Project Report

On

All in One Video Downloader Using MAD

Submitted by

N Reddy Thejaswini (R170139)

U Ramudu (R170199)

P Sandhya (R170428)

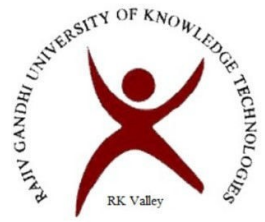
Under the guidance of

P Siva Lakshmi

Department of Computer Science and Engineering

Rajiv Gandhi University of Knowledge and Technologies (RGUKT),

R.K.Valley, Kadapa, Andra Pradesh.



Rajiv Gandhi University of Knowledge Technologies

RK Valley, Kadapa (Dist), Andhra Pradesh, 516330

CERTIFICATE

This is to certify that the project work titled "**All In One Downloader**" is a bonafied project work submitted by N Reddy Thejaswini, U Ramudu, P Sandhya in the department of COMPUTER SCIENCE AND ENGINEERING in partial fulfillment of requirements for the award of degree of Bachelor of Technology in Computer science and engineering for the year 2021-2022 carried out the work under the supervision.

P Siva Lakshmi
Assistant Professor
Department of CSE
Project Internal Guide
RGUKT R.K. Valley

P Harinadha
Head of the Department
Computer Science and Engineering
RGUKT R.K. Valley

Submitted for the Practical Examination held on…….

Internal Examiner **External Examiner**

ACKNOWLEDGMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose constant guidance and encouragement crown all the efforts success.

I am extremely grateful to our respected Director,Prof. K SANDHYA RANI for fostering an excellent academic climate in our institution.

I also express my sincere gratitude to our respected Head of the Department Mr. P HARINADHA for his encouragement, overall guidance in viewing this project a good asset and effort in bringing out this project.

I would like to convey thanks to our guide at college Mrs. P SIVA LAKSHMI for his guidance, encouragement, co-operation and kindness during the entire duration of the course and academics.

My sincere thanks to all the members who helped me directly and indirectly in the completion of project work. I express my profound gratitude to all our friends and family members for their encouragement.

Index

Abstract

This project presents a app online video downloader. This app is used to download videos from various apps. The source link for the video is provided to the app it sends that link to the server from there the server processes the link and get the video to the app. This enables the user to get the requested video available. The app is implemented using Flutter for providing user interface and Flask is used in back-end to process the link with the help of selenium and bs4.

Introduction

There are many web pages in the internet for downloading a video from a site. For ex: we have in store site for downloading Instagram videos. It will be difficult for the user to maintain different apps for downloading different app videos. So, Here the idea for this app is instead of using different apps for downloading different app videos, This app provides various apps videos get downloaded using a single app, so that we can download videos from various apps.

Purpose

The main purpose of this app is to have a single app for downloading a video from different platforms. With the help of this app this app we can eliminate the system of having different platforms for different apps. User can enter the required source link of videos of different apps and can download the video into the mobile.

Scope

Today we have different platforms for downloading videos from different apps. Having number of apps for downloading a video from each app is difficult. In this All in One video downloader app we can have a single app to download videos from multiple platforms which in turn saves the memory storage and provides flexibility to the user.

Advantages

- User-Friendly
- Can download your favored track for offline enjoyment
- Free of cost
- Flexibility
- One platform for downloading 7 different app videos
- Saves memory storage as we are not maintaining different apps for different platforms.

Disadvantages

- Time consuming depends on the speed of internet
- Can only download the videos through internet

Requirement Specification

Hardware Configuration

Client Side:

Ram	512 MB
Hard disk	10 GB
Processor	1.0 GHz

Server side

Ram	1 GB
Hard disk	20 GB
Processor	2.0 GHz

Software Requirement

Front end	Flutter
Server side Language	Flask
Web Browser	Google Chrome
Operating System	Android
Software	Android Studio

Flutter

- Flutter is an open source framework by Google for building beautiful, natively compiled, multi-platform applications from a single codebase.
- To develop with Flutter, you will use a programming language called Dart.
- Dart focuses on front-end development, and you can use it to create mobile and web applications.
- Flutter consists of two important parts:
- An SDK (Software Development Kit): A collection of tools that are going to help you develop your applications. This includes tools to compile your code into native machine code (code for iOS and Android).
- A Framework (UI Library based on widgets): A collection of reusable UI elements (buttons, text inputs, sliders, and so on) that you can personalize for your own needs.

Flask

Flask is a micro web framework written in Python. It is classified as a micro framework 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.

Selenium

Selenium is an open source umbrella project for a range of tools and libraries aimed supporting browser automation. It provides a playback tool for authoring functional tests across most modern web browsers, without the need to learn a test scripting language.

Beautiful Soup

Beautiful Soup is a Python package for parsing HTML and XML documents. It creates a parse tree for parsed pages that can be used to extract data from HTML, which is useful for web scraping.

