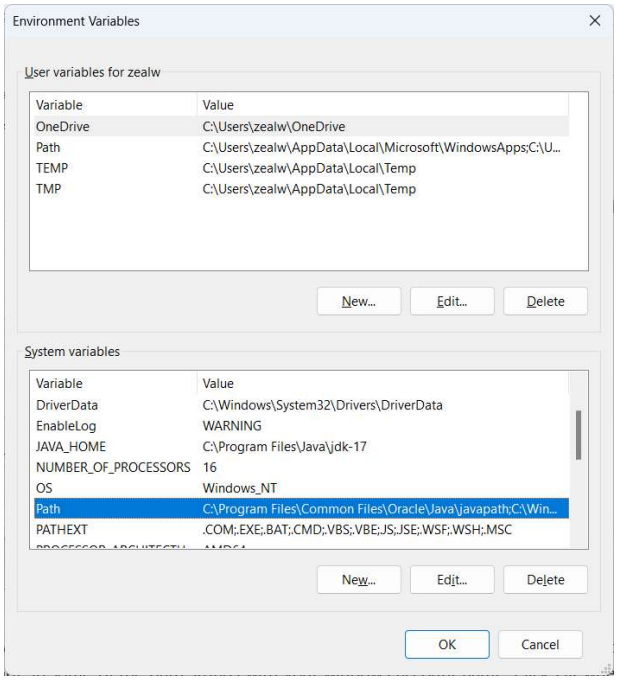
Where [xxxxx] is the Windows user account name. You can use the 'Browse directory' button to navigate to the Android SDK installation folder following the path example. Click OK when done to save your new system Variable.

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

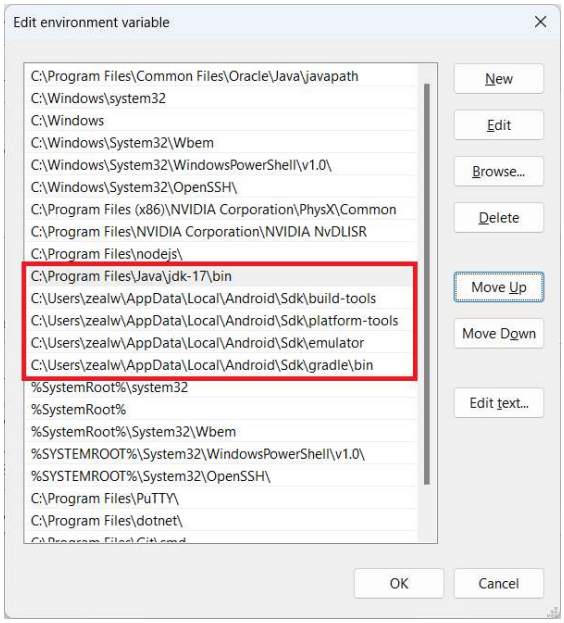


Your java version might be a bit different from this, use the 'Browse directory' button to navigate to the java installation folder. Click OK when done to add the new environment variable.

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.

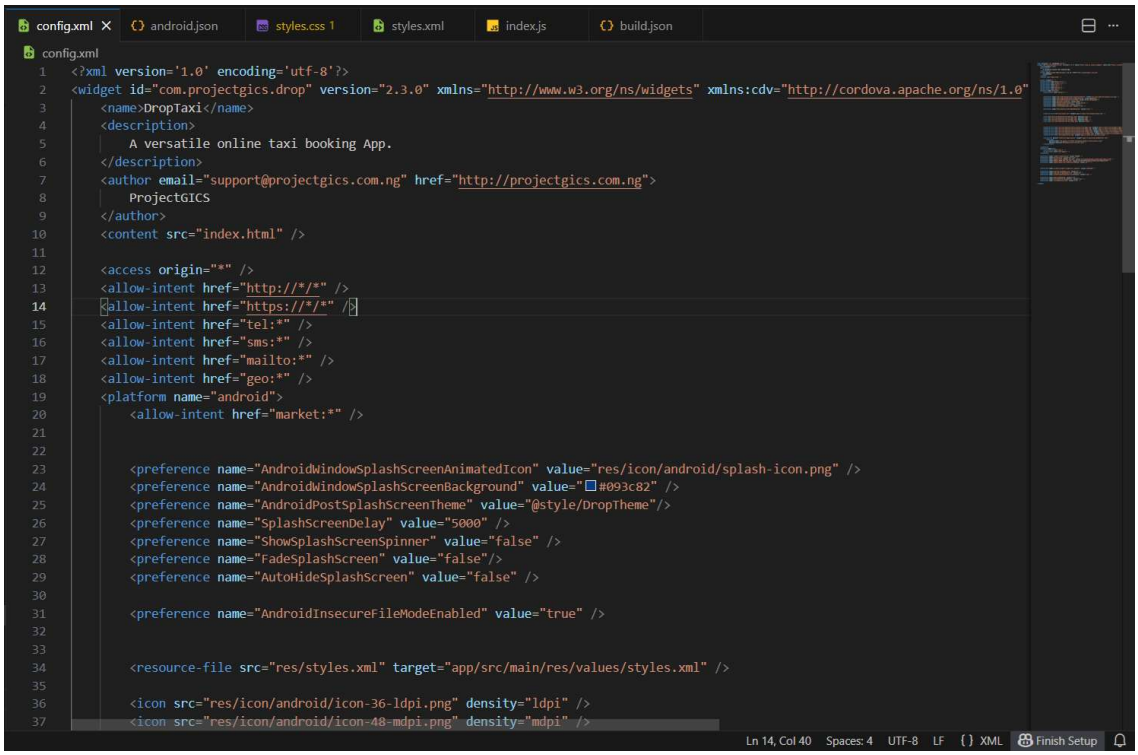
Visual studio code is required to modify some Droptaxi source files as well as running some commands through the integrated terminal. Download and install Visual Studio code through this link:

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. 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.



