**LOSSLESS VIDEO COMPRESSION**

**SYNOPSIS**

Today there is a growing need for Video Compression in order to reduce file size on storage requirement. Higher compression ratio can be achieved using lossy compression technique, but this will lead to loss of information and may result in errors. Hence there is a need to store video in lossless format. Traditional lossless compression technique results in low compression ratio, the goal is to maximize the compression using a lossless compression tool. HEVC encoding can effectively exploit the temporal and spatial redundancy observed in video sequence. This project outlines lossless encoding mode of HEVC. It is prepared as the newest video coding standard of the ITU-T Video Coding Experts Group and the ISO/IEC Moving Picture Experts Group. Lossless compression is useful when it is necessary to minimize the storage space or transmission bandwidth of data while still maintaining archival quality. Many applications such as medical imaging, preservation of artwork, image archiving, remote sensing, and image analysis require the use of lossless compression, since these applications cannot allow any distortion in the reconstructed images. As the HEVC standard requires the video sequence in non-camera capture format i.e. in YUV format, thus to encode the image sequence for camera captured format, an internal conversion from RGB to YUV format is required.

**EXISTING SYSTEM**

Huge software like Adobe Media Encoder is required to convert a video file format but there may me slight quality reduction. Few existing web video format converters are not efficient enough to convert without quality loss under feasible timing.

**PROPOSED SYSTEM**

Lossless video compression is done just by Web UI platform. A video file can be easily compressed without quality loss under less timing (according to the Computer/Server’s Speed).There are multiple mkv player for all the Operating platform which makes not only easy to store efficiently also to access and preview it faster.

**MODULE DESCRIPTION**

**Flask**

* Flask is a micro web framework written in Python.
* It is classified as a microframework because it does not require particular tools or libraries.
* It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself.
* Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.

Components

**Werkzeug** is a utility library for the Python programming language for Web Server Gateway Interface (WSGI) applications. Werkzeug can instantiate objects for request, response, and utility functions. It can be used as the basis for a custom software framework and supports Python 2.7 and 3.5 and later.

**Jinja** is a template engine for the Python programming language. Similar to the Django web framework, it handles templates in a sandbox.

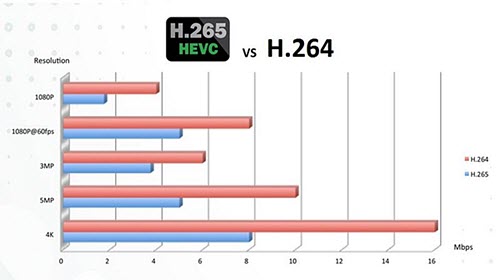
**MarkupSafe** is a string handling library for the Python programming language. The eponymous MarkupSafe type extends the Python string type and marks its contents as "safe"; combining MarkupSafe with regular strings automatically escapes the unmarked strings, while avoiding double escaping of already marked strings.

**ItsDangerous** is a safe data serialization library for the Python programming language. It is used to store the session of a Flask application in a cookie without allowing users to tamper with the session contents.

**HEVC (.mkv) File Format**

An HEVC file contains a video stored in the High Efficiency Video Coding (HEVC) format. This format, also known as H.265, improves over the H.264 standard by allowing videos to be stored with a lower file size but with the same video quality. HEVC helps users store more videos on their devices and also substantially reduces the file size of high-resolution videos such as 4K and 8K videos.

It has higher compression compared to other formats which means that a 1-minute 4K video file recorded in the HEVC format will consume relatively less storage space compared to the same video file recorded in the regular H.264 format.

