Government Polytechnic,Nashik

(An Autonomous Institute of Government of Maharashtra)

New building campus ,Samangaon road ,Nashik road

PROJECT REPORT

"LearnoScope : An Educational Video Sharing Platform"

FOR THE COURSE
THIRD YEAR DIPLOMA IN INFORMATION TECHNOLOGY

Submitted By:

Mr. Madhukrishna Nipankar	(195138)
Mr. Yadnesh Gangurde	(195118)
Mr. Vedant Kulkarni	(195161)
Mr. Amit Sali	(195151)
Mr. Kiran Dahake	(195109)

Guided By:

Mrs. N.S Nikale

Submitted To:

Government Polytechnic, Nashik

For academic Year

2021-2022

Government Polytechnic , Nashik

(An Autonomous Institute of Government of Maharashtra)
New building campus ,Samangaon road ,Nashik road

CERTIFICATE

This is to certify that, the Project Report on **"LearnoScope: An educational video sharing platform"** has been successfully completed by

Mr. Madhukrishna Nipankar	(195138)
Mr. Yadnesh Gangurde	(195118)
Mr. Vedant Kulkarni	(195161)
Mr. Amit Sali	(195151)
Mr. Kiran Dahake	(195109)

In fulfilment of requirement of Diploma in "Computer Technology" from Government Polytechnic, Nashik during academic year 2017-2018. They have satisfactory completed project

Guided by
(Prof. N.S.Nikale)

H.O.D
(Prof. N.S.Nikale)

Principal
(Prof. D. P. Nathe)

ACKNOWLEDGEMENT

I express my deep sense of gratitude and respect and respect to my guide Prof. S.N. Nikale for this valuable guidance and help during Project work.

I am thankful for her coherent encouragement during the entire period that has been an inspiration for me In achieving these goals.

I am very thankful to the head of department Prof. S.N. Nikale Information Technology , for this valuable guidance and constant source of inspiration. I also wish to express my Deepest gratitude to all staff members of the Information Technology Department for their Support.

I also want to thank all my colleagues for contribution in making this seminar successful

Mr .Madhukrishna Nipankar	(195138)
Mr. Yadnesh Gangurde	(195118)
Mr. Vedant Kulkarni	(195161)
Mr. Amit Sali	(195151)
Mr. Kiran Dahake	(195109)

ABSTRACT

During the period of Lock-down, In times where schools, colleges, major learning institutes were closed, still enthusiasts gained valuable knowledge through remote learning. Remote learning has been present on the internet for a long time but during lockdown enthusiasts fully utilised resources to gain and improve their knowledge. This shows us that remote learning is such a blessing for knowledge seekers. Many platforms are out there, which are quite good at their job, but as people are preferring online learning more and more, major video sharing platforms are trying to make more profit by putting advertisements on that particular platform which actually creates a lot of chaos. Also there are suggestions of irrelevant videos . Putting video advertisements in a video sharing platform as well as the suggestion feed distracts the learners from the actual content. Which makes student's/learner's/enthusiast's focus far lower than their potential which is directly proportional to lesser productivity.

By observing this situation there is a need to make a platform where learners can learn in-demand skills, on demand, without advertisements, without unnecessary clutter, without irrelevant suggestions, with the quality and the productivity that learner deserves and to deliver that experience and overcome problems with rest of the platforms , LearnoScope is ideal platform to rely on.

INDEX

Chapter :-1 INTRODUCTION

1.1 Problem Definition

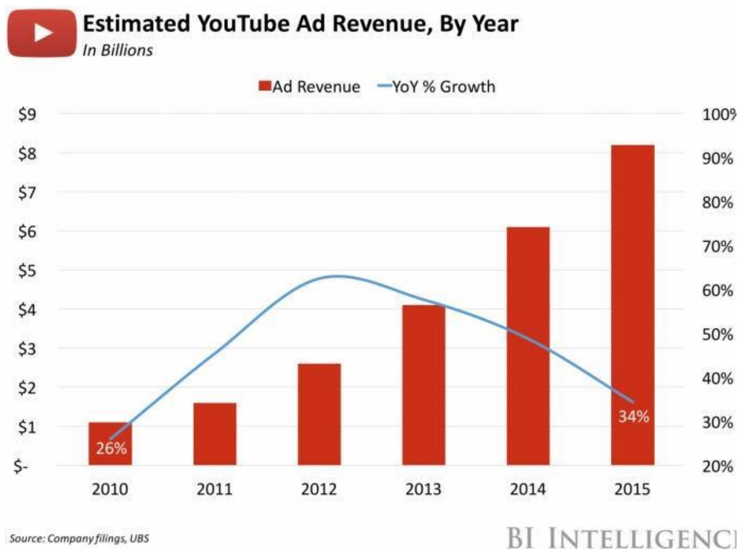
In the days of lockdown we have seen the problems we faced during the COVID-19 phase where everything was intentionally closed due to covid. schools/colleges/institutes were closed. which directly means students were unable to learn new skills and gain knowledge.

In such a difficult time, a lot of students were unable to learn. but technology gives them rays of hope. remote education became the door to knowledge in days of closed gates. Remote education was present on the internet for a long time. but in lockdown we used online education to its full potential. Remote education showed us new possibilities of learning. Such as video lectures ,online notes ,digital study material which actually made learning possible on digital devices such as laptop,tablet,smartphones

Online video lectures replaced the actual live lectures. Teachers and students can make very good interactions with each other with the help of e-learning. Video tutorials also played a major role to share and increase the knowledge of the students who were willing to learn without any compromise. Learners can watch this video tutorial on their preferable time.

as the popularity of online video tutorials increased. The user traffic also increased on these platforms. more users == more views. so platforms started showing more and more advertisements to make more profit.