Design(Flow Chart)

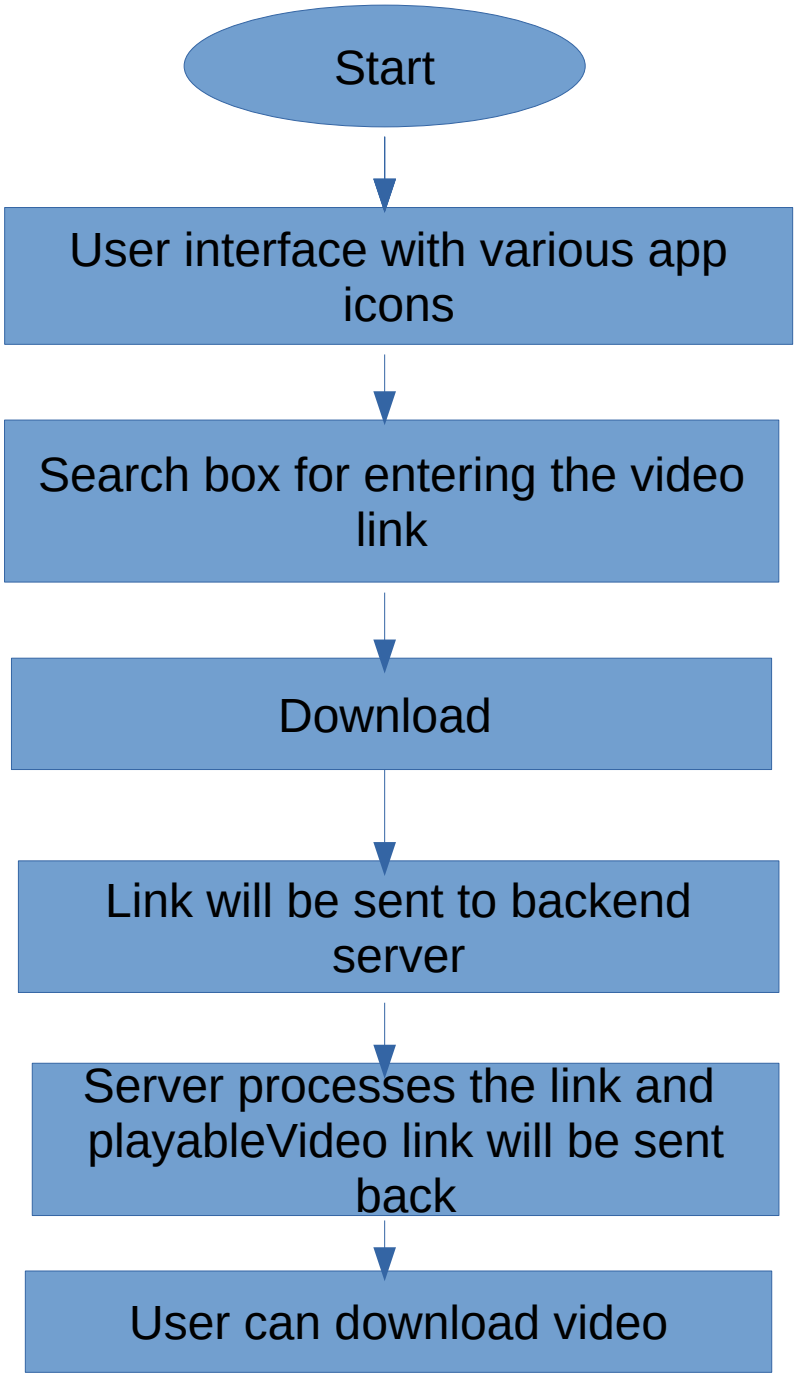


Fig 1. Working Flow of the application

This figure explains the working flow of the application from starting to ending.

Methodology

The working of full all in one video downloader app can be divided into two broad sections. The front-end part and back-end part .The working mechanism of all the parts is described in details below.

Front-end part:

The first and the most important part in this process is the user interface. This user interface is made using flutter. The app is implemented in Android Studio.

The algorithm is divided into following parts:

- Building layout for the app
- Collecting the icon images,fonts and importing them into app
- Providing functionality to the icon buttons
- Navigating to next screen
- Placing text field
- Placing a text button
- Launching the downloadable link

1. BUILDING A LAYOUT FOR THE APP

The first step is to build the layout for the app. The layout is build using Scaffold widget. This creates a base to the app screen. It includes the widgets app bar and body. The app bar is a horizontal bar that is displayed at the top of the screen which include background color and a text displayed with in it.

It is the primary and main part of the app where the content is displayed.

Here the main content is placing various app icon buttons.