FIG 1 : HEVC vs AVC

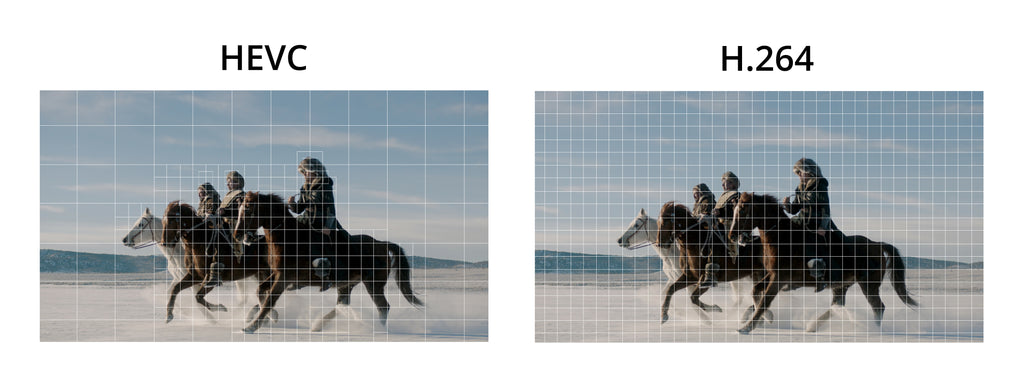
 HEVC is the next generation compression standard that offers a number of enhancements over AVC. HEVC compression is 50% more efficient than AVC, which translates into maintaining the same video quality at half the bitrate or double the video quality at the same bitrate. The figure above represents the resolution: bitrate ratio of different output qualities. In all categories, bandwidth costs are significantly reduced while resolutions remain identical.

FIG 2 : Pixel mapping Advantage in HEVC

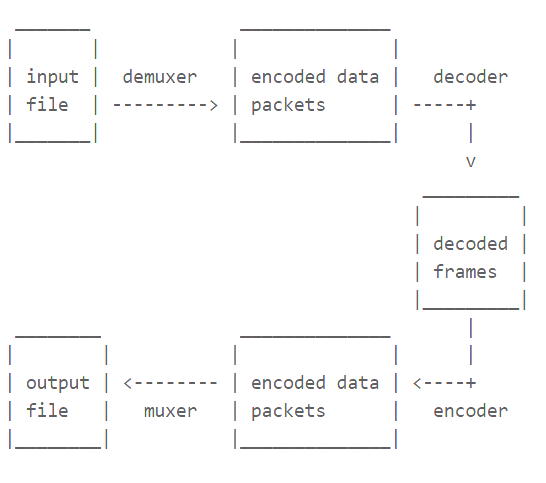
**FFmpeg**

FFmpeg is a collection of libraries and tools to process multimedia content such as audio, video, subtitles and related metadata.

Convert a video file from one format to HEVC format (mp4 to mkv).

ffmpeg -i yourvideoname.mp4 -c:v libx264 outputfilename.mkv

The command above is an example of when converting a .mp4 file into a .mkv file.

FIG 3 Level 0 – Data Flow of FFmpeg

**HARDWARE SPECIFICATION**

|  |  |
| --- | --- |
| **COMPONENTS** | **REQUIREMENTS** |
| CPU | x86 64-bit CPU (Intel / AMD architecture) |
| RAM | 2GB Minimum |
| STORAGE | 5GB Minimum disk space |
| CONNECTIVITY | Internet (WiFi/Ethernet) |
| PERIPHERALS | Common Peripherals (Mouse, Keyboard,..) |

**SOFTWARE SPECIFICATION**

|  |  |
| --- | --- |
| **COMPONENTS** | **REQUIREMENTS** |
| OPERATING SYSTEM | Windows 8 and Above |
| FRONT END | Python Flask |
| PLATFORM | VS Code & Web Environment |
| BACK END | Python FFmpeg |

**PROGRAM**

from flask import Flask, request, jsonify,send\_file

import os

app = Flask(\_\_name\_\_)

CONVERTED\_FOLDER = 'C:/converted/'

app.config['CONVERTED\_FOLDER'] = CONVERTED\_FOLDER

@app.route('/')

def index():

return ''' <style>

h1{

color: #512E5F;

}

.sub{

position: absolute;

right: 31%;

color: #512E5F;

}

.column {

float: left;

width: 25%;

padding: 5% 37%;

}

.row:after {

content: "";

display: table;

clear: both;