The problem isn't the Ads that they display on the platform. The main problem is the distraction among learners created by it. initially when this concept got implemented. The number of advertisements on each video was moderate. but over time the number of ads per video increased which really messes up the focus of the learner.



You can see the above image of a popular platform. These are statistics of revenue made by ads on the platform. as the bar is increasing No of ads. per video. We can say these increasing bars point towards the rise of distractions caused by ads.

To overcome the problems with existing video sharing platforms, And create the whole new video sharing platform which is solely focused on content. With features of note/file sharing, note taking, profile sharing, bookmark a video, discussion forum creation. which gives learners more flexibility to learn skills in an organised manner.

1.2 Existing System

While studying another video sharing platform like youtube we found that youtube is the great platform for entertainment but most of the time the students prefer youtube for learning purposes also . As youtube is a free platform it provides free content for all users (including students) but also there are a lot of distractions which can easily distract the students from study to another world. This is a general problem faced by many students across the world. There are also other platforms like unacademy, Udemy etc which provide free courses for different subjects but they are all paid courses which can't fit everyone's budget. So in short there is no platform for video sharing which provides completely free educational video content.

Also while studying the different Video sharing platform we found following some points :

1. The video sharing platform is youtube is generalized not specialized i.e there is no any fixed categories for the video content.
2. Platform which provides different educational content that provides paid content that

can be expensive.

3. Platforms like youtube are only allowed to share educational videos, not their notes.
4. Platform like youtube doesn’t provide any discussion forum so that students can submit their doubts.
5. Not any facility to store the notes while playing the video.
6. More distraction for students.

1.3 Proposed System

LearnoScope is an education sharing platform via the internet, network, or standalone computer. LearnoScope is basically the video sharing platform solely focused on content. without any digital distractions, convey skills and knowledge. LearnoScope is an E-learning platform. Which fulfills the lack of features that other platforms are missing. E-learning refers to using electronic applications and processes to learn. E-learning includes all forms of electronically supported learning and teaching. E-learning applications and processes include web-based learning, computer-based learning, virtual education opportunities and digital collaboration. Content is delivered via the Internet, intranet, audio or video tape.

LearnoScope is entirely based on educational-video sharing theme. Where all users have their account on the site, each user gets treated as a regular user account, which holds uploaded videos of that particular user. With title, description, and attachment files like pdf, photos etc. attachments are for study purposes.

A user can search a particular video by entering the topic name in the search bar and results will be displayed according to your input.

LearnoScope has many features built in favor of learners, such as notes sharing, quick notes, dark mode, discussion forum, bookmark.

When a user uploads a video he/she is also able to attach study material so other viewers(learner) can also get study material in order to learn that skill properly. Quick notes is something that boosts the productivity of users, learners can take notes on the go without having any separate note taking medium. And user's data will be saved in the backend so saved quick notes can be accessed later.

Chapter :-2 LITERATURE SURVEY

We have been gathering information about how to integrate all the information a user needs to and what requirements it will generate to do so. As we are thinking of developing a web application it gives rise to many things from language to be use , Operating system ,various API to be used to access different kind of data

2.1 What is a Web Application ?

A web application is a computer program that utilizes web browsers and web technology to perform tasks over the Internet .Millions of businesses use the Internet as a cost-effective communications channel. It lets them exchange information with their target market and make fast, secure transactions. However, effective engagement is only possible when the business is able to capture and store all the necessary data, and have a means of processing this information and presenting the results to the user .Web applications use a combination of server-side scripts (PHP ,DJango ASP) to handle the storage and retrieval of the information, and client-side scripts (JavaScript and HTML) to present information to users. This allows users to interact with the company using online forms, content management systems, shopping carts and more. In addition, the applications allow employees to create documents, share information, collaborate on projects, and work on common documents regardless of location or device.

2.2 How does a Web application work ?

Web applications are usually coded in browser-supported languages such as JavaScript and HTML as these languages rely on the browser to render the program executable. Some of the applications are dynamic, requiring server-side processing. Others are completely static with no processing required at the server.The web application requires a web server to manage requests from the client, an application server to perform the tasks requested, and, sometimes, a database to store the information. Application server technology ranges from ASP.NET, ASP and ColdFusion, to PHP and JSP.

Here's what a typical web application flow looks like:

1. User triggers a request to the web server over the Internet, either through a web browser or the application’s user interface
2. Web server forwards this request to the appropriate web application server

3. Web application server performs the requested task – such as querying the database or processing the data – then generates the results of the requested data
4. Web application server sends results to the web server with the requested information or processed data
5. Web server responds back to the client with the requested information that then appears on the user's display

2.2.1 Web Server

A web server is software and hardware that uses HTTP (Hypertext Transfer Protocol) and other protocols to respond to client requests made over the World Wide Web. The main job of a web server is to display website content through storing, processing and delivering web pages to users. Besides HTTP, web servers also support SMTP (Simple Mail Transfer Protocol) and FTP (File Transfer Protocol), used for email, file transfer and storage.