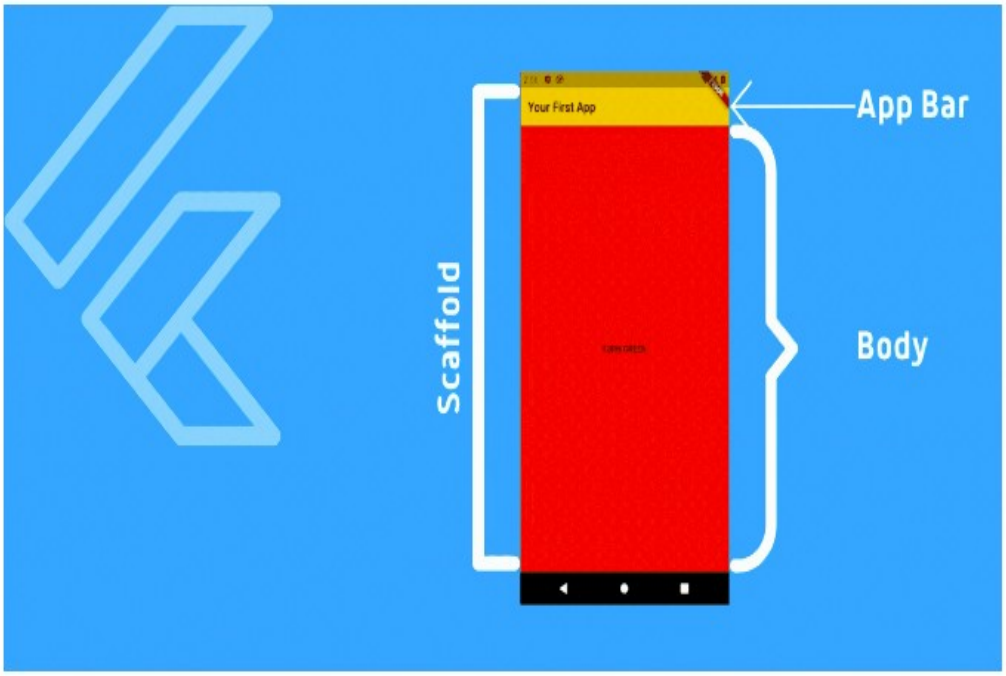


Fig 2: Structure of Scaffold

This figure shows that scaffold is a widget that consists of app bar and body. In the body the main content is placed.

2. COLLECTING THE APP ICONS, FONTS AND IMPORT THEM INTO APPLICATION

The next step after building a scaffold is to collect required icon images and fonts. Download the images of icons from the google and place them in a folder. Place this folder along with the app folders. Also download the required fonts and place them all in a folder and place the folder along with app folders. Go to the pubspec.yaml file and under the assets mention the required images and fonts. This will include the respective images and fonts into the app.

3. PROVIDING FUNCTIONALITY TO THE ICON IMAGES

Without providing functionality the icon images will act as normal images but we have to make them as icon buttons. The icon button is the button having an icon, and on tap it does something. What to do when icon button is pressed is declared by a method called “onPressed()”.

4. NAVIGATING TO NEXT SCREEN

Navigation means moving between pages. The Navigator.push() method is used to navigate/switch to a new route/page/screen. Here, the push() method adds a page/route on the stack and then manage it by using the Navigator. Again we use MaterialPageRoute class that allows transition between the routes using a platform-specific animation.

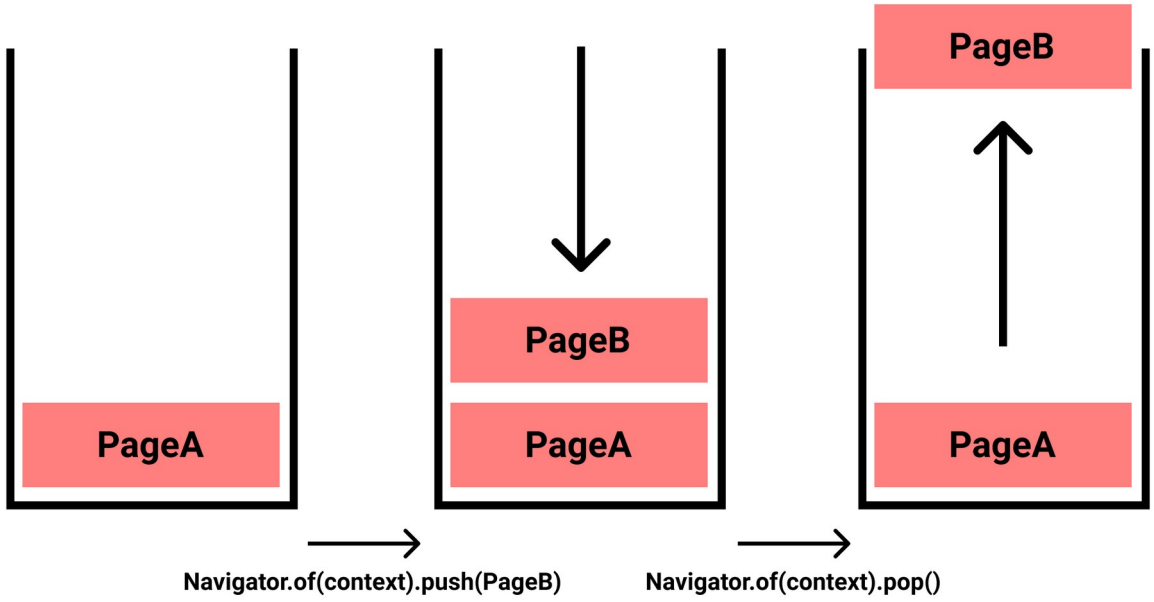


Fig:3 Navigation across screen

This figure shows that how the screens navigate from one activity to another activity

5. PLACING TEXT FIELD

A Text Field or Text Box is an input element which holds the alphanumeric data, such as name, password, address, etc. It is a GUI control element that enables the user to enter text information using a programmable code. It allows users to collect inputs from the keyboard into an app.

6. PLACING A TEXT BUTTON

Text button is a material design's button where user can add required text to the button. There are two required parameters. You have to pass a Widget as child, typically a Text or an Icon. The other required parameter is on Pressed, a callback which is called when the button is pressed. We can style the text button using 'style' property of text button. On Pressed function will send http request to back end server.

7. LAUNCHING THE DOWNLOADABLE LINK

After getting the link from back-end we launch the URL with the help of 'url_launcher' which will open the required video in the next screen. Here user can view,play and can download the video in required location.

BACK-END PART

1.Creating a server using Flask

2.Providing video links to the flask server

3.Getting the downloadable links with the help of Selenium.

4.Scraping the page source received from previous step to get the required download link

1. CREATING A SERVER USING FLASK

First of all we have to import "Flask" class from flask module. Create a app/server using Flask(). After creating the server we can use event handlers so that these methods will execute a function when the server get a request. app.route() is used to bind a URL to a function. We can add as many event handlers as we want. Once we added all the event handlers ,we can host the application/server either in local-host or web.app.run(). After sever is started we can send different web requests to the server then the server will handle all those requests to give output.

