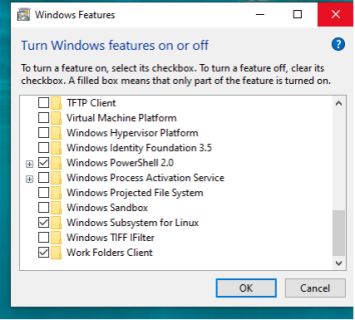
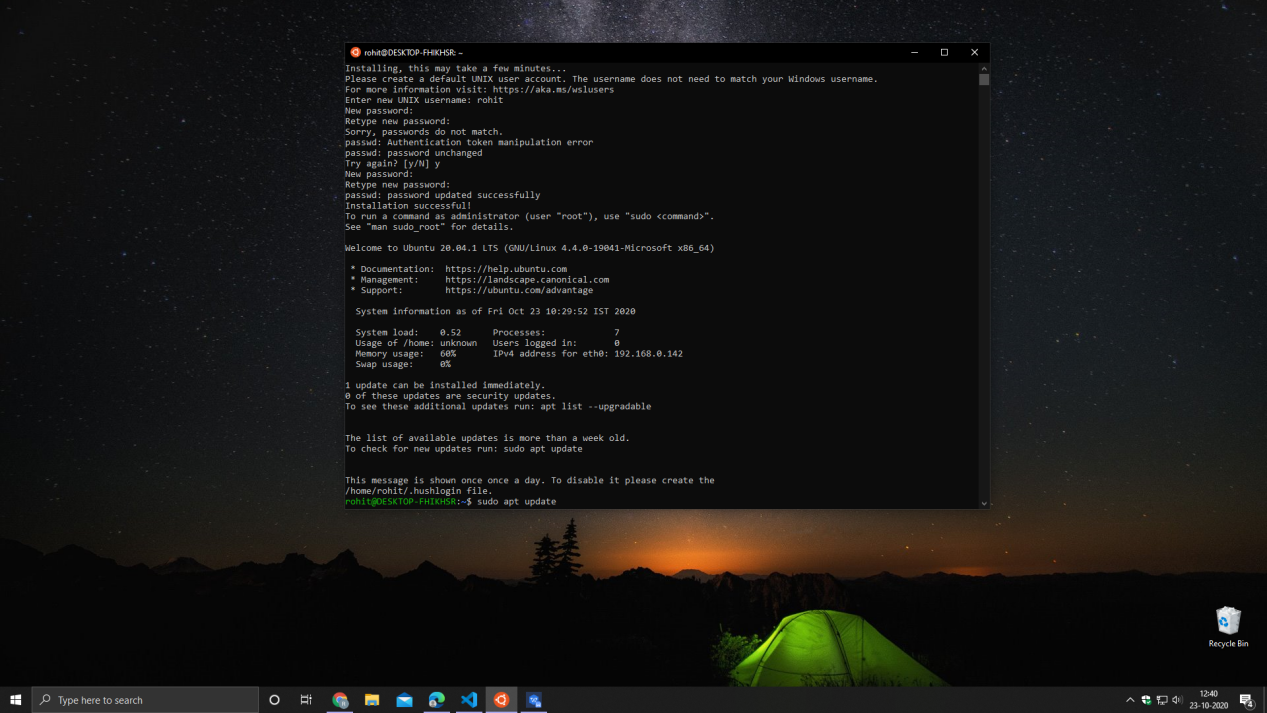
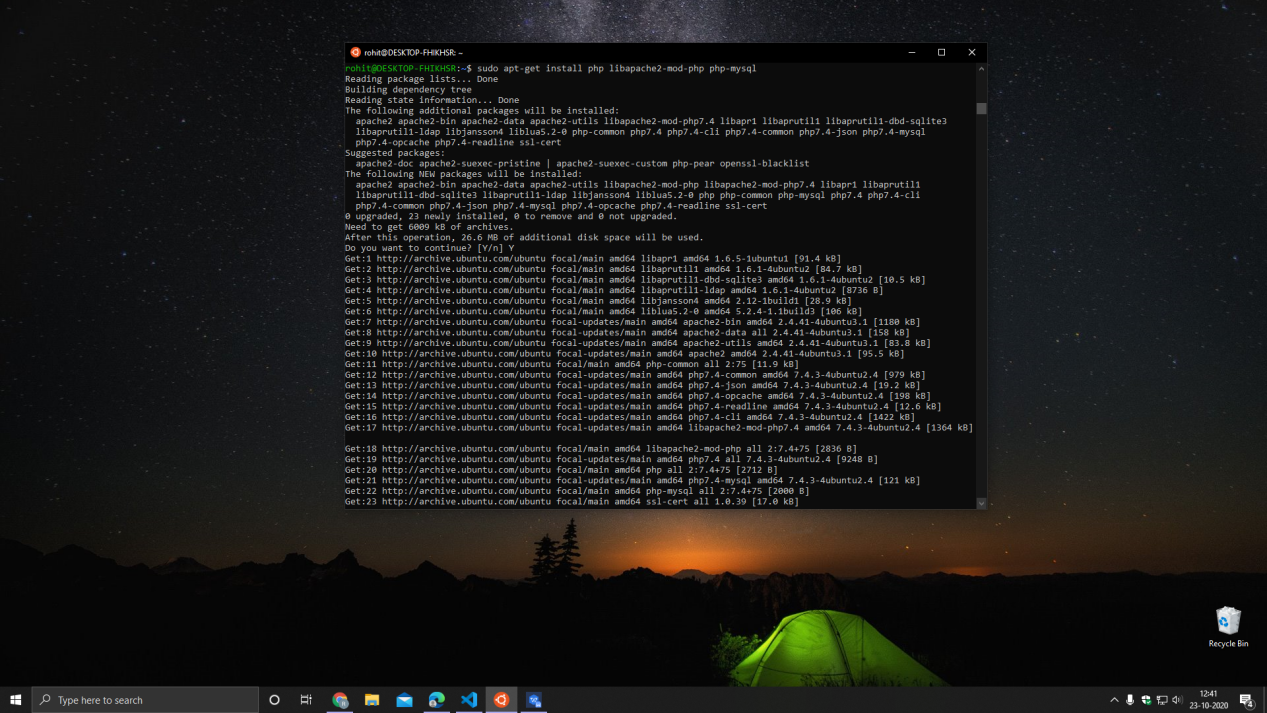
1. Install ubuntu (wsl)from microsoft store / use ubuntu if you don’t have windowsI
   1. I have used windows subsystem of linux for this project as ffmpeg has better support for linux than windows
   2. Also enable windows subsystem for linux from ‘*turn windows features on or off*’ menu