2.3 What is a Single Page Application ?

A single-page application is an app that doesn't need to reload the page during its use and works within a browser. Think of the apps you use daily: Facebook, Google Maps, Gmail, Twitter, Google Drive, or even GitHub. All these are examples of a SPA. One of the best advantages of a correctly-configured SPA is the user experience (UX), where the user enjoys a natural environment of the app without having to wait for the page reloads and other things. You remain on the same page, which is powered by the JavaScript programming language. The main advantage of single-page applications is its speed. Most resources SPA needs (HTML + CSS + Scripts) are loaded at the launch of the app and don't need to be reloaded during the usage. The only thing that changes is the data that is transmitted to and from the server. As a result, the application is very responsive to the user's queries and doesn't have to wait for client-server communication all the time.

- **Feature of Single page application**

1. Fast and responsive
2. Caching capabilities
3. Linear user experience

4. Debugging with Chrome

2.4 Technology

1. React JS (Frontend) :

ReactJS is one of the most popular JavaScript front-end libraries which has a strong foundation and a large community .ReactJS is a **declarative**, **efficient**, and flexible **JavaScript library** for building reusable UI components. It is an open-source, component-based front end library which is responsible only for the view layer of the application. It was initially developed and maintained by Facebook and later used in its products like WhatsApp & Instagram .ReactJS includes all the topics like Pros and Cons of ReactJS, ReactJS JSX, ReactJS Components, ReactJS State, ReactJS Props, ReactJS Forms, ReactJS Events, ReactJS Animation and many more.

2. Django (Backend) :

Django is a web application framework written in Python programming language. It is based on the MVT (Model View Template) design pattern. Django is very demanding due to its rapid development feature. It takes less time to build an application after collecting client requirements .This framework uses a famous tagline : **The web framework for perfectionists with deadlines .**By using Django, we can build web applications in very less time. Django is designed in such a manner that it handles much of the configuration automatically, so we can focus on application development only.

3. SQLite (Database) :

SQLite is an embedded relational database management system. It is self-contained, serverless, zero configuration and transactional SQL database engine.SQLite is free to use for any purpose commercial or private. In other words, "SQLite is an open source, zero-configuration, self-contained, stand alone, transaction relational database engine designed to be embedded into an application".SQLite is different from other SQL databases because unlike most other SQL databases, SQLite does not have a separate server process. It reads and writes directly to ordinary disk files. A complete SQL database with multiple tables, indices, triggers, and views, is contained in a single disk file.

Chapter :-3 Software requirement specification

<u>Minimum System Requirements: -</u>

Hardware Requirements: PC

➢ System	:	Intel Core i3, Android 8.0 above
➢ Hard Disk	:	40 GB (PC), 18GB(Phone)
➢ RAM	:	2 GB of RAM.

Software Requirements: Smartphone

➢ Hard Disk	:	40 GB (PC), 18GB(Phone)
➢ RAM	:	2 GB of RAM.
➢ Browsers		

Firefox	Mozilla Foundation	98
Chrome	Google	99
Opera	Opera Software	83

(these are the latest versions of smartphone browsers testing performed on this versions)

Software Requirements:

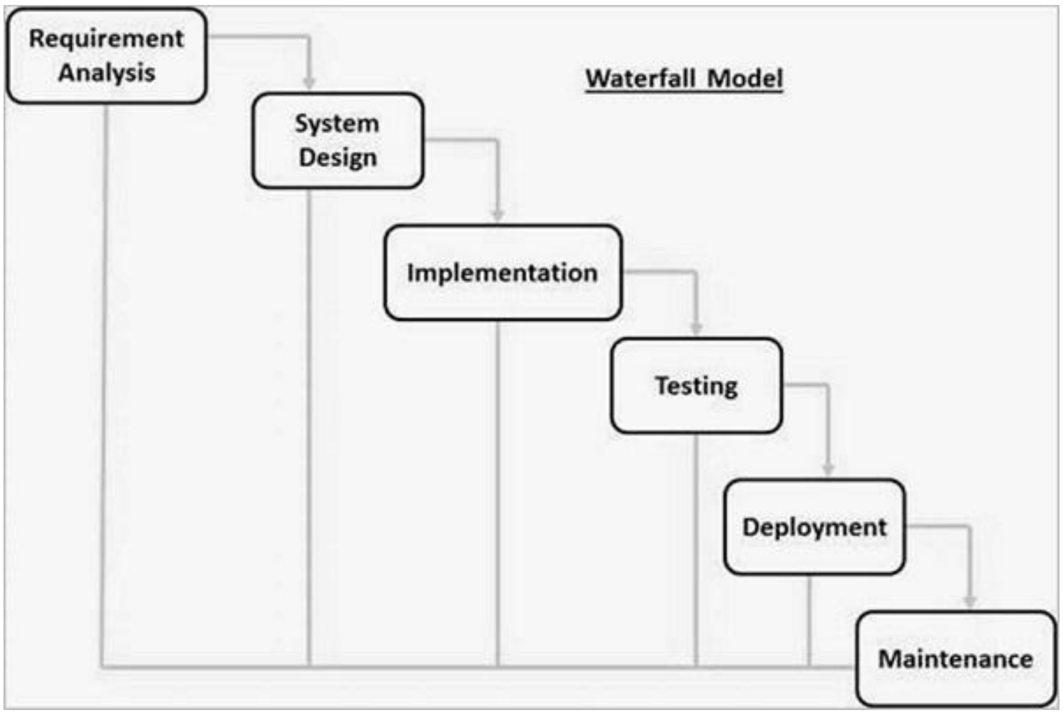
➢ Software for development	:	Browser (Chrome, Firefox)
➢ Coding Language	:	React JS (JavaScript, HTML, CSS, Bootstrap)
➢ Back-end	:	Django,Python
➢ Web browser	:	Chrome, Firefox, Safari

Chapter :-4 DESIGN

4.1 Analysis Model :

The Waterfall approach was the first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

The following illustration is a representation of the different phases of the Waterfall Model.