2. PROVIDING VIDEO LINKS TO THE FLASK SERVER

Front-end will send the URL as HTTP request to the server. This request is handled through a respective function defined in the back-end.

3. GETTING THE DOWNLOADABLE LINK USING SELENIUM

After getting the request,the respective function will create a web driver using selenium. Then another website will be opened in web-driver there the URL from the request will be converted into different formats. That page source will be saved.

4. SCRAPING THE PAGE SOURCE RECEIVED FROM PREVIOUS STEP TO GET THE REQUIRED DOWNLOADABLE LINK

with bs4 we scrape the page source we got from the previous step to get so that we get the downloadable URL of specific format and return that to front-end.

Implementation

main.dart

```
import 'package:flutter/material.dart';
import 'Screen1.dart';
void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
        home: Screen1()
    );
  }
}
```

screen1.dart

```
import 'package:flutter/material.dart';
import 'package:mini/Screen2.dart';

class Screen1 extends StatelessWidget{
  @override
  Widget build(BuildContext context){
    return Scaffold(
      backgroundColor: Colors.white,
      appBar: AppBar(
        backgroundColor: Colors.lightBlue,
        title: Center(
          child: Text(

            'All In One',
            style: TextStyle(
              fontFamily: 'Source Sans Pro',
              fontWeight: FontWeight.bold,
              fontSize: 30.0,
            ),
          ),
```

```
  ),
),
body: SafeArea(
  child: ListView(
    children: <Widget>[
    Center(
      child: Image(
        image: AssetImage(
          'images/back.jpg',
        ),
      )
    ),
      Row(
      children: <Widget>[
        Column(
          children: <Widget>[
            IconButton(
              icon: Image.asset(
                'images/Insta.png',
              ),
              iconSize: 60,
              padding: EdgeInsets.fromLTRB(20, 100, 10, 10),
              onPressed: () {
                Navigator.push(context,MaterialPageRoute(builder: (context)=>Screen2()));
              },
            ),

          ],
        ),
        SizedBox(
          width: 5,
        ),
        Column(
          children: <Widget>[
            IconButton(
              icon: Image.asset(
                'images/twitter.png',
              ),
              iconSize: 60,
              padding: EdgeInsets.fromLTRB(10, 100, 0, 10),
              onPressed: () {
```

```
    Navigator.push(context,MaterialPageRoute(builder: (context)=>Screen2()));
  },
),


IconButton(
  icon: Image.asset(
    'images/facebook.png',
  ),
  iconSize: 60,
  padding: EdgeInsets.fromLTRB(20, 10, 10, 10),
  onPressed: () {
    Navigator.push(context,MaterialPageRoute(builder: (context)=>Screen2()));
  },
),

],
),
SizedBox(
  width: 5,
),
Column(
children : <Widget>[
IconButton(
  icon: Image.asset(
    'images/dailymotion.png',
  ),
  iconSize: 60,
  padding: EdgeInsets.fromLTRB(30, 100, 10, 10),
  onPressed: () {
    Navigator.push(context,MaterialPageRoute(builder: (context)=>Screen2()));
  },
),
  IconButton(
    icon: Image.asset(
      'images/vimeo.png',
    ),
    iconSize: 60,
    padding: EdgeInsets.fromLTRB(20, 10, 0, 10),
    onPressed: () {
      Navigator.push(context,MaterialPageRoute(builder: (context)=>Screen2()));
    }
```

```
                ),

              ],
            ),

          SizedBox(
            width: 5,
          ),
          Column(
            children: <Widget>[
              IconButton(
                icon: Image.asset(
                  'images/tumblr.png',
                ),
                iconSize: 60,
                padding: EdgeInsets.fromLTRB(30, 100, 10, 10),
                onPressed: () {
                  Navigator.push(context,MaterialPageRoute(builder: (context)=>Screen2()));
                },
              ),
              IconButton(
                icon: Image.asset(
                  'images/youtube.png',
                ),
                iconSize: 60,
                padding: EdgeInsets.fromLTRB(20, 10, 0, 10),
                onPressed: () {
                  Navigator.push(context,MaterialPageRoute(builder: (context)=>Screen2()));
                },
              ),
            ],
          ),
            ],
          ),
        ],
      ),
    ),
  );
 }
}
```