}

.card {

box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2);

padding: 16px;

text-align: center;

background-color: #f1f1f1;

}

.back{

background-color: #AED6F1;

}

</style>

<body class ="back">

<br>

<center><h1>LOSSLESS VIDEO COMPRESSION</h1></center>

<center><h3 class ="sub">By Nivyadharsini R</h3></center>

<br><br>

<section class ="move">

<div class ="row">

<div class="column">

<div class="card">

<form id="upload-form" method="POST" enctype="multipart/form-data" action="/convert">

<input type="file" name="mp4\_file"><br><br>

<input type="submit" value="Convert">

</form>

</div>

</div>

</div>

</section>

</body>'''

@app.route('/convert', methods=['POST'])

def convert():

mp4\_file = request.files['mp4\_file']

mp4\_path = os.path.join( mp4\_file.filename)

mkv\_file = os.path.splitext(mp4\_file.filename)[0] + '.mkv'

mkv\_path = os.path.join(app.config['CONVERTED\_FOLDER'], mkv\_file)

mp4\_file.save(mp4\_path)

os.system(f'ffmpeg -i {mp4\_path} -c:v libx265 -preset medium -crf 28 -c:a aac -strict experimental -b:a 128k -movflags +faststart -y {mkv\_path}')

return jsonify({'The Video was downloaded in converted folder as': mkv\_file})

@app.route('/download/<filename>')

def download(filename):

return send\_file(os.path.join(app.config['CONVERTED\_FOLDER'], filename), as\_attachment=True)