The sequential phases in Waterfall model are −

- **Requirement Gathering and analysis** − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
- **System Design** − The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
- **Implementation** − With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
- **Integration and Testing** − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
- **Deployment of system** − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.

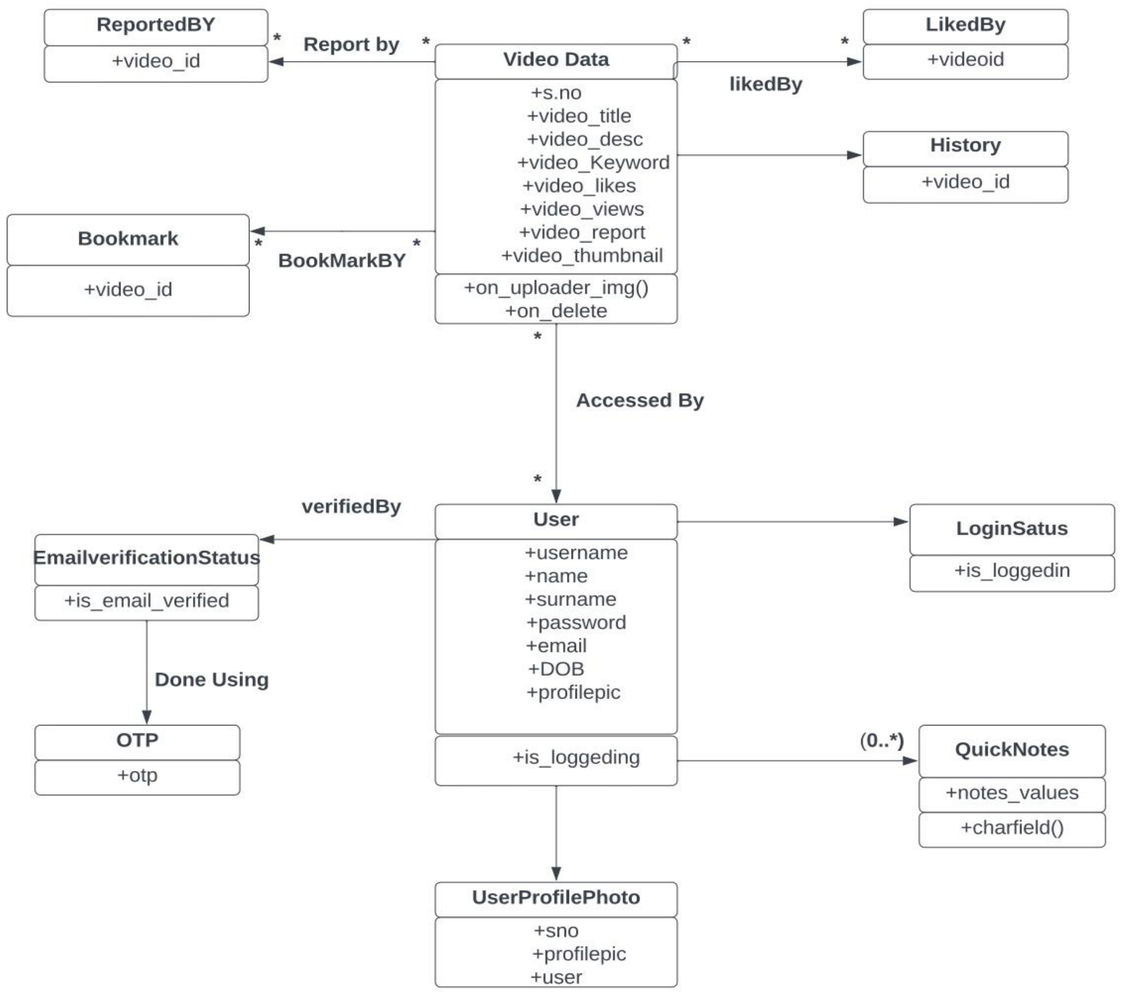
- **Maintenance** − There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