1. Install php using the command (php version 7.4.3)

‘*sudo apt-get install php libapache2-mod-php php-mysql*’



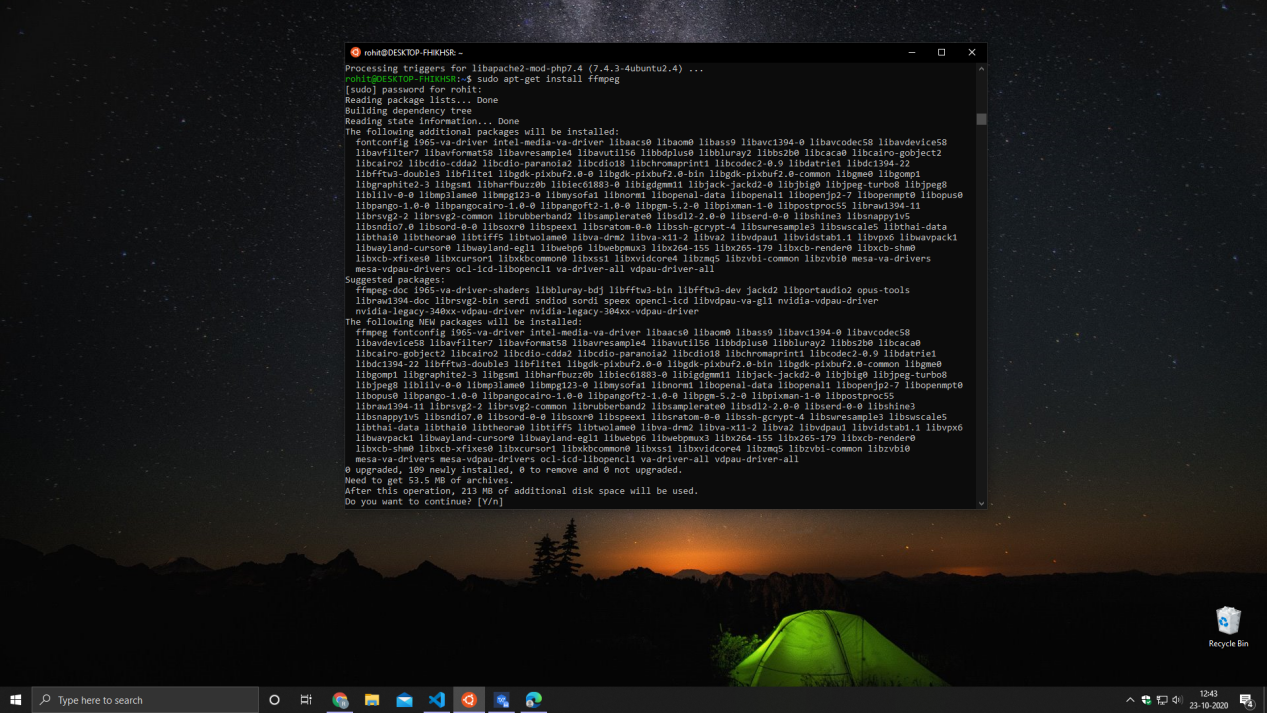
1. Install ffmpeg extension using the command

‘*sudo apt-get install ffmpeg*’

Add ‘*extension=ffmpeg*’ in php.ini file below all the extensions

Change post\_max\_size to 1024M

Using the command ‘*service apache2 restart*’