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

**ER Diagram**

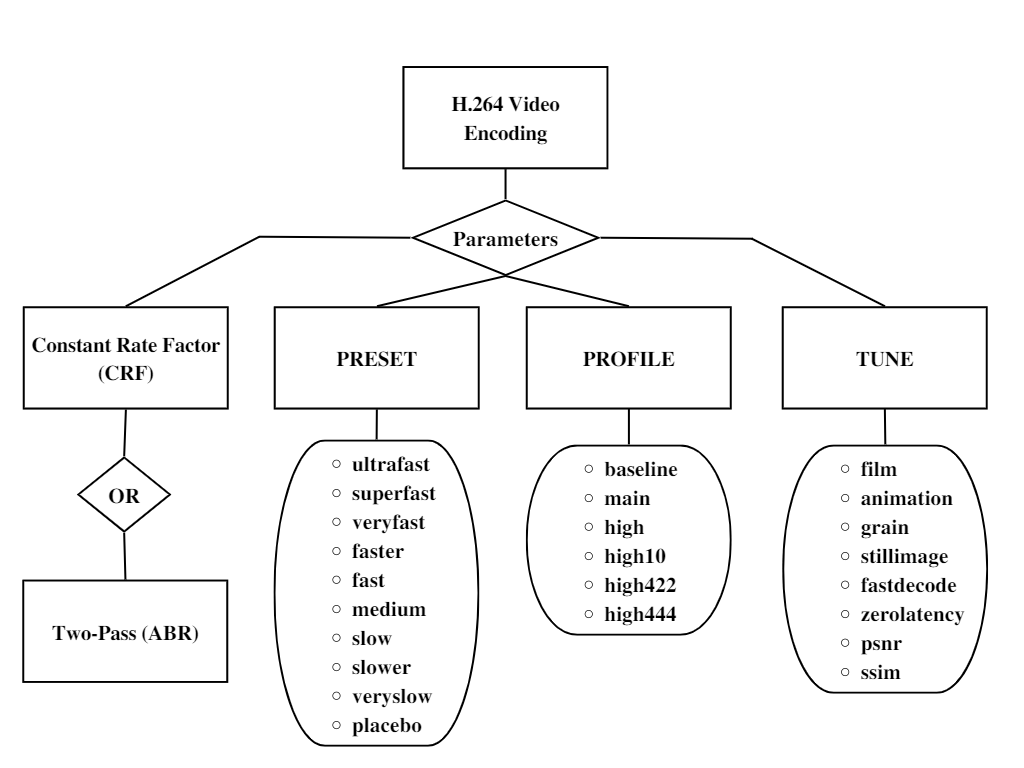
****

FIG 4 – Entity relationship Diagram

**DF Diagram**

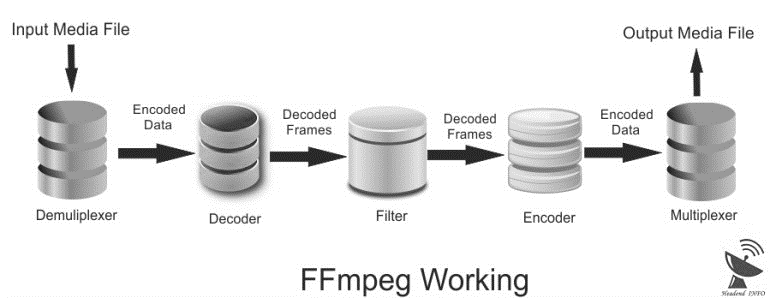


FIG 5 – Level 1 Data Flow Diagram of FFmpeg

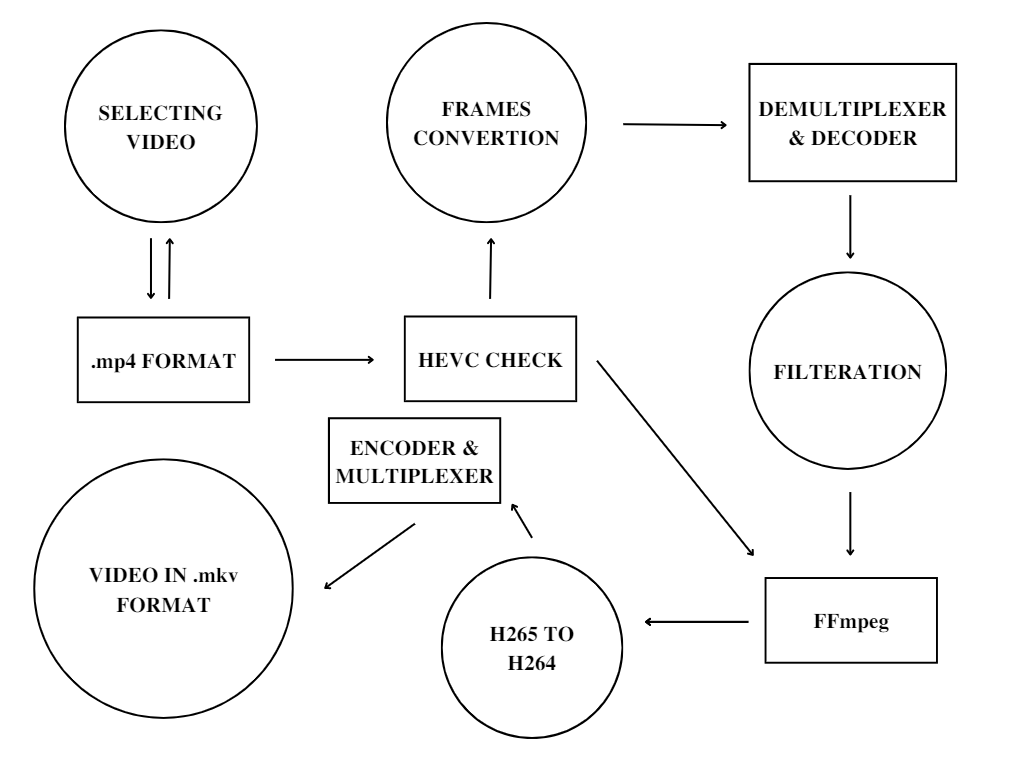


FIG 6 – Level 2 Data Flow Diagram.