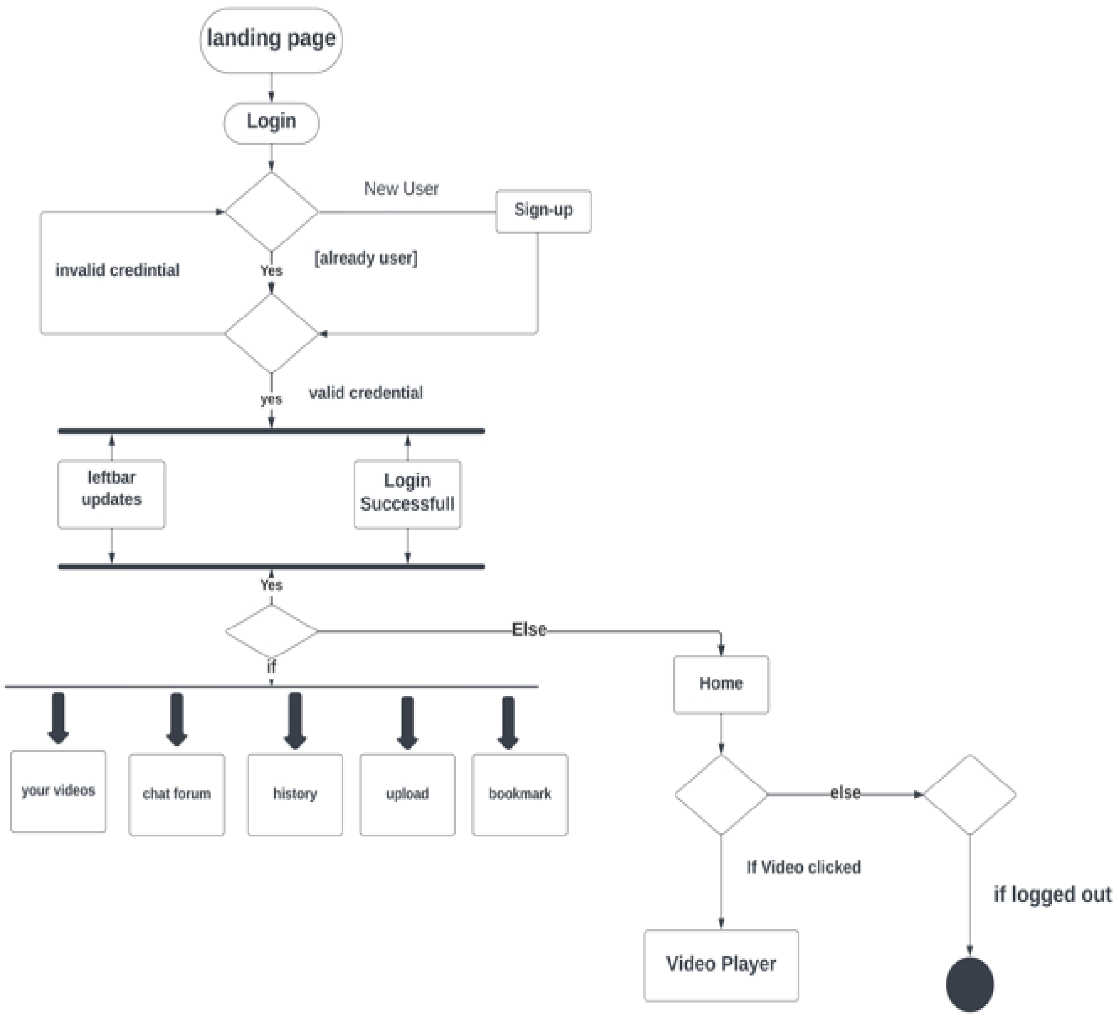
All these phases are cascaded to each other in which progress is seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined set of goals are achieved for the previous phase and it is signed off, so the name "Waterfall Model". In this model, phases do not overlap.