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

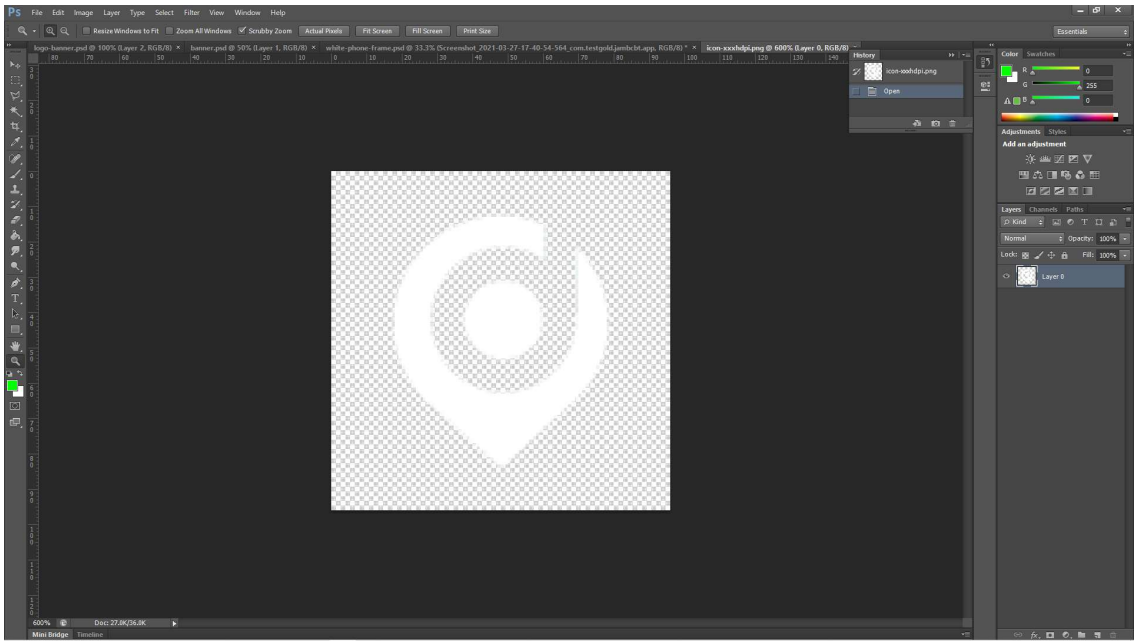
At the top of the file are the variable declarations. Under APP_TITLE enter your app name in quotes E.g "Dreamtaxi". Under APP_VERSION_ANDROID enter the app version you entered in config.xml file. Under 'siteurl' variable enter the url of the backend server E.g https://dreamtaxi.com (Do not include a 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.

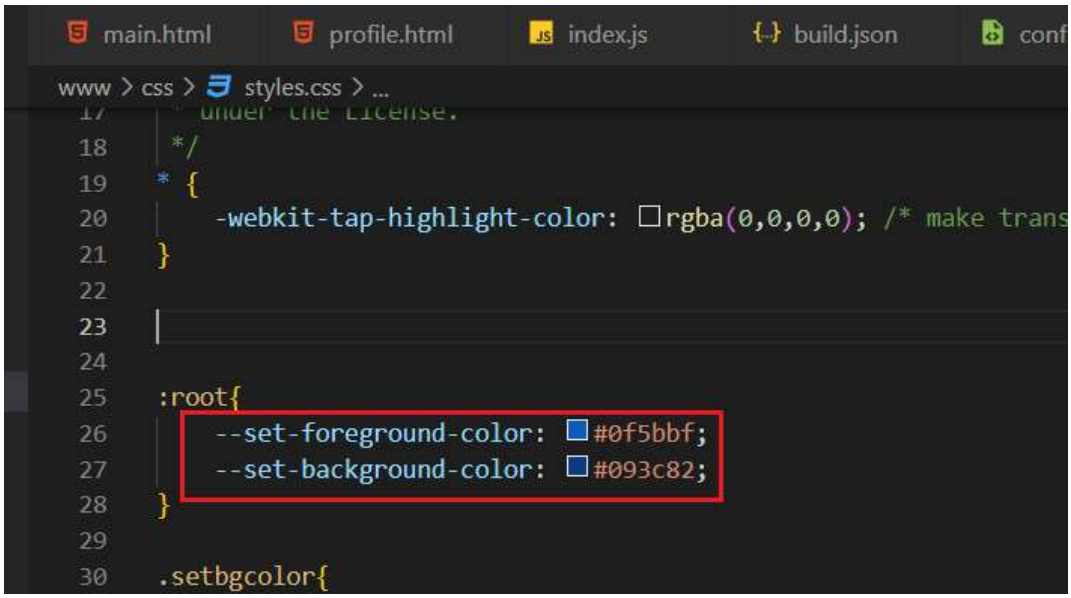
For the icon images, replace the image files in **android->rider->res->icon->android** folder with your own images. Ensure you maintain their respective filenames and resolution.

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.



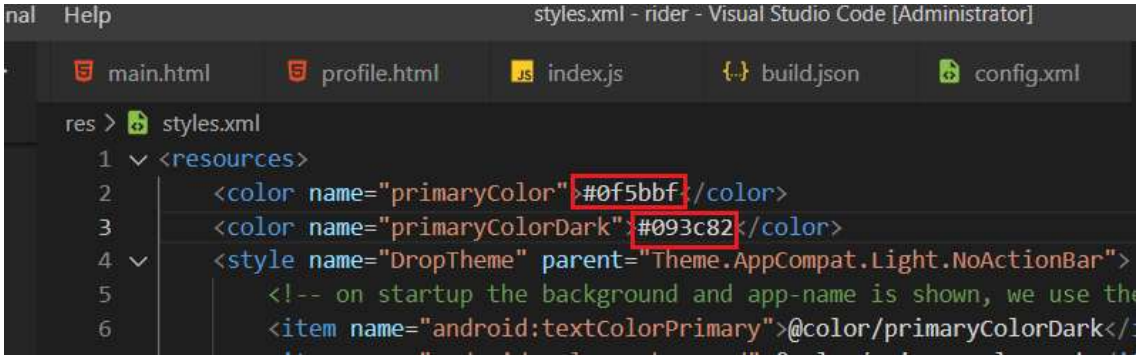
Change the app logo by replacing the logo.png file located at **android->rider->www->img->logo.png** with your own logo file. Use a 200 X 200 pixel resolution logo.

Droptaxi 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.



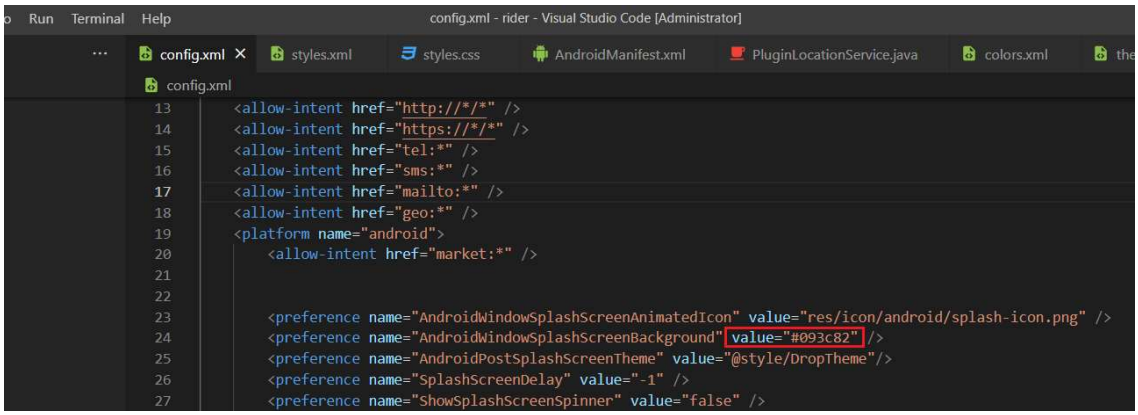
Locate the :root css selector and replace the values of the variables –set-foreground-color and –set-background-color with your desired hex color values. Save the file when done. Hint: always use darker shade colors for the background color and lighter shade colors for the foreground color.