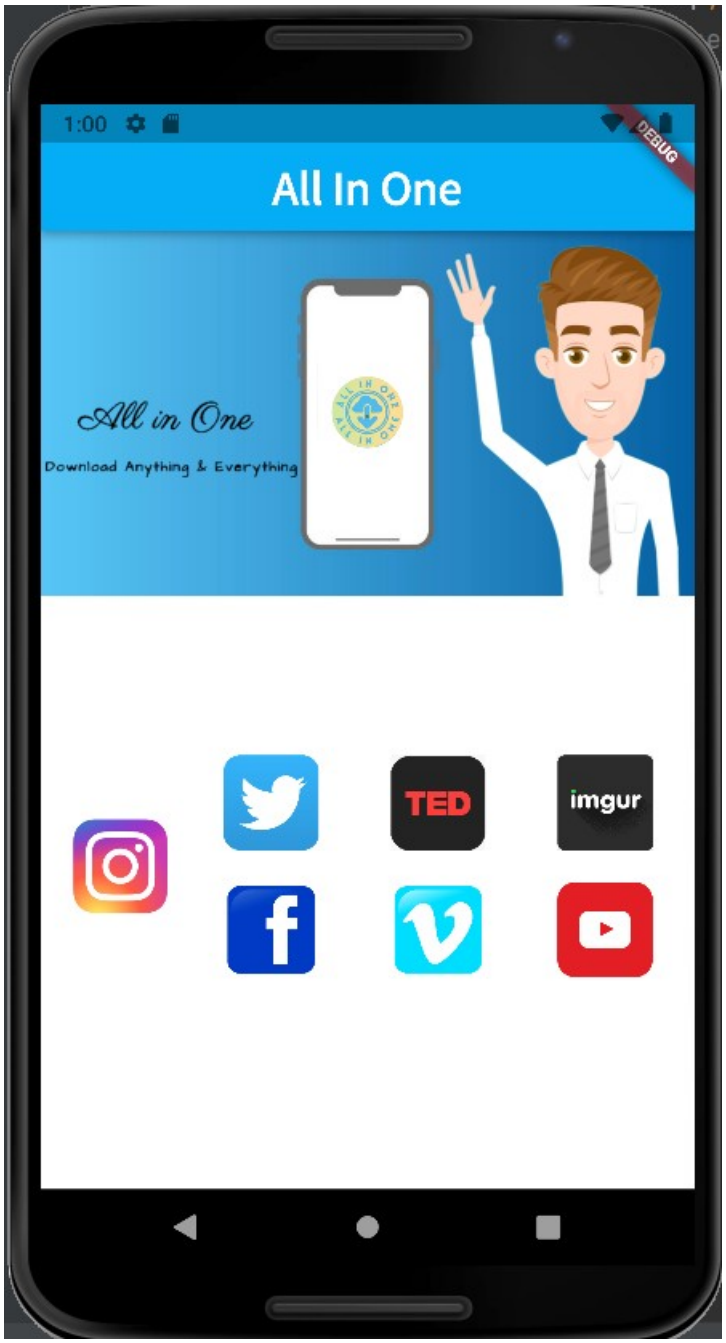


Fig:4 output screen1

This screen shows an image and 7 app icons when an user click on anyone of these icon it navigates to screen2

screen2.dart

```
import 'package:flutter/material.dart';
import 'package:mini/backend.dart';
import 'package:url_launcher/url_launcher.dart';

class Screen2 extends StatelessWidget {
  final urlController=TextEditingController();
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      backgroundColor: Colors.white,
      appBar: AppBar(
        backgroundColor: Colors.lightBlue,
        title: Center(
          child: Text(
            'Videos for You',
            style: TextStyle(
              fontFamily: 'Pacifico',
              fontWeight: FontWeight.bold,
              fontSize: 30.0,
            ),
          ),
        ),
      ),
      body: SafeArea(
        child: ListView(
          padding: EdgeInsets.fromLTRB(32, 170, 32, 32),
          children: [
            TextField(
              controller: urlController,
              textInputAction: TextInputAction.done,
              keyboardType: TextInputType.url,
              decoration: InputDecoration(
                labelText: 'Enter The Link',
                prefixIcon: Icon(Icons.add_link),
                prefixIconColor: Colors.red,
                suffixIcon: IconButton(
                  icon: Icon(Icons.close),
                  onPressed: (){
                      urlController.clear();
                  },
                ),
```

```
            border: OutlineInputBorder(),

          ),
           ),
           SizedBox(height: 20),
           TextButton(
             onPressed: () async {
               var a = await otherSites(urlController.text);

               var url = Uri.parse(a);
               if (await canLaunchUrl(url)){
                 await launchUrl(url,mode: LaunchMode.externalApplication);
               }
             },
             style: TextButton.styleFrom(
               minimumSize: Size(200, 60),
               backgroundColor: Colors.lightBlue,
             ),
             child: Text(
               'Download',
               style: TextStyle(
                 color: Colors.white,
                 fontFamily: 'Source Sans Pro',
                 fontWeight: FontWeight.bold,
                 fontSize: 30.0,
               ),
             ),
           ),
         ],
       ),
     ),
   );
}
```