4.2 UML Diagram

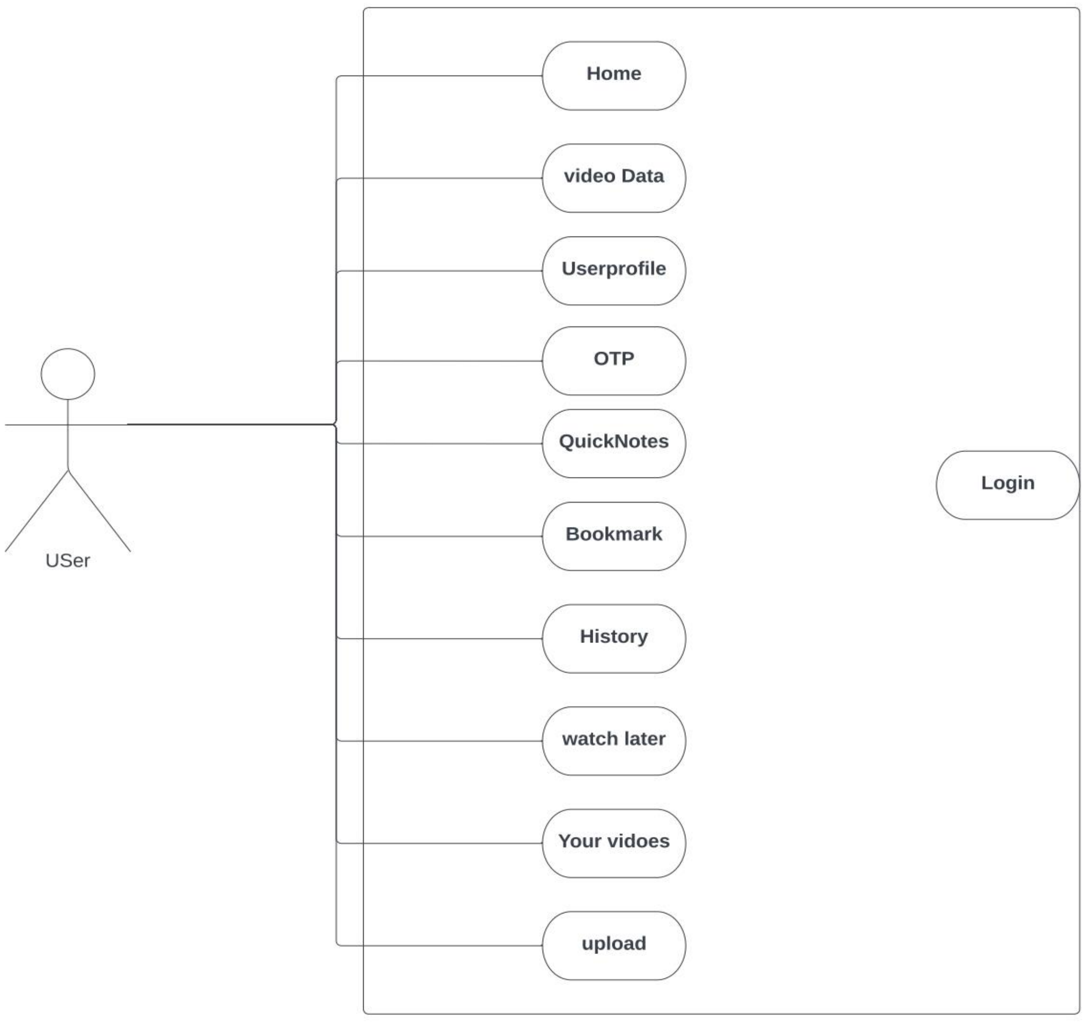
4.2.1 Class Diagram

4.2.2 Activity Diagram





4.2.3 Use-case Diagram


Home
video Data
Userprofile
OTP
QuickNotes
Bookmark
History
watch later
Your vidoes
upload
Login
USer

4.3 Project Plan

Date / Phase	7-3-2022 to 16-3-2022	17-3-2022 to 22-3-2022	25-3-2022 to 30-4-2022	1-5-2022 to 7-5-2022	8-5-2022 to 10-5-2022
Phase 1	🟦				
Phase 2		🟦			
Phase 3			🟦		
Phase 4				🟦	
Phase 5					🟦

Chapter :- 5.IMPLEMENTATION

5.1 Coding

5.1.1—Index.html

```
<!DOCTYPE html>
<html lang="en">


<head>
  <meta charset="utf-8" />
  <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <meta name="theme-color" content="#000000" />
  <meta name="description" content="Web site created using create-react-app" />
  <!-- <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" /> -->
  <link rel="preconnect" href="https://fonts.googleapis.com">
  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
  <link
    href="https://fonts.googleapis.com/css2?family=Cinzel:wght@700;900&family=Fruktur&family=Manrope:wght@200&family=Poppins:wght@200;300;400&family=Shadows+Into+Light&family=Sigmar+One&family=Urbanist:wght@200&family=Zen+Kurenaido&display=swap"
    rel="stylesheet">


  <!-- Bootstrap CSS -->
        <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" rel="stylesheet"
        integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbrCEXSU1oBoqyl2QvZ6jIW3" crossorigin="anonymous">


  <!-- Google Fonts -->
  <link
     href="https://fonts.googleapis.com/css?family=Open+Sans:300,300i,400,400i,600,600i,700,700i|Montserrat:300,300i,400,400i,500,500i,600,600i,700,700i|Poppins:300,300i,400,400i,500,500i,600,600i,700,700i"
    rel="stylesheet">
  <!-- LANDING PAGE CSS -->
  <link href="assets/vendor/aos/aos.css" rel="stylesheet">
  <link href="assets/vendor/bootstrap/css/bootstrap.min.css" rel="stylesheet">
  <link href="assets/vendor/bootstrap-icons/bootstrap-icons.css" rel="stylesheet">
  <link href="assets/vendor/boxicons/css/boxicons.min.css" rel="stylesheet">
  <link href="assets/vendor/glightbox/css/glightbox.min.css" rel="stylesheet">
  <link href="assets/vendor/remixicon/remixicon.css" rel="stylesheet">
  <link href="assets/vendor/swiper/swiper-bundle.min.css" rel="stylesheet">
  <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
  <!-- END LANDING PAGE CSS -->

  <title>LearnoScope</title>
</head>

<body id="body">
  <noscript>You need to enable JavaScript to run this app.</noscript>
  <div id="root"></div>

  <!--  Bootstrap Js -->
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"
                                                                        integrity="sha384-
ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+OMhuP+IlRH9sENBO0LRn5q+8nbTov4+1p"
    crossorigin="anonymous"></script>
</body>

</html>
```