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



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.



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 'drematax' as an example. You can replace it with you 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 l**ocated 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.

```
{
    "android": {
        "release": {
            "keystore": "dreamtaxi.keystore",
            "storePassword": "12345678",
            "alias": "dreamtaxi-app-key",
            "password" : "12345678",
            "keystoreType": "",
            "packageType": "apk"
        }
    }
}
```

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.

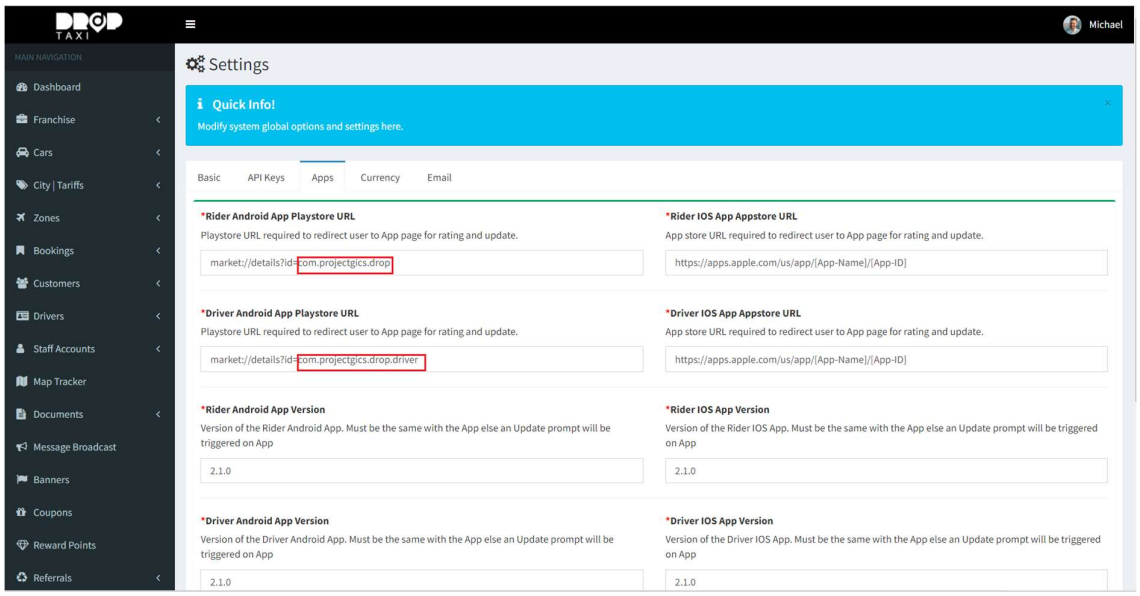
Open the file located at **android/driver/platforms/android/android.json**. Look for all occurrences of "**com.dreamtaxi.driverapp**" and rename it to your driver app package name.

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 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 16.3 was used to build Droptaxi 2.3. 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.

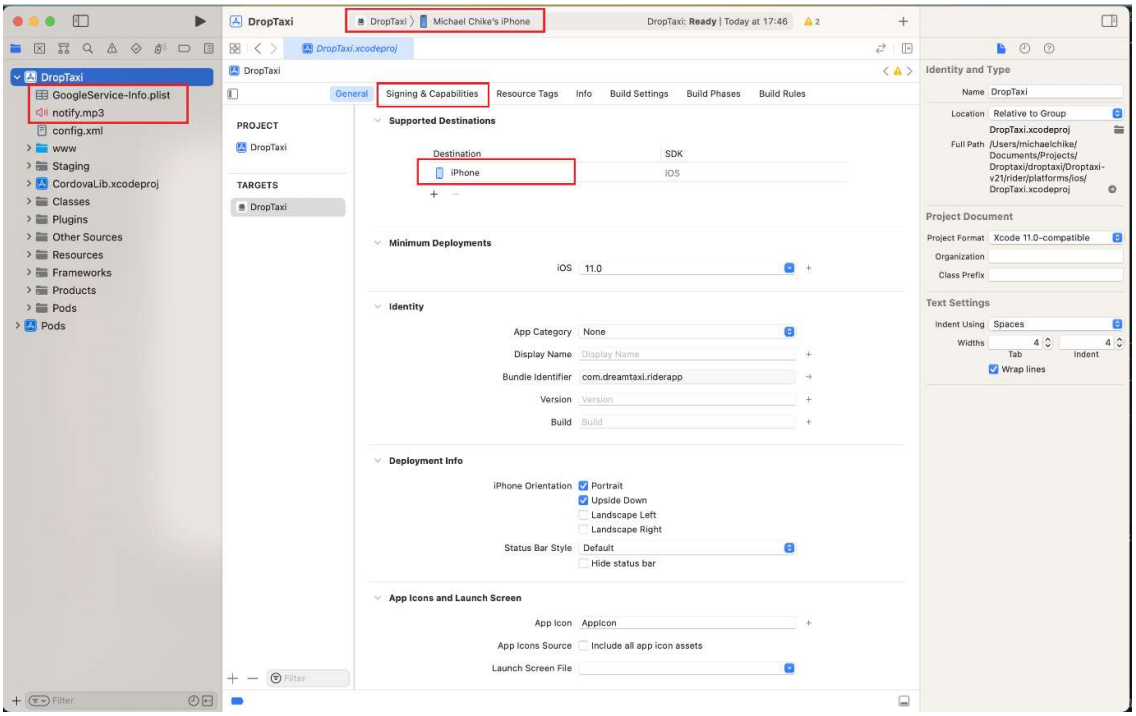
Setup firebase to use Apple push notification service (APNs). In your Apple developer account dashboard, on the left side menu, select certificates, Ids and profiles -> keys.

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.