**OUTPUT**

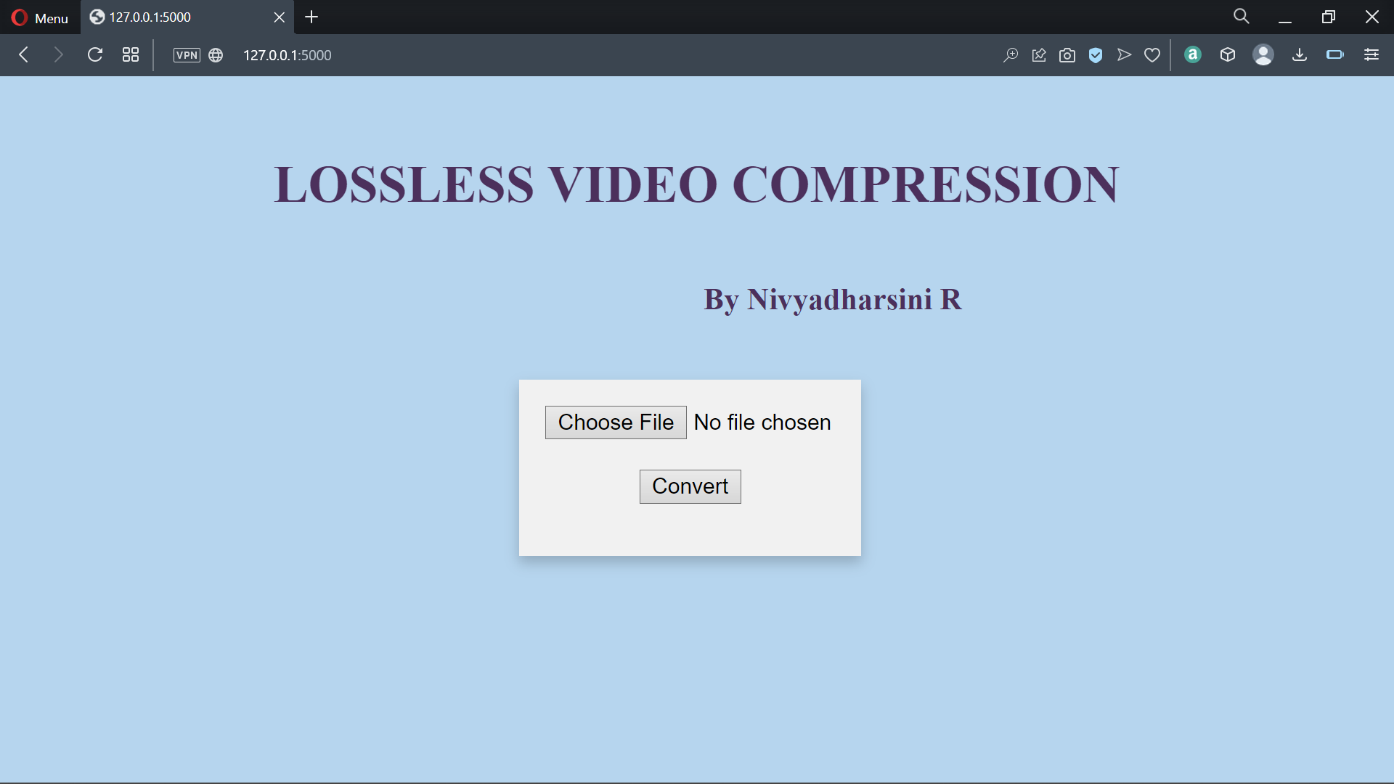
**UI**

FIG 7 – UI

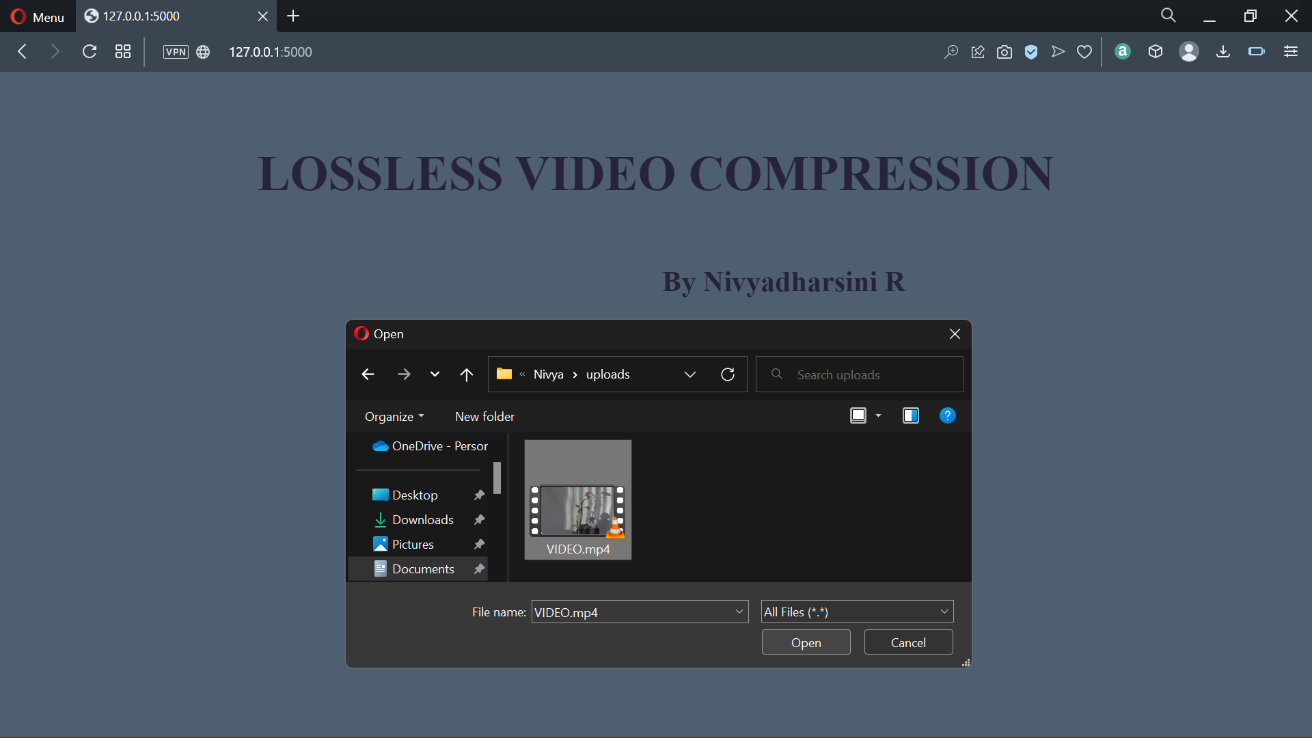
 The static page UI is the Front-End to Upload the