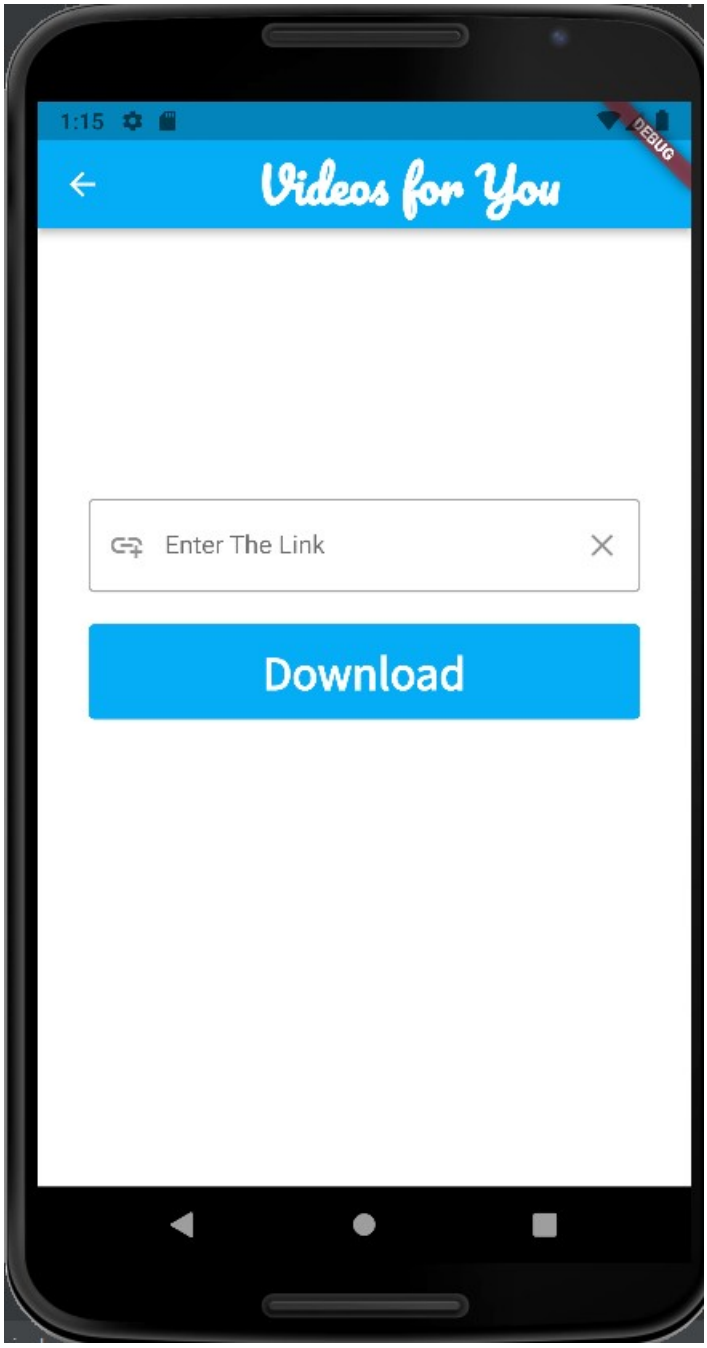


Fig:5 Output screen2

This figure shows the output of screen2 in which user can paste the source link and can click the download button.

backend.dart

```
import 'dart:convert';
import 'package:http/http.dart' as http;
import 'package:url_launcher/url_launcher.dart';
otherSites(String url) async
 {
   if (url.contains("youtube")) {
     http.Response res = await http.get(
         Uri.parse("http://10.0.2.2:8080/youtube/?url=$url"));
     var decod = jsonDecode(res.body);
     //print(decod["url"]);
     return decod["url"];
   }
   if(url.contains("instagram")){
        http.Response res = await http.get(Uri.parse("http://10.0.2.2:8080/instagram/?
url=$url"));
     var decod = jsonDecode(res.body);
     return decod["url"];
   }
   if(url.contains("facebook")){
         http.Response res = await http.get(Uri.parse("http://10.0.2.2:8080/facebook/?
url=$url"));
     var decod = jsonDecode(res.body);
     return decod["url"];
   }
   if(url.contains("twitter")){
           http.Response res = await http.get(Uri.parse("http://10.0.2.2:8080/twitter/?
url=$url"));
     var decod = jsonDecode(res.body);
     return decod["url"];
   }
   if(url.contains("imgur")){
           http.Response res = await http.get(Uri.parse("http://10.0.2.2:8080/imgur/?
url=$url"));

var decod = jsonDecode(res.body);
     return decod["url"];
   }
   if(url.contains("vimeo")){
           http.Response res = await http.get(Uri.parse("http://10.0.2.2:8080/vimeo/?
url=$url"));
```

```
    var decod = jsonDecode(res.body);

return decod["url"];
  }
  if(url.contains("ted.com")){
    http.Response res = await http.get(Uri.parse("http://10.0.2.2:8080/ted/?url=$url"));
    var decod = jsonDecode(res.body);
    return decod["url"];
  }

 }
```