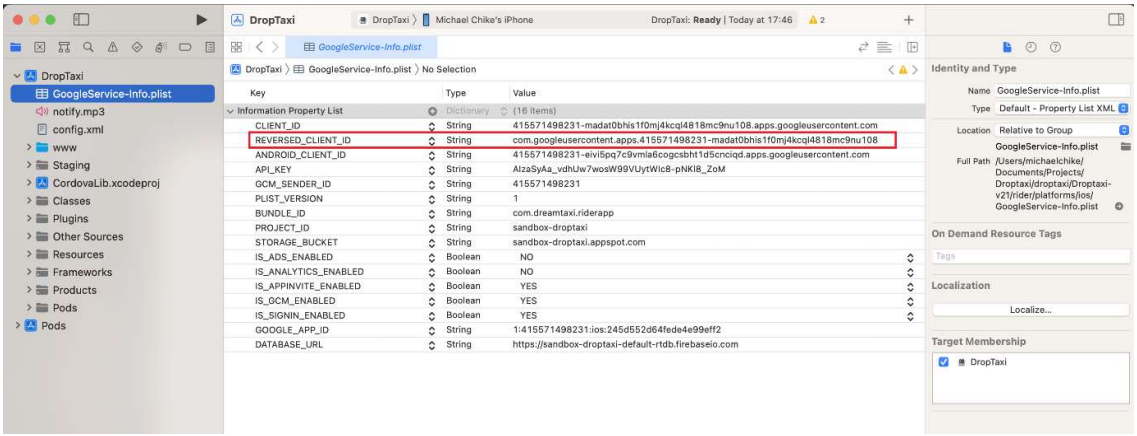
Open the [MyAppName].xcworkspace file located at **ios->rider-platforms->ios** folder in Xcode. On the Navigation panel, drag and drop the GoogleService-info.plist file into the project folder. A popup dialog will appear. Ensure Destination checkbox is checked. Also on Add to targets, ensure the checkbox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



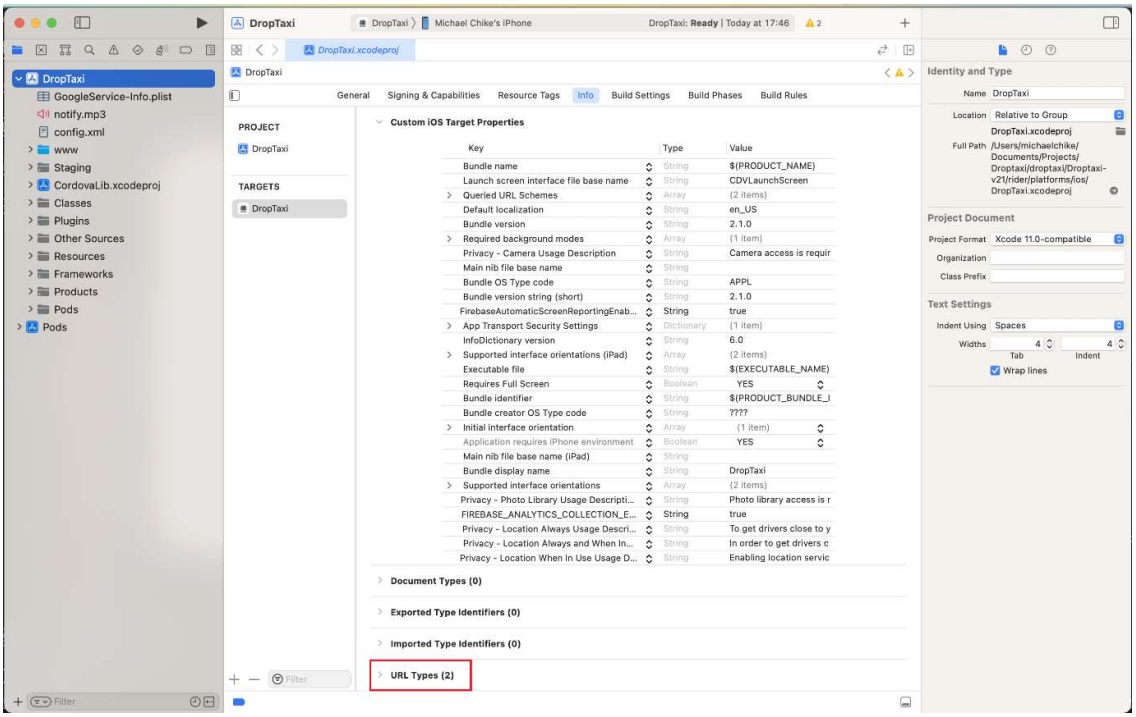
Also drag and drop the notify.mp3 file located at **ios->rider->res->sound.** A popup dialog will appear. Ensure Destination checkbox is checked. Also on Add to targets, ensure the checkbox there is checked.

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.



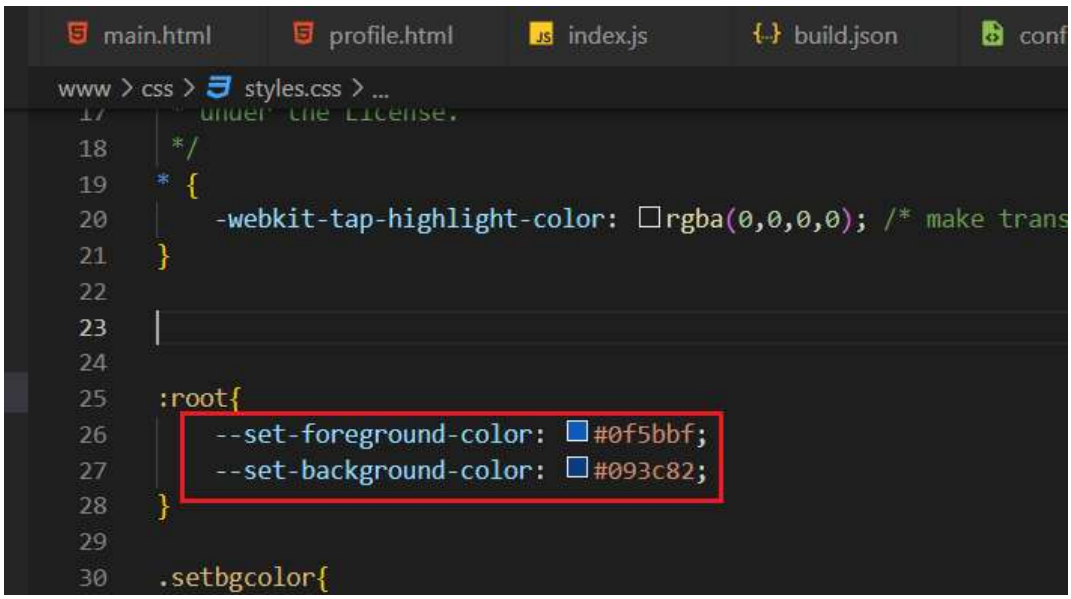
Plugin your iPhone to your Mac and ensure it is detected in Xcode. This is necessary as Xcode will automatically create a provisioning profile in your apple developer account and at least a device UUID is required for this purpose.

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 AppIcons.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 AppIcon.appiconset folder to **ios->rider->platforms->ios->[MyAppName]->Assets.xcassets** folder and overwrite the **AppIcon.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.