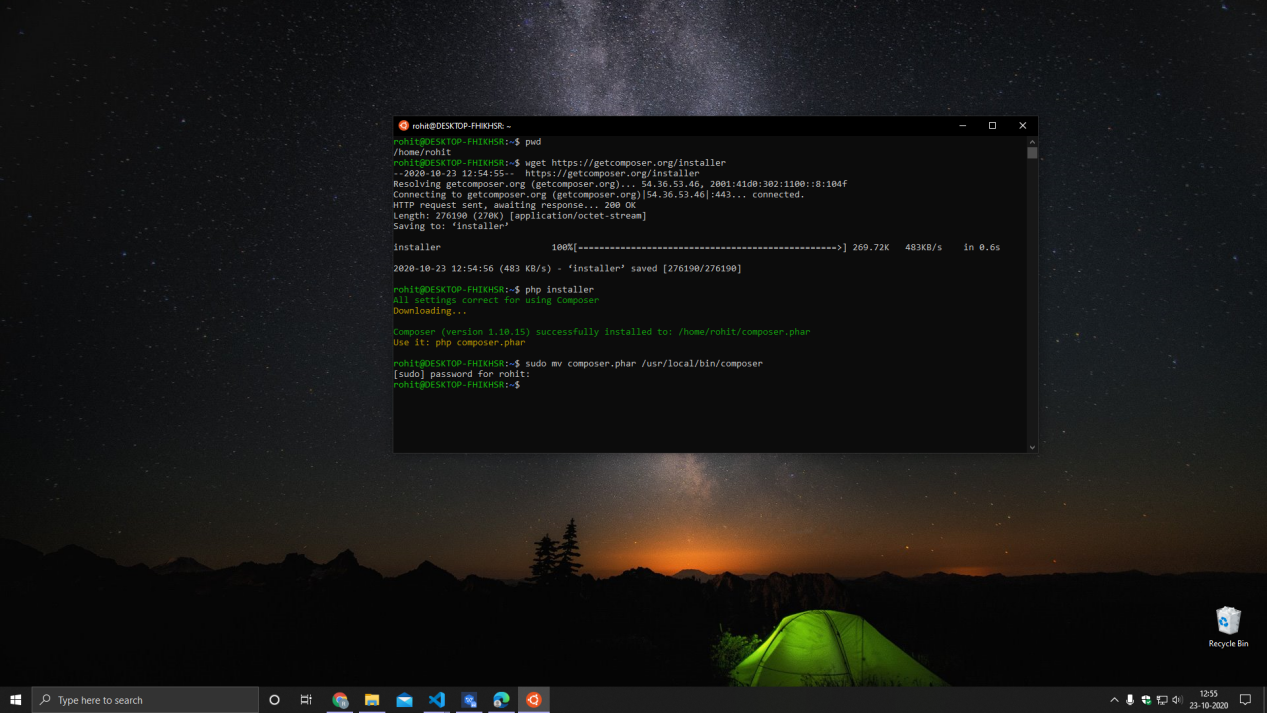


1. Get the repository listing for composer and install it using the commands

*wget <https://getcomposer.org/installer>* ( to get repository listing)

*php installer* (to install composer which is named as ‘installer’)

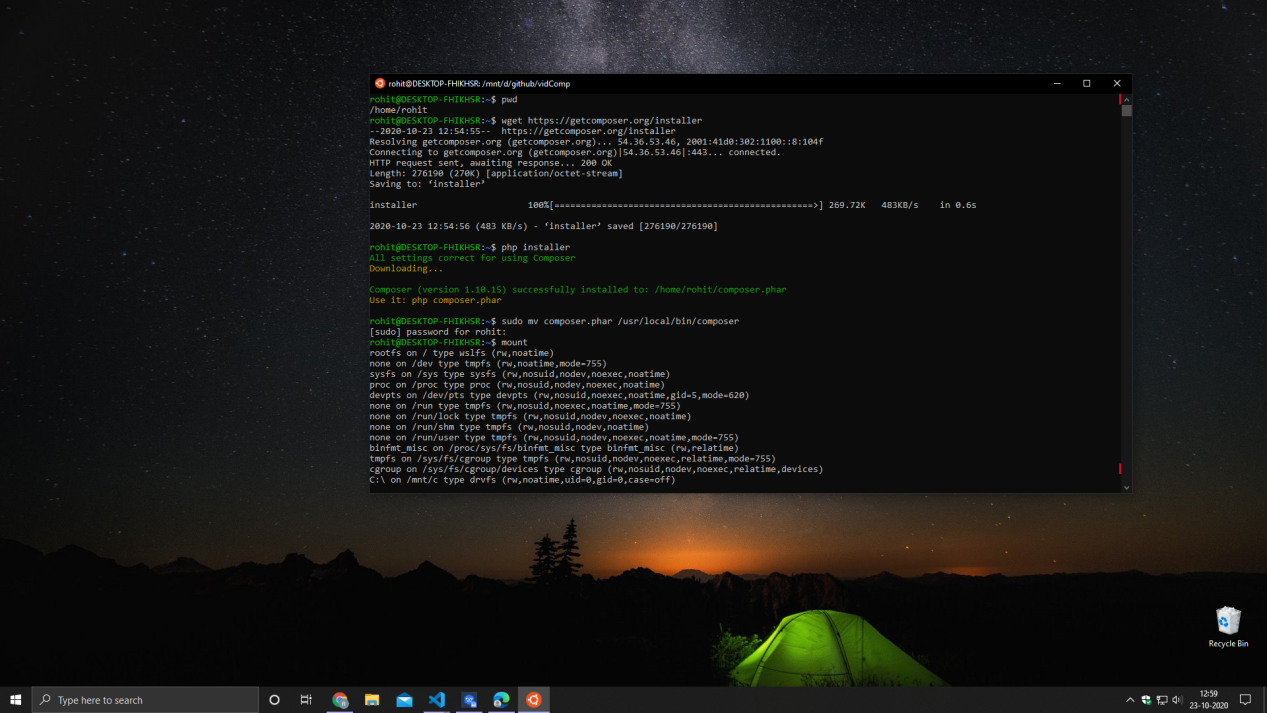
*mv composer.phar /usr/local/bin/composer* (move the Composer file to /usr/local/bin directory)