App.py

```
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
import uu, os
from datetime import datetime
chrome_options = Options()
chrome_options.add_argument('--no-sandbox')
chrome_options.add_argument('--disable-dev-shm-usage')
chrome_options.add_argument("--headless")
from flask import Flask, request
from time import sleep
from bs4 import BeautifulSoup
from selenium.webdriver.chrome.options import Options

app = Flask(__name__)


@app.route("/")
def ram():
  return "vgdgd"


@app.route("/twitter/", methods=["GET", "POST"])
def fun():
  try:
    url = request.args.get("url")
                                                    driver                    =
webdriver.Chrome("C:/Users/nrthe/OneDrive/Desktop/chromedriver_win32/
chromedriver.exe",options=chrome_options)
```

```
        driver.get("https://allinonedownloader.com/")
        sleep(2)
        search_bar = driver.find_element(By.NAME,"url")

search_bar.send_keys(url)
        search_bar.send_keys(Keys.RETURN)
        sleep(5)
        html = driver.page_source
        driver.close()
        soup = BeautifulSoup(html)
        table_body = None
        while (table_body == None):
            sleep(2)
            table_body = soup.find('tbody', attrs={'class': 'video-links'})
        rows = table_body.find_all('tr')
        a = rows[0].find_all("td")[3].find("a").get("href")
        return {"url":a}
    except:
        return {"url":"error"}

@app.route("/ted/", methods=["GET", "POST"])
def ted():

 try:
        url = request.args.get("url")
                                                                    driver          =
webdriver.Chrome("C:/Users/nrthe/OneDrive/Desktop/chromedriver_win32/
chromedriver.exe",options=chrome_options)
        driver.get("https://allinonedownloader.com/")
        sleep(2)
        search_bar = driver.find_element(By.NAME,"url")
        search_bar.send_keys(url)
        search_bar.send_keys(Keys.RETURN)
        sleep(10)
        html = driver.page_source
        driver.close()
        soup = BeautifulSoup(html)
        table_body = None
        while (table_body == None):
            sleep(2)
            table_body = soup.find('tbody', attrs={'class': 'video-links'})
        rows = table_body.find_all('tr')
        print(rows)
```

```
        a = rows[0].find_all("td")[3].find("a").get("href")
        return {"url":a}
    except:
        return {"url":"error"}
@app.route("/facebook/", methods=["GET", "POST"])
def fb():
    try:
        url = request.args.get("url")
                                                                driver =
webdriver.Chrome("C:/Users/nrthe/OneDrive/Desktop/chromedriver_win32/
chromedriver.exe",options=chrome_options)
        driver.get("https://allinonedownloader.com/")
        sleep(2)
        search_bar = driver.find_element(By.NAME,"url")

       search_bar.send_keys(url)
        search_bar.send_keys(Keys.RETURN)
        sleep(5)
        html = driver.page_source
        driver.close()
        soup = BeautifulSoup(html)
        table_body = None
        while (table_body == None):
            sleep(2)
            table_body = soup.find('tbody', attrs={'class': 'video-links'})
        rows = table_body.find_all('tr')
        a = rows[0].find_all("td")[3].find("a").get("href")
        return {"url":a}
    except:
        return {"url":"error"}
@app.route("/vimeo/", methods=["GET", "POST"])
def vim():
    try:
        url = request.args.get("url")
                                                                driver =
webdriver.Chrome("C:/Users/nrthe/OneDrive/Desktop/chromedriver_win32/
chromedriver.exe",options=chrome_options)
        driver.get("https://allinonedownloader.com/")
        sleep(2)
        search_bar = driver.find_element(By.NAME,"url")
        search_bar.send_keys(url)
        search_bar.send_keys(Keys.RETURN)
```

```
        sleep(10)
        html = driver.page_source
        driver.close()
        soup = BeautifulSoup(html)
        table_body = None

        while (table_body == None):

sleep(2)

 table_body = soup.find('tbody', attrs={'class': 'video-links'})
        rows = table_body.find_all('tr')
        print(rows)

a = rows[0].find_all("td")[3].find("a").get("href")
        return {"url":a}
    except:
        return {"url":"error"}
@app.route("/imgur/",methods=["GET","POST"])
def img():
    id = request.args.get("url")[-7:]
    s = "https://i.imgur.com/"+id+".mp4"
    return {"url":s}

@app.route("/youtube/", methods=["GET", "POST"])
def youtube():
    try:
        url = request.args.get("url")
                                                                    driver          =
webdriver.Chrome("C:/Users/nrthe/OneDrive/Desktop/chromedriver_win32/
chromedriver.exe",options=chrome_options)
        driver.get("https://youtils.cc/yt-dlp/main/en")
        sleep(2)
        search_bar = driver.find_element(By.XPATH,'//*[@id="app"]/div[2]/div/div/input')
        search_bar.send_keys(url)
        driver.find_element(By.XPATH,'//*[@id="app"]/div[2]/div/div/div[2]/button').click()
        id = url[-11:]
        driver.get(f"https://youtils.cc/yt-dlp/result/{id}/en")
        html = driver.page_source
        driver.close()
        soup = BeautifulSoup(html)
        table = soup.find("table",attrs={"class":"table"})
```

```
i = table.find_all("tbody")[4].find_all("a")[0].get("href")
        return {"url":i}
    except:
        return "error"



@app.route("/instagram/", methods=["GET", "POST"])
def insta():

try:
        url = request.args.get("url")
driver=      webdriver.Chrome("C:/Users/nrthe/OneDrive/Desktop/chromedriver_win32/
chromedriver.exe",options=chrome_options)
        driver.get("https://allinonedownloader.com/")
        sleep(2)
        search_bar = driver.find_element(By.NAME,"url")
        search_bar.send_keys(url)
        search_bar.send_keys(Keys.RETURN)
        sleep(5)
        html = driver.page_source
        driver.close()
        soup = BeautifulSoup(html)
        table_body = None
        while (table_body == None):
            sleep(2)
            table_body = soup.find('tbody', attrs={'class': 'video-links'})
        rows = table_body.find_all('tr')
        a = rows[0].find_all("td")[3].find("a").get("href")
        return {"url":a}
    except:
        return {"url":"error"}
app.run(host="0.0.0.0", port=8080)
```