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:

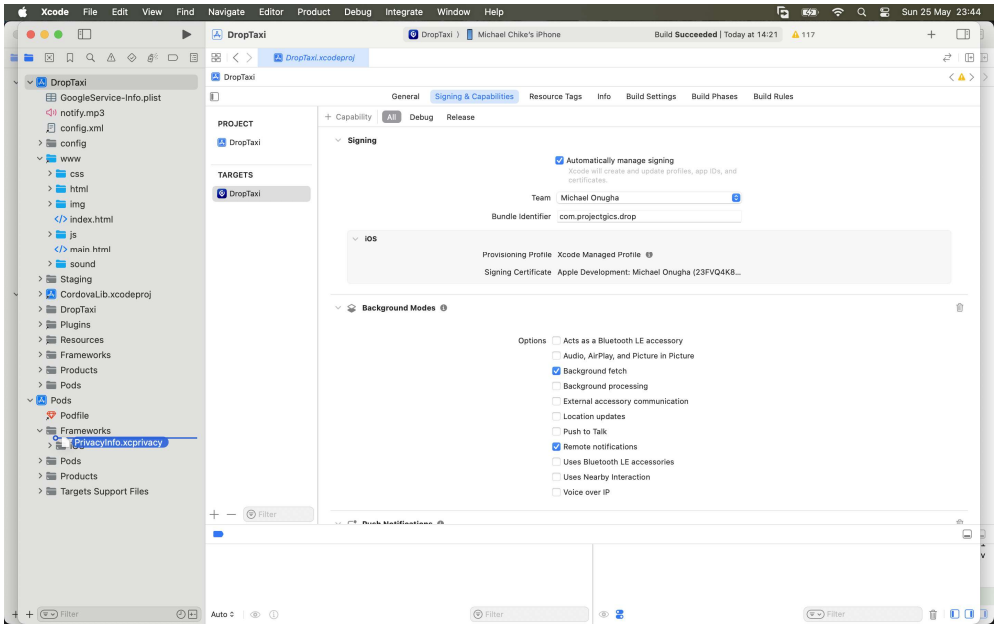
cordova run --device

This will copy all code changes to xcode. Switch back to xcode and click the run button. xcode will build the code and copy it to your connected iPhone. You should see the app start and run on your iPhone. After testing on the iPhone, click the stop button on xcode.

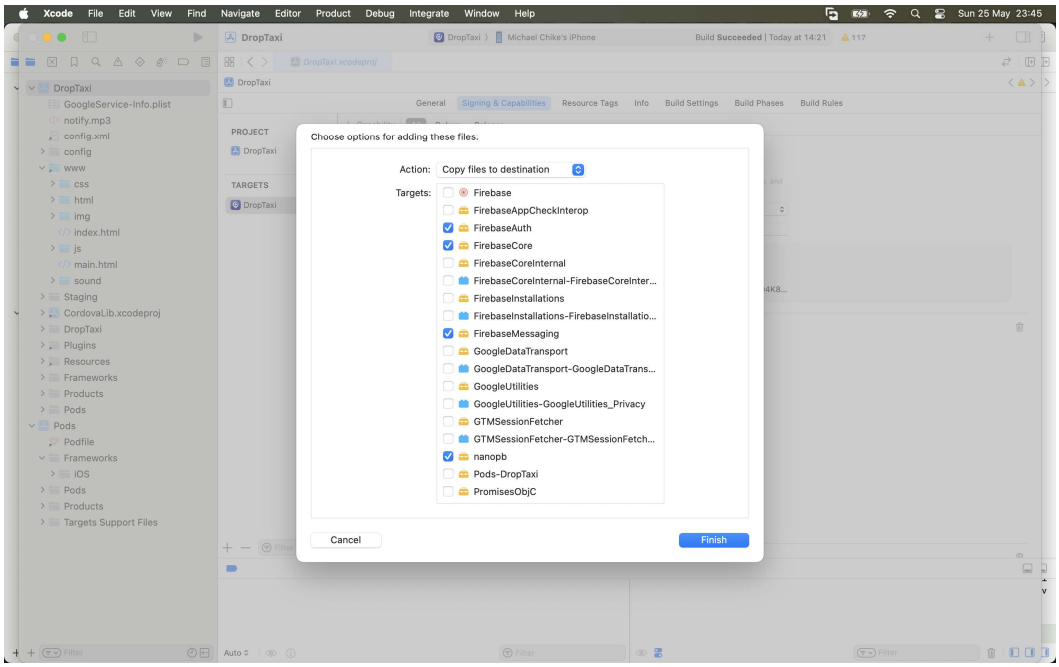
Apple now requires that a privacy manifest be submitted for certain SDKs when used in a project. Droptaxi uses the following SDK libraries: Firebase Authentication, Firebase Core, Firebase Messaging and Nanopb. These require a privacy manifest to be included else you will not be able to submit the app for review on App store. Follow these steps to add a privacy manifest to your project. (Please note that you should

perform this step last before archiving your project. If you perform any build operation on cordova, you must repeat this step again).

Locate the **privacyinfo.xcprivacy** file in the root of the rider folder. Drag and drop this file into the left sidebar panel of Xcode under **Pods->Frameworks** as shown in the image below.



When you drop the file, a dialog will be appear as shown in the image below.



Ensure you select only the options selected in the image above. When done click finish. Next is to archive your project in preparation of uploading to app connect.

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.

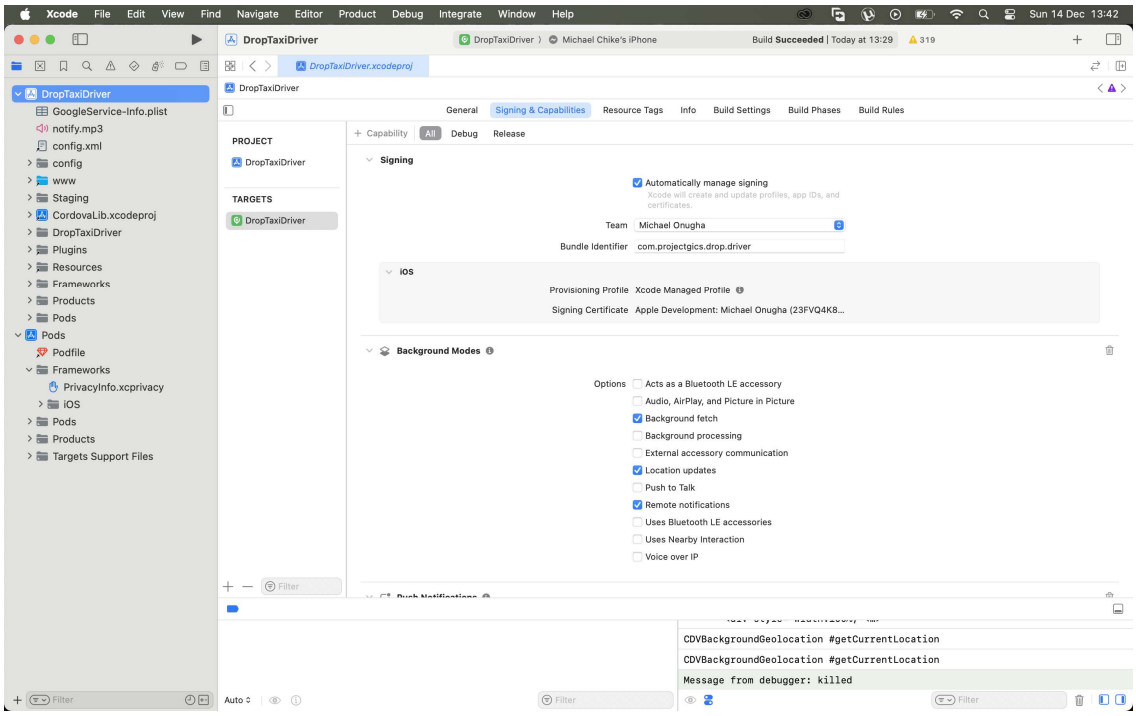
DRIVER APP

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, Background fetch 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 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.