1. create a laravel project using composer

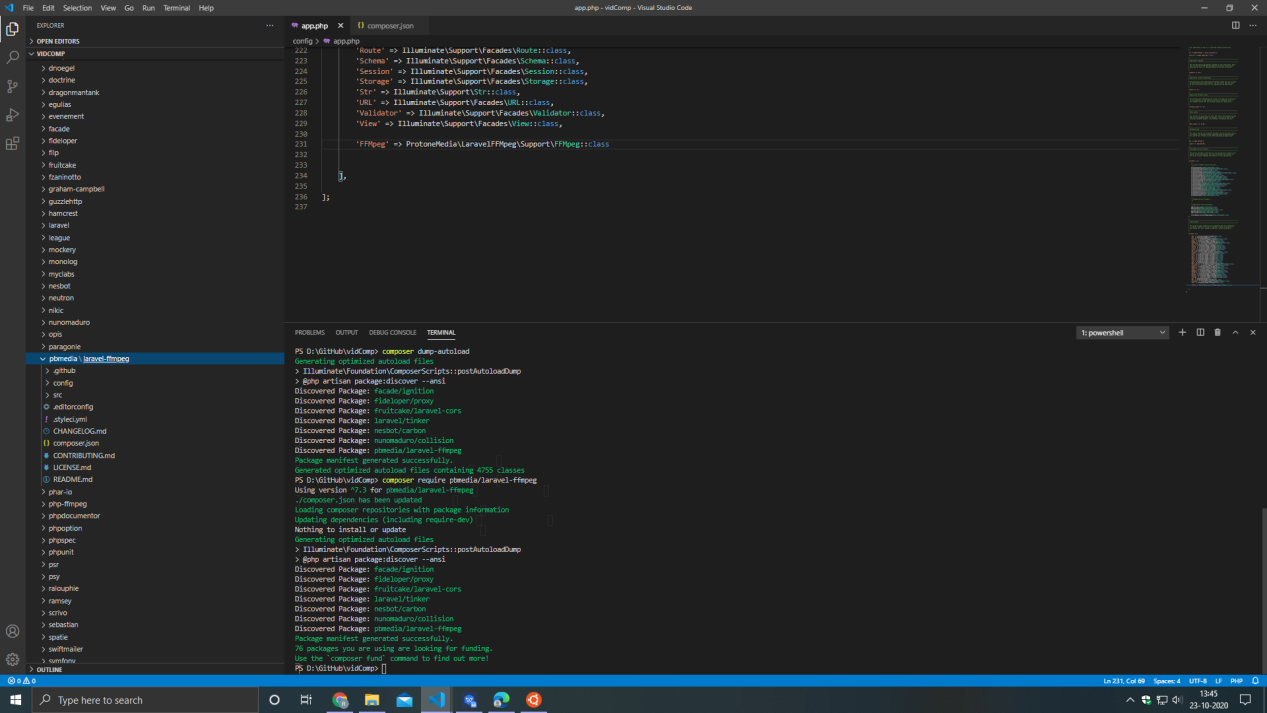
*composer create-project --prefer-dist laravel/laravel vidComp*