5.1.1.1-App.js

```
 import LoggedInStatusState from '../Context/LoggedInStatus/LoggedInStatusState';
import ApplicationModeState from '../Context/ApplicationMode/ApplicationModeState';
import UserDataState from '../Context/UserData/UserDataState';
import '../Styles/App.css';
import Navbar from './Navbar';
import LeftBar from './LeftBar';
import VideoFeed from './VideoFeed';
import Login from './Login';
import Signup from './Signup';
import Upload from './Upload';
import ForgetPass from './ForgetPass';
import OTP from './OTP';
import VideoWatchSection from './VideoWatchSection';
import ChatInterface from './ChatSection/ChatInterface';
import JoinRoom from './ChatSection/JoinRoom';
import CreateRoom from './ChatSection/CreateRoom';
import Chatting from './ChatSection/Chatting';
import UserProfile from '../Components/UserProfile';
import Bookmark from './Bookmark';
import YourVideoSection from './YourVideoSection';
import History from './History';
import SearchResult from './SearchResult';



import {
  BrowserRouter as Router,
  Switch,
  Route
} from "react-router-dom";


function App() {

  return (
    <>
      {/* <Vedionote/> */}
    <Router>
      <LoggedInStatusState>
        <UserDataState>
          <ApplicationModeState>
            <div className='App'>
              <Navbar />
              <LeftBar />
              <Switch>
                <Route exact path="/videoFeed">
                  <VideoFeed />
                </Route>
                <Route exact path="/">
                  <VideoFeed />
                </Route>
                <Route exact path="/videoWatchSection">
                  <VideoWatchSection />
                </Route>
                <Route exact path="/signup">
                  <Signup />
                </Route>
                <Route exact path="/login">
                  <Login />
                </Route>
                <Route exact path="/upload">
                  <Upload />
                </Route>
                <Route exact path="/forgetPass">
                  <ForgetPass />
                </Route>
                <Route exact path="/otp">
                  <OTP />
                </Route>
                <Route exact path="/chatInterface">
                  <ChatInterface />
                </Route>
                <Route exact path="/joinRoom">
                  <JoinRoom />
                </Route>
                <Route exact path="/createRoom">
                  <CreateRoom />
                </Route>
                <Route exact path="/chatting">
                  <Chatting />
                </Route>
                <Route exact path="/userProfile">
                  <UserProfile />
                </Route>
                <Route exact path="/bookmark">
                  <Bookmark />
                </Route>
                <Route exact path="/yourVideos">
                  <YourVideoSection />
                </Route>
                <Route exact path="/history">
                  <History />
                </Route>
                <Route exact path="/searchResult">
                  <SearchResult />
                </Route>
              </Switch>
            </div>
          </ApplicationModeState>
        </UserDataState>
      </LoggedInStatusState>
    </Router>

   </>
  );
```

```
}
export default App;
```