video file and convert it at 127.0.0.1:5000 Static IP.

FIG 8 – Choosing the File

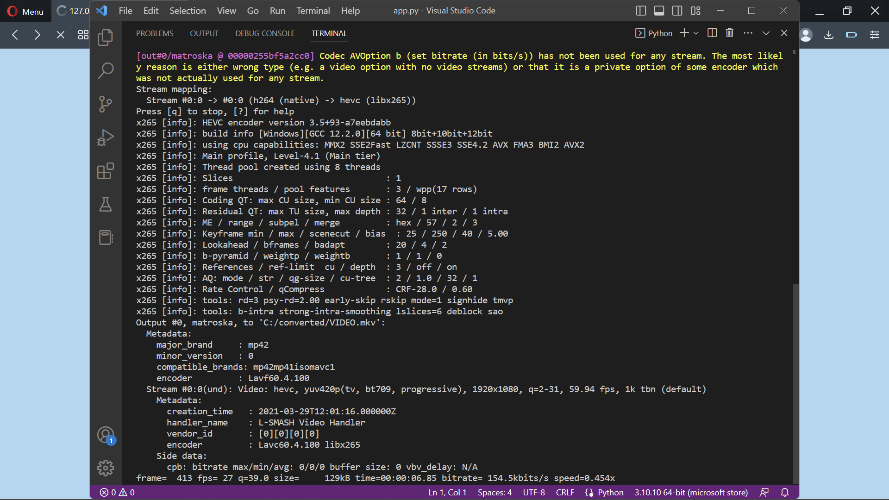


FIG 9 – Conversion Process (Background)