1. Change your working directory to the newly created project

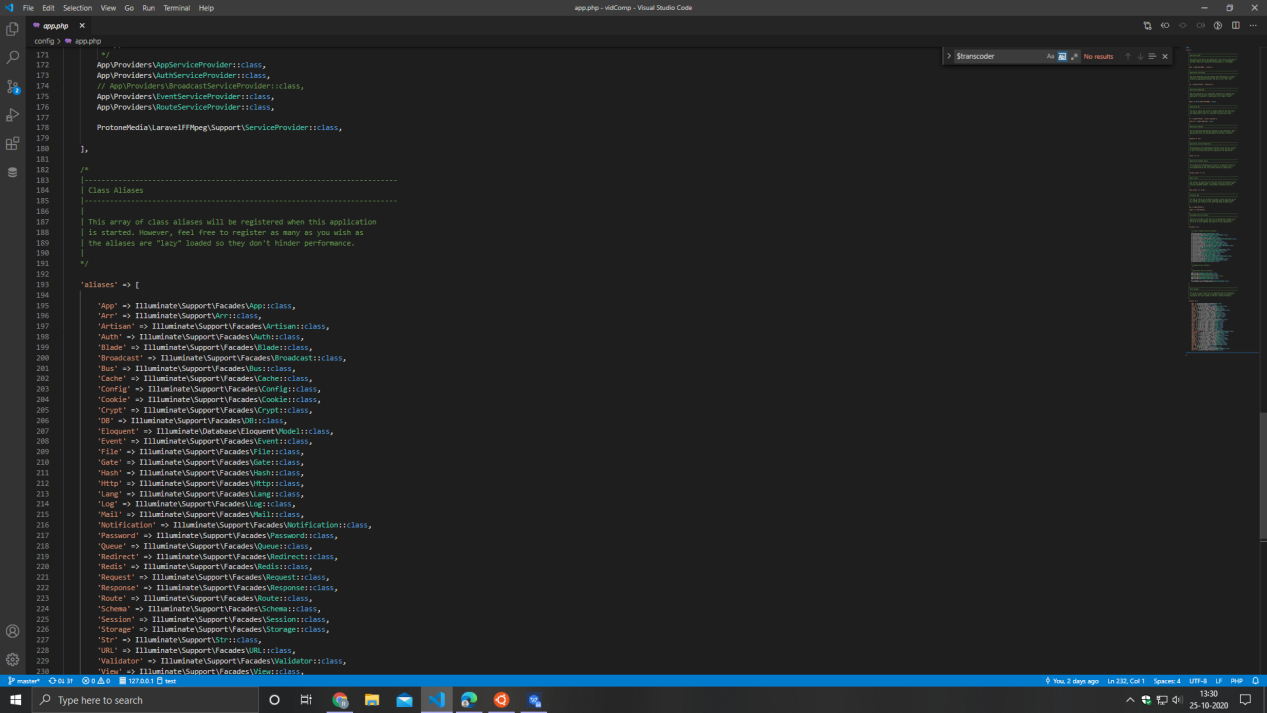


1. Install ffmpeg vendor package by pbmedia (now ProtoneMedia)

*composer require pbmedia/laravel-ffmpeg*



1. Add its provider in *config/app.php* as shown below



1. Install handbrakeCLI
   1. I have used handbrake GUI for windows in the past, so I have tried using handbrakeCLI for video compression
   2. I have referred to this website for installation (*https://launchpad.net/~stebbins/+archive/ubuntu/handbrake-releases*)
      1. Add this package archive using

(*sudo add-apt-repository ppa:stebbins/handbrake-releases*)

* + 1. Run the command (*sudo apt-get update*)
    2. Install handbrakeCLI using (*apt-get install handbrake-cli*).