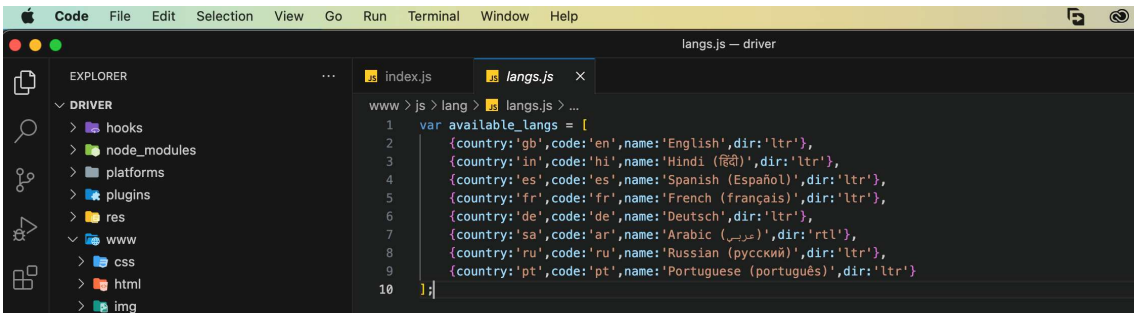
Users can easily switch between the eight languages from the login screen or from the profile page on the rider and driver apps.

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



order they appear in the apps.

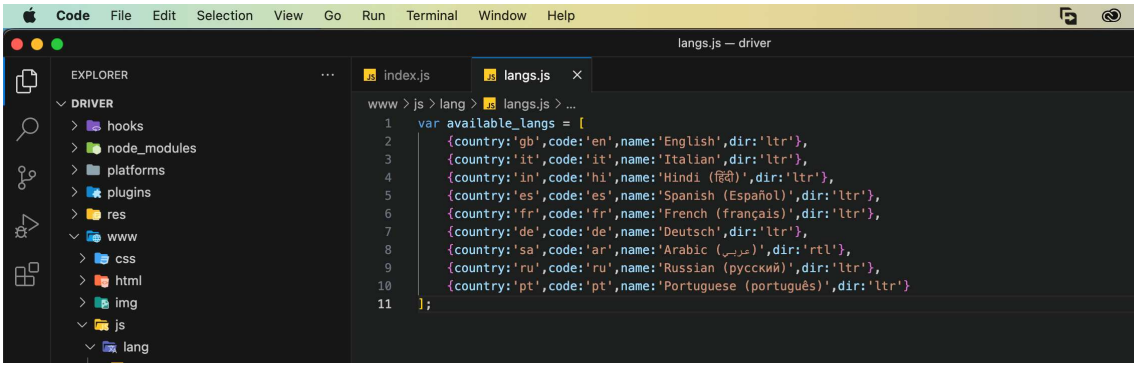
Let's add the Italian language. We will copy the code line (Line 2 as shown in the above image) for English language and paste it just below the same English language code line. Then we will edit the code line from this:

{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.

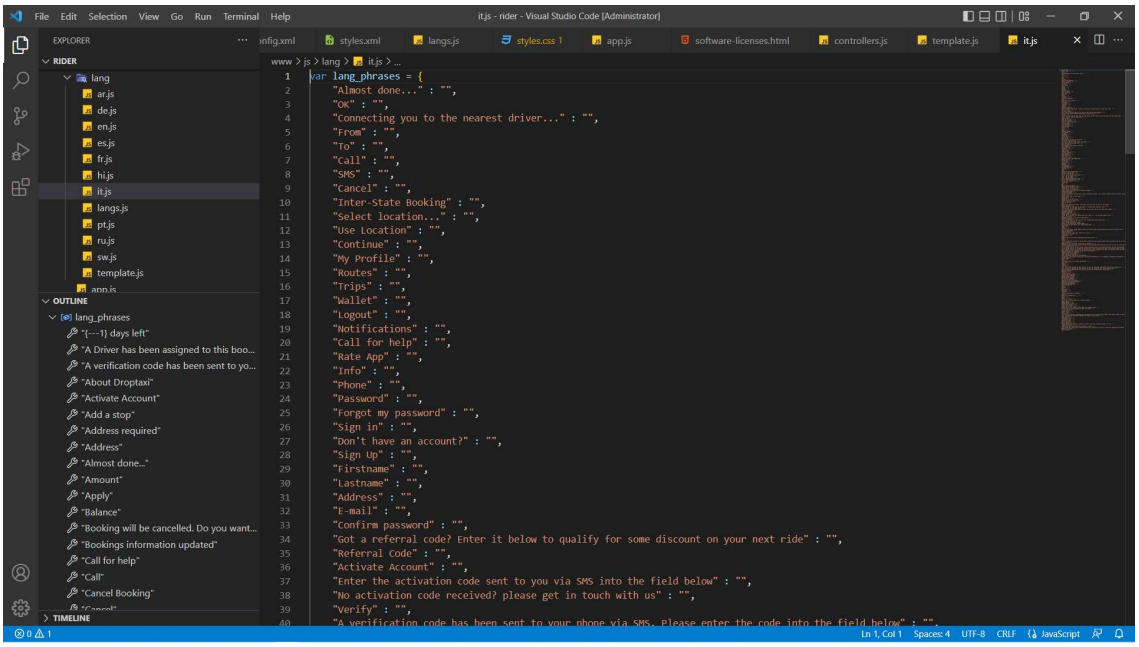
Save the changes you have made to lang.js file.