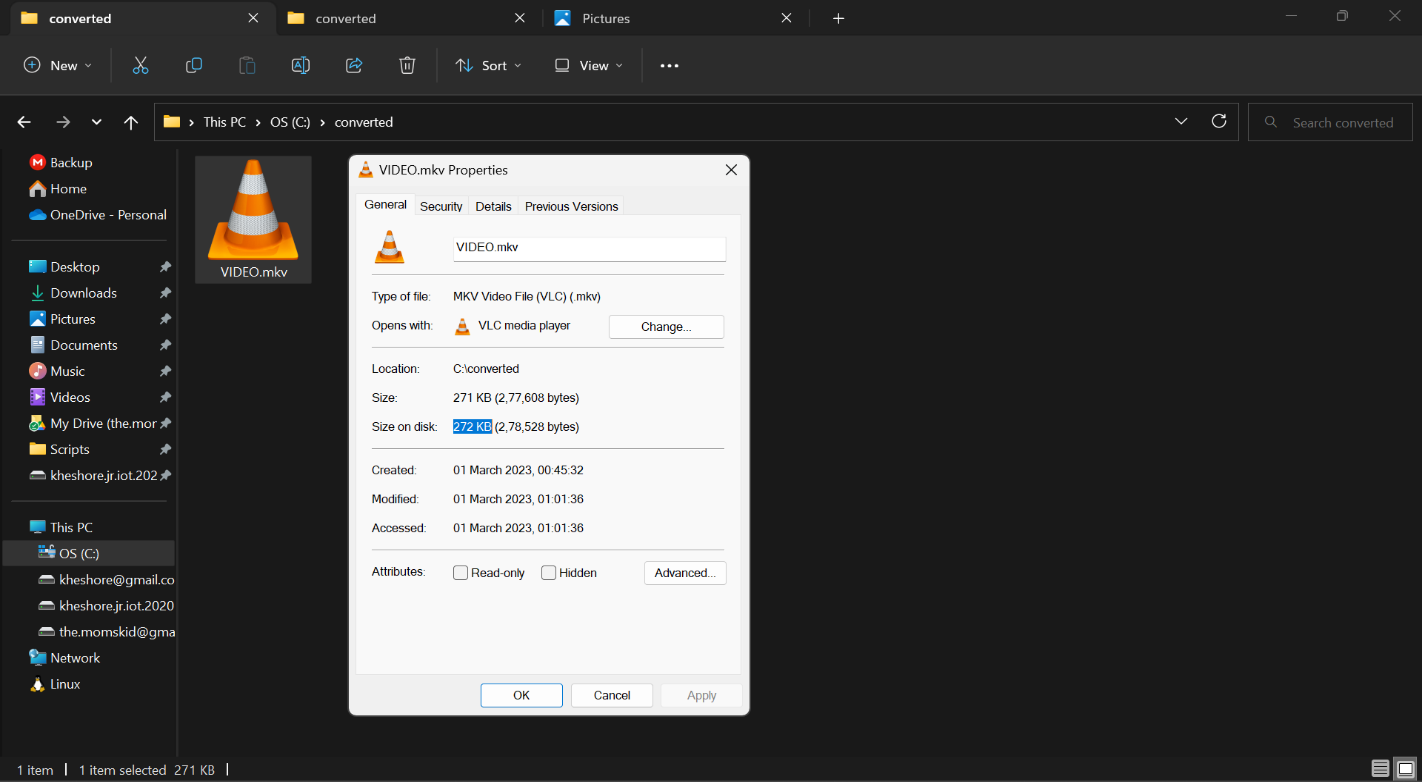
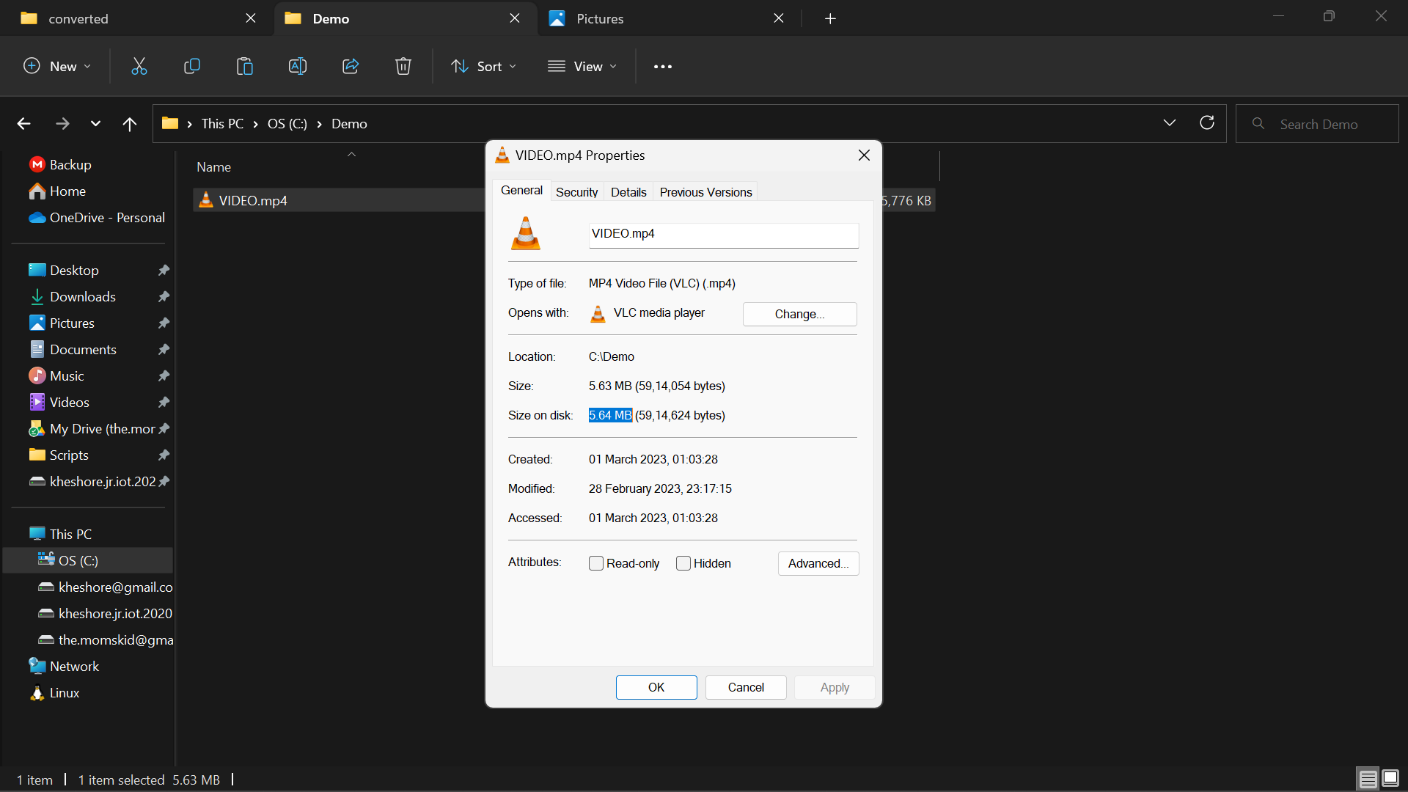
 The uploaded Video file is Converted using convert button in UI, For which in background the encoding and decoding of mp4 file format to mkv file format.

FIG 10 – File Before Conversion (mp4) FIG 11 – File After Conversion (mkv)

~5.64MB ~272 kb

The Video is converted into mkv from mp4 file format successfully as shown in the above the Figure [7] [8].

No quality is lost, as the bitrate is only different.

More than 95% of the size is saved in the above demo.