1. Set up mysql connection by providing credentials in ‘*.env’* file
2. Set up laravel jobs and failed jobs table as well as the table ‘*transcodes*’ which will be used to tracking all transcoding jobs, using the command

*‘php artisan:migrate’*

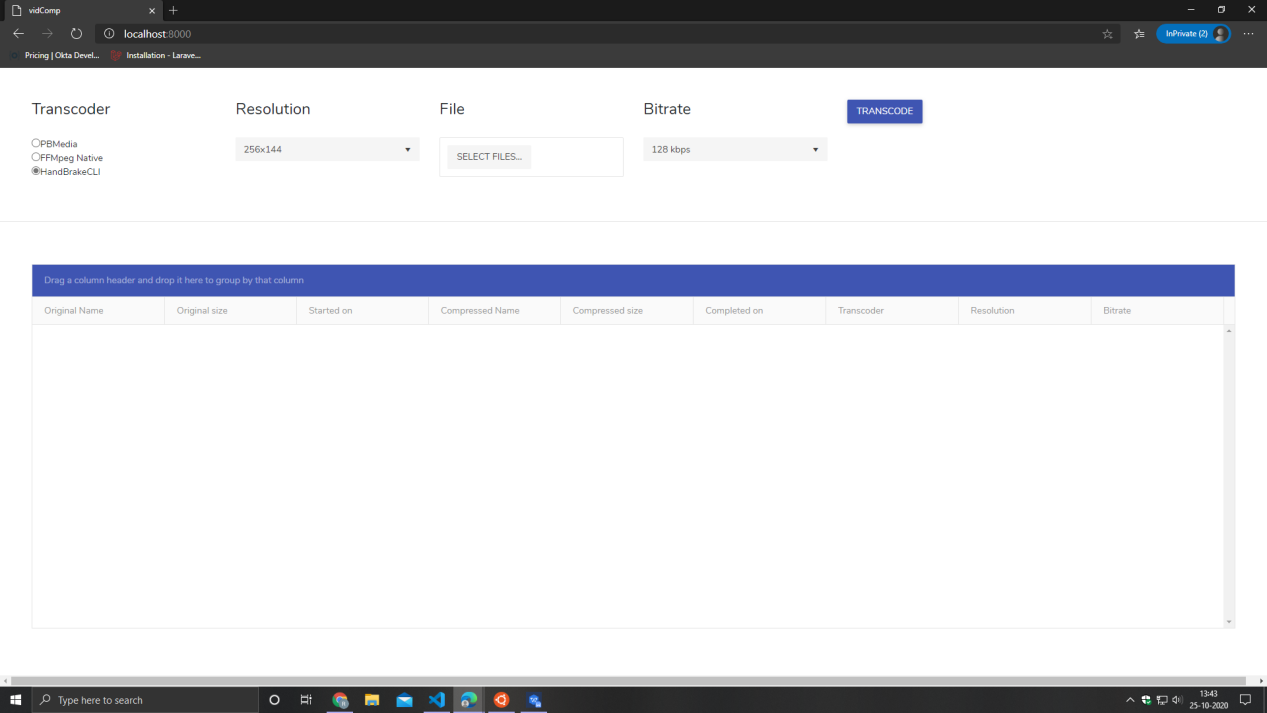
1. Start the application using the command

‘*php artisan serve*’ (*using windows for this*)