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

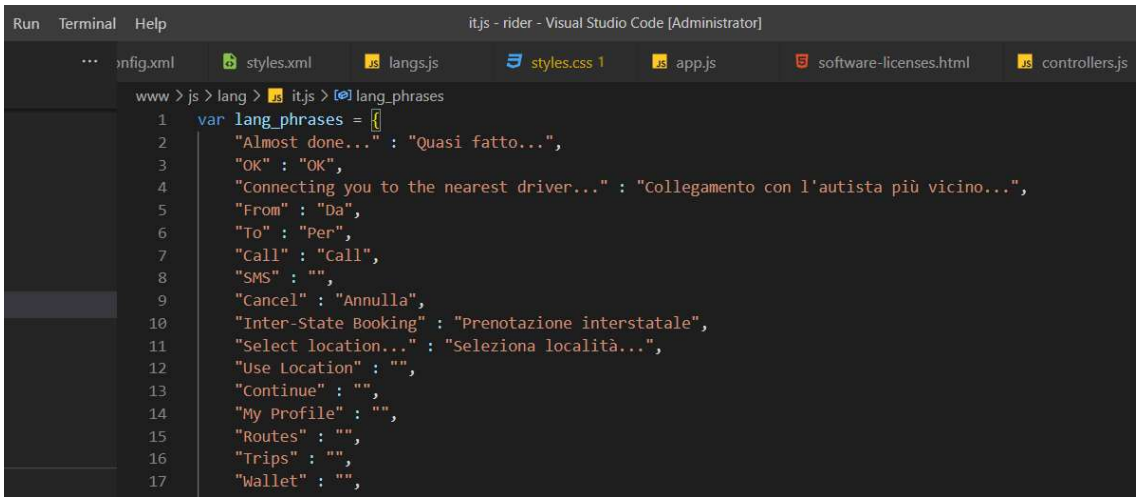
[2-letter language code].js

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:

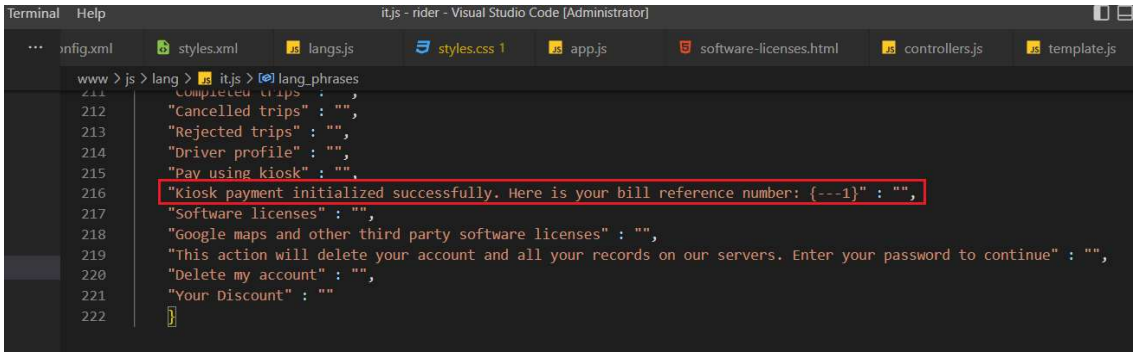


The file contains a JavaScript object variable with all the English words and phrases used in the app arranged as properties of the object. The property values will hold corresponding language translations of the English word and phrases properties. You can also look at it this way; each line has a word 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:



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.



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.

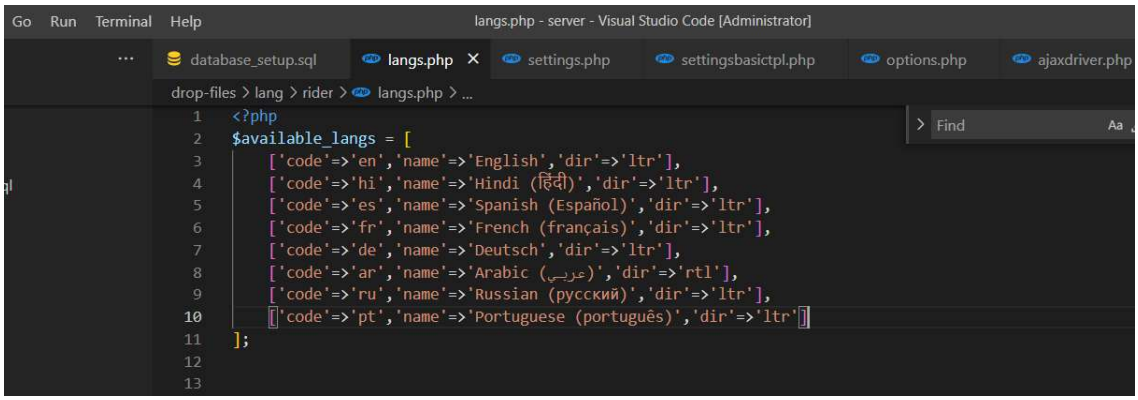
After translating all the words and phrases, save the file. That’s all for the app.

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**

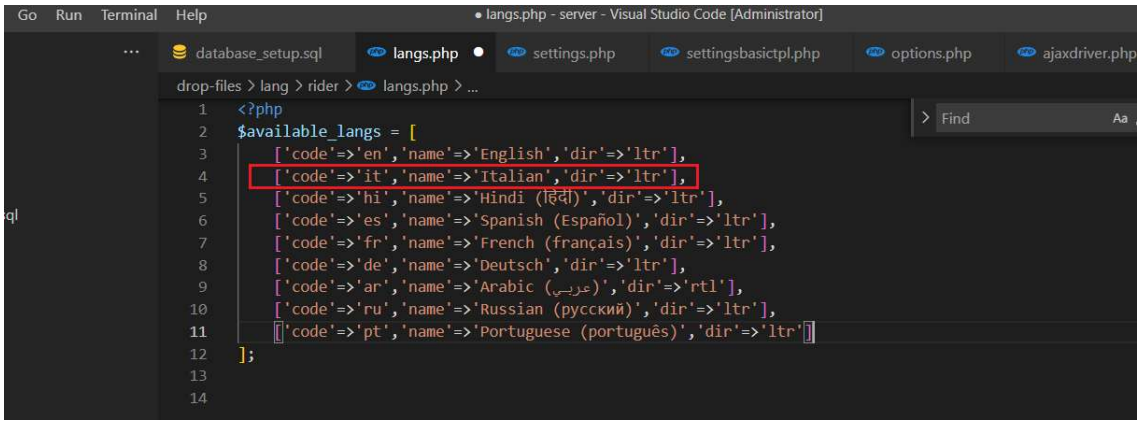
You will find three folders – autodispatch, rider and driver. The rider and driver folders each contains the language descriptor file **langs.php** and the language translation files of various languages. The language translation files in the rider folder provide language translations for strings used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



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 is 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:



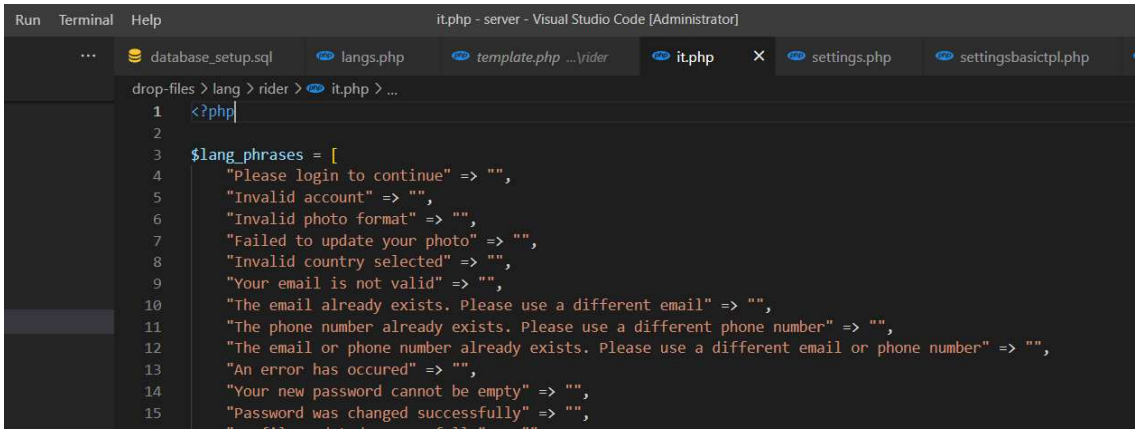
When done, save the file.