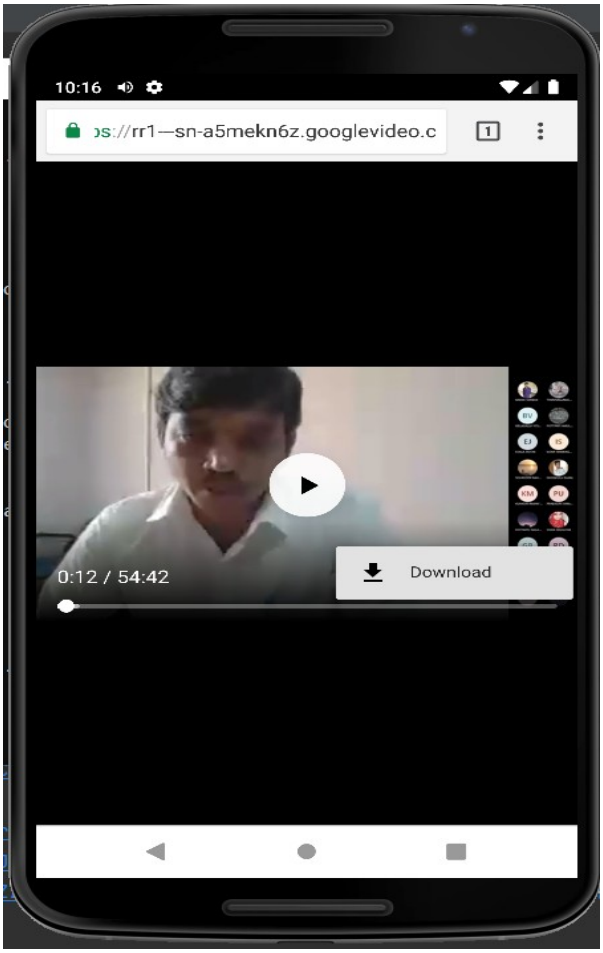


Fig:6 Playable video with download option

This figure shows that the video returned from the back end which can be played or downloaded.

Project Output

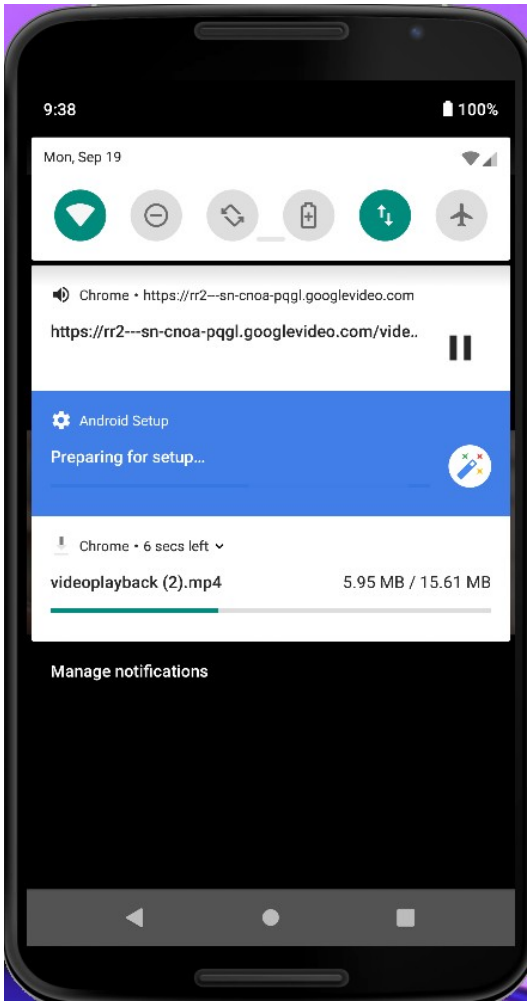


Fig:7 Downloading is showing in notification

This figure shows that that when the download button is clicked then then the video download is started and the progress is showed in Notification bar

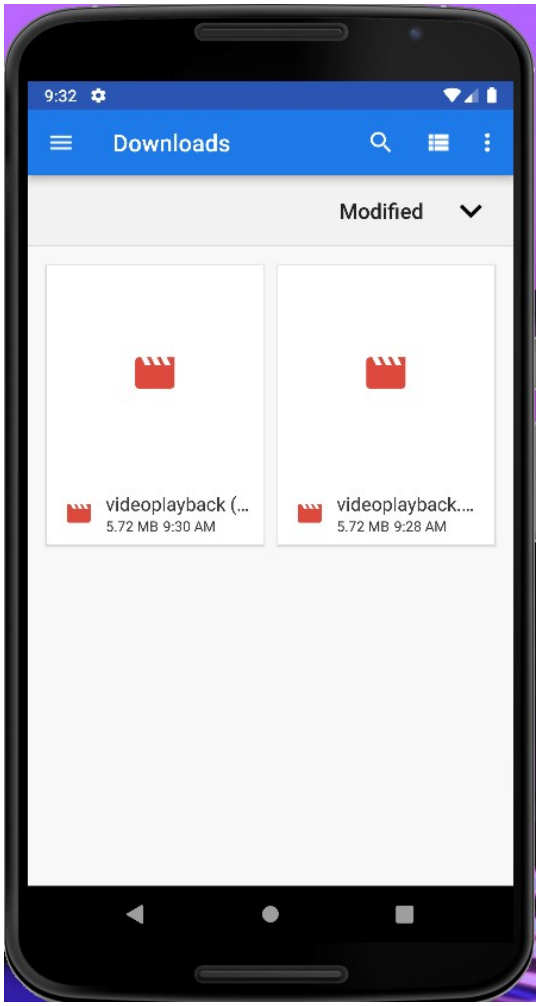


Fig-8 Video is downloaded in the path

This figure shows that the video is stored in internal storage of the user and user can now watch the video without any internet

Conclusion

We have implemented the all in one video downloader app. Our app is successfully downloading the video based on the link we provided. We have applied our testing on many videos and found that it successfully downloading. The project was designed keeping in mind that making a single app for downloading videos could replace the system of having different apps for different app videos. This project was a success in downloading the video which user wants. The app could be able to download videos from different apps and it has the ability to process the entered link and can get the playable video to the user activity. It provides the ability to get the video downloaded into the mobile.

References

- Flutter course videos from Udemy
- Standard website of Flutter http://www.flutter.dev
- Documentation on Flask in https://www.flask.palletsprojects.com
- Docs on selenium- https://www.selenium-python.readthedocs.io
- Docs on Beautiful soup- https://www.beautiful-soup-4.readthedocs

********** THANK YOU **********