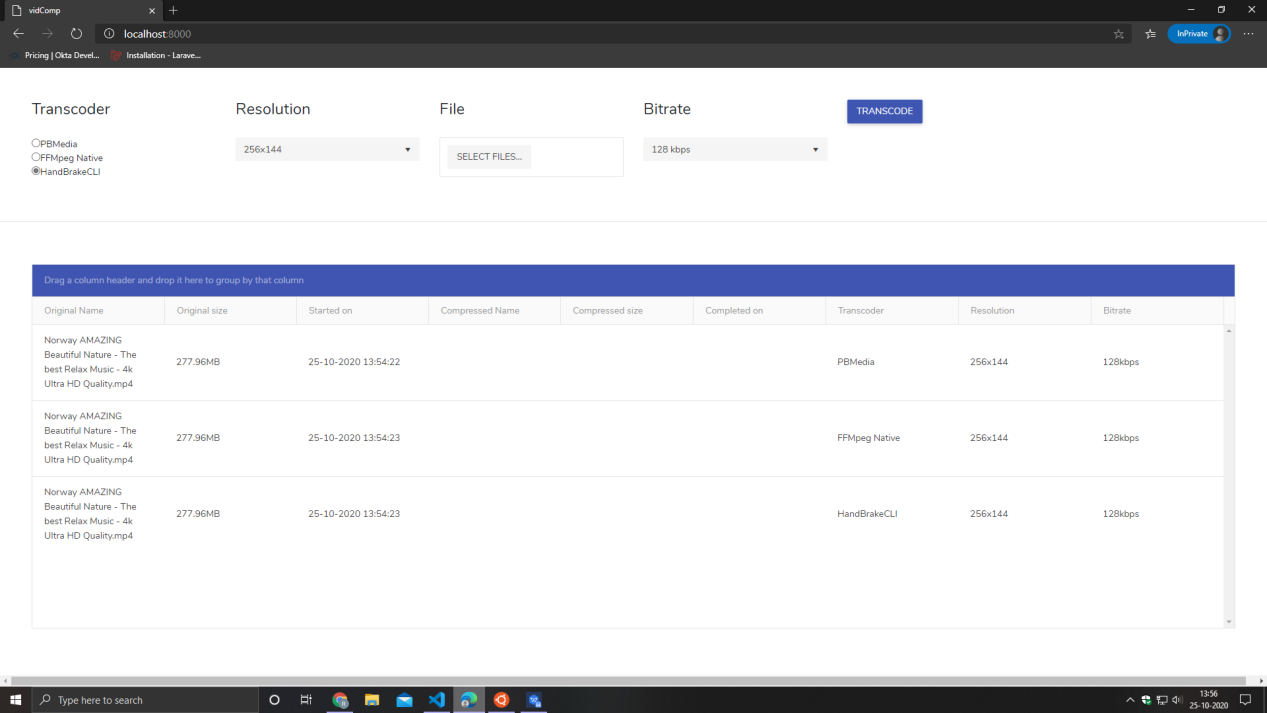
1. Start the job worker using the command

‘*php artisan queue:work’ (using wsl ubuntu for this, since ffmpeg is installed on ubuntu)*

1. Upload the file after choosing the plugin to use for compressing, the resolution of the final output video and its bitrate



1. The table below the form shows the list of jobs and the details for the job



I have used 3 variations for compression

**FFmpeg (native)**

Fmpeg is a free and open-source software project consisting of a large suite of libraries and programs for handling video, audio, and other multimedia files and stream.

It allows for format, dimension and bitrate compression / changing

For the purpose of this project I have used x264 encoding which provides good quality output with low size and low noise for the output

**ProtoneMedia/ PBmedia**

It is built on top of FFmpeg library, sort of like a wrapper

It allows for the same compression format and bitrates

The syntax is much more cleaner than the native FFmpeg library.

It has better progress tracking than the native FFmpeg library

**HandBrakeCLI**

It is CLI version of Handbrake which is used for video format conversion, ranging from ‘*mp4*’ to ‘*m4v’* to *‘matroska’* and many more. It has a large list of features which others don’t like chapters, subtitle embedding, multiple audio track embedding.

It also has support for x265 encoding which is even smaller in size than x264. x265 has a better quality to size ration than x264.

HandbrakeCLI is the most versatile of all, but it is not a plugin, its just a CLI which can be used directly from any application as well as from the CLI

* In my testing I have found, handbrake to perform the best in terms of speed, with almost the same size as FFMpeg and PBmedia (2 mins)
* PBmedia and native FFMpeg, perform almost the same since they use the same underlying library, with more or less, the same output file size for same resolution and bitrate (5 mins)