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

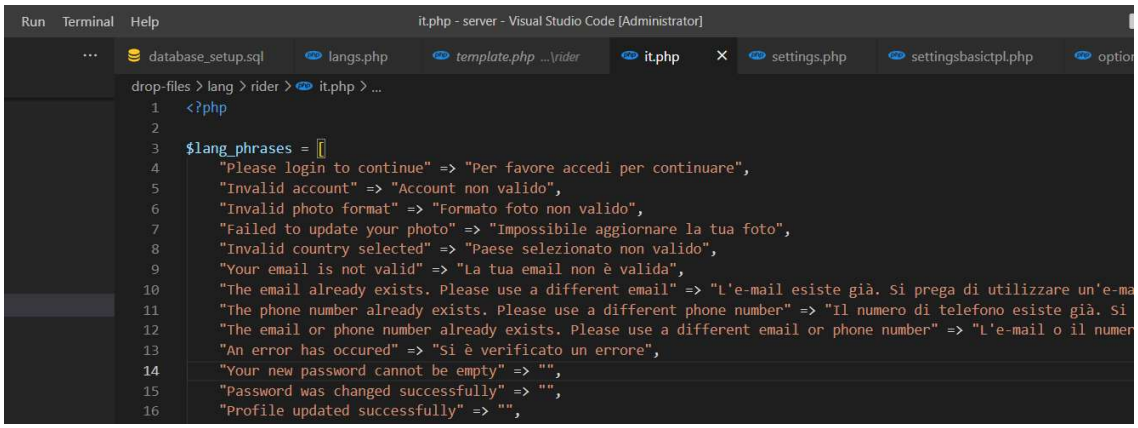
[2-letter language code].php

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.



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.



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

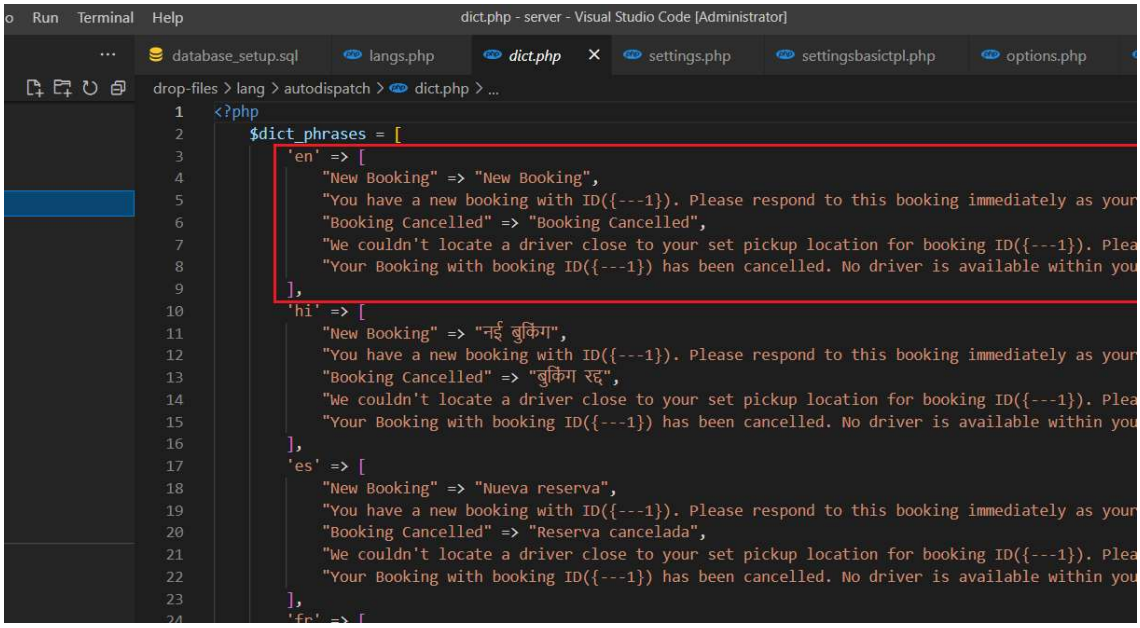
Some key strings will contain one or more variables that look like this {---1} {---2} {---3} or {---4}. Always ensure the translated string contains the same number of variables as the key.

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

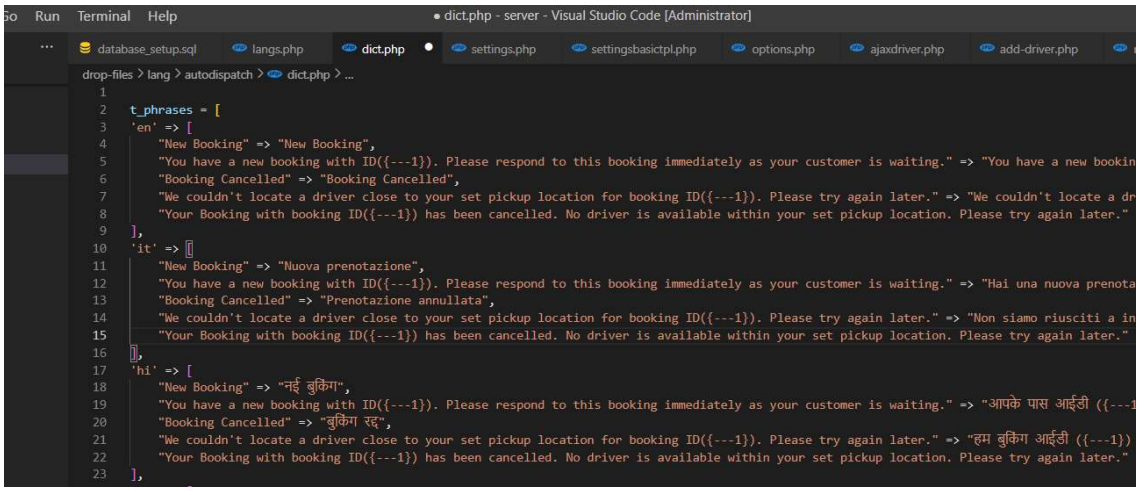
server->drop-files->lang->driver

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:



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:



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 2.3.0!

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

Whatsapp: +2348035631219

Thank you