5.1.2 Video Watch Section

```
import React from 'react';
import { useEffect, useState } from 'react';
import '../Styles/VideoWatchSection.css';
import { useContext } from 'react';
import LoggedInStatusContext from '../Context/LoggedInStatus/LoggedInStatusContext';
import ApplicationModeContext from '../Context/ApplicationMode/ApplicationModeContext';
import UserDataContext from '../Context/UserData/UserDataContext';
import Vedionote from './Vedionote'
import ReminderCard from '../Components/ReminderCard'

export default function VideoWatchSection() {
  const is_loggedin = useContext(LoggedInStatusContext);
  const applicationMode = useContext(ApplicationModeContext);
  const userData = useContext(UserDataContext);
  const [currTimeSec, setCurrTime] = useState(0);
  useEffect(() => {
    if (applicationMode.mode === "light") {
        document.getElementById("Video_metadata").style.color = "black";
    }
    else {
        document.getElementById("Video_metadata").style.color = "white";
    }
    if (localStorage.getItem("userEmail") !== null) {
        is_loggedin.setLoggedin(true);
    }
  })
```

```
const increaseLikeCount = () => {
   likeVideo(userData.currentSno);
}

async function likeVideo(sno) {
   let userObject = {
      "sno": sno,
      "email": localStorage.getItem("userEmail")
   }
   await fetch(`${userData.backendApi}/likeVideo/`, {
      method: "POST",
      headers: {
         'Content-Type': 'application/json',
      },
      body: JSON.stringify(userObject),
   }).then(response => response.json()).then((data) => {
      if (data.status === 200) {
         document.getElementById("likeBtnSvg").style.color = "rgb(19, 124, 237)";
         userData.setCurrentVideoLikes((userData.currentVideoLikes) + 1);
      }
      else {
      }
   })
}

let setReminder = () => {
   console.log("clicked reminder btn");
   document.getElementById("VideoWatchSection").style.display="none";
   document.getElementById("ReminderCard").style.display="block";
}

//Functions for handling hover effect for like, share ...btns
const DisplayLikeText = () => {
    document.getElementById("likeBtnText").style.display = "block";
}
const HideLikeText = () => {
    document.getElementById("likeBtnText").style.display = "none";
}

const DisplayShareText = () => {
    document.getElementById("shareBtnText").style.display = "block";
}
const HideShareText = () => {
    document.getElementById("shareBtnText").style.display = "none";
}

const DisplayWatchLaterText = () => {
    document.getElementById("wlBtnText").style.display = "block";
}
const HideWatchLaterText = () => {
    document.getElementById("wlBtnText").style.display = "none";
}

const DisplayReportText = () => {
    document.getElementById("reportBtnText").style.display = "block";
}
const HideReportText = () => {
    document.getElementById("reportBtnText").style.display = "none";
}
const setTimeHere = () => {
    setCurrTime(document.getElementById('ActualVideo').currentTime)
}
return (
    <>
        <div id="VideoWatchSection" className="VideoWatchSection">
            <div className='VideoPlaySection my-4'>
                <div className="VideoPlayer">
            <div className="Video">
                      <video id="ActualVideo" controlsList="nodownload" src={userData.currentVideoLink} className="VideoTag" type="video/mp4" controls>
              </video>
            </div>

            <div className="VideoMetaData" id="Video_metadata">
              <div className="VideoTitle" >
                {userData.currentVideoTitle}
              </div>

              <div className="LikeShareWatchLaterReportBtns">
                <span style={{ display: "flex", flexDirection: "column" }}>
                  {/* likebutton */}

                  <div id="likeBtnDiv" onClick={increaseLikeCount}>
                    <svg id="likeBtnSvg" xmlns="http://www.w3.org/2000/svg" style={{ height: "35px", width: "35px", margin: "5px 10px" }} onMouseEnter={DisplayLikeText} onMouseLeave={HideLikeText} fill="currentColor" className=" VideoMetaBtn bi bi-hand-thumbs-up-fill" viewBox="0 0 16 16" >
                      <path d="M6.956 1.745C7.021.81 7.908.087 8.864.325l.261.066c.463.116.874.456 1.012.965.22.816.533 2.511.062 4.51a9.84 9.84 0 0 1 .443-.051c.713-.065 1.669-.072 2.516.21.518.173.994.681 1.2 1.273.184.532.16 1.162-.234 1.733.058.119.103.242.138.363.077.27.113.567.113.856 0 .289-.036.586-.113.856-.039.135-.09.273-.16.404.169.387.107.819-.003 1.148a3.163 3.163 0 0 1-.488.901c.054.152.076.312.076.465 0 .305-.089.625-.253.912C13.1 15.522 12.437 16 11.5 16H8c-.605 0-1.07-.081-1.466-.218a4.82 4.82 0 0 1-.97-.484l-.048-.03c-.504-.307-.999-.609-2.068-.722C2.682 14.464 2 13.846 2 13V9c0-.85.685-1.432 1.357-1.615.849-.232 1.574-.787 2.132-1.41.56-.627.914-1.28 1.039-1.639.199-.575.356-1.539.428-2.59z" />
                    </svg>
                  </div>
                  <p className="likeCountText" id="likeBtnText">{userData.currentVideoLikes} Likes</p>
                </span>

                <span style={{ display: "flex", flexDirection: "column" }}>
                  {/* sharebutton */}
                  <svg xmlns="http://www.w3.org/2000/svg" onMouseEnter={DisplayShareText} onMouseLeave={HideShareText} style={{ height: "35px",
width: "35px", margin: "5px 10px" }} fill="currentColor" className="VideoMetaBtn bi bi-send-
fill" viewBox="0 0 16 16">
                                    <path d="M15.964.686a.5.5 0 0 0-.65-.65L.767 5.855H.766l-.452.18a.5.5
0 0 0-.082.887l.41.26.001.002 4.995 3.178 3.178 4.995.002.002.26.41a.5.5 0 0 0 .886-.083l6-15Zm-
1.833 1.89L6.637 10.07l-.215-.338a.5.5 0 0 0-.154-.154l-.338-.215 7.494-7.494 1.178-.471-.47
1.178Z" />
                              </svg>
                              <p id="shareBtnText">Share</p></span>
                          <span onClick={setReminder} style={{ display: "flex", flexDirection: "column"
}}>
                                                                        <svg xmlns="http://www.w3.org/2000/svg"
onMouseEnter={DisplayWatchLaterText} onMouseLeave={HideWatchLaterText} style={{
height: "35px", width: "35px", margin: "5px 10px" }} fill="currentColor"
className="VideoMetaBtn bi bi-watch" viewBox="0 0 16 16">
                                      <path d="M8.5 5a.5.5 0 0 0-1 0v2.5H6a.5.5 0 0 0 0 1h2a.5.5 0 0 0 .5-
.5V5z" />
                                    <path d="M5.667 16C4.747 16 4 15.254 4 14.333v-1.86A5.985 5.985 0 0
1 2 8c0-1.777.772-3.374 2-4.472V1.667C4 .747 4.746 0 5.667 0h4.666C11.253 0 12 .746 12
1.667v1.86a5.99 5.99 0 0 1 1.918 3.48.502.502 0 0 1 .582.493v1a.5.5 0 0 1-.582.493A5.99 5.99 0
0 1 12 12.473v1.86c0 .92-.746 1.667-1.667 1.667H5.667zM13 8A5 5 0 1 0 3 8a5 5 0 0 0 10 0z" />
                              </svg>
                              <p id="wlBtnText">Remind me</p></span>
                           <span style={{ display: "flex", flexDirection: "column" }}>
                              {/* reportbutton */}
                                                                        <svg xmlns="http://www.w3.org/2000/svg"
onMouseEnter={DisplayReportText} onMouseLeave={HideReportText} style={{ height: "35px",
width: "35px", margin: "5px 10px" }} fill="currentColor" className="VideoMetaBtn bi bi-shield-
fill-exclamation" viewBox="0 0 16 16">
                                      <path fillRule="evenodd" d="M8 0c-.69 0-1.843.265-2.928.56-1.11.3-
2.229.655-2.887.87a1.54 1.54 0 0 0-1.044 1.262c-.596 4.477.787 7.795 2.465 9.99a11.777 11.777
0 0 0 2.517 2.453c.386.273.744.482 1.048.625.28.132.581.24.829.24s.548-.108.829-.24a7.159
7.159 0 0 0 1.048-.625 11.775 11.775 0 0 0 2.517-2.453c1.678-2.195 3.061-5.513 2.465-9.99a1.541
1.541 0 0 0-1.044-1.263 62.467 62.467 0 0 0-2.887-.87C9.843.266 8.69 0 8 0zm-.55 8.502L7.1
4.995a.905.905 0 1 1 1.8 0l-.35 3.507a.552.552 0 0 1-1.1 0zM8.002 12a1 1 0 1 1 0-2 1 1 0 0 1 0 2z"
/>
                              </svg>
                              <p id="reportBtnText">Report</p></span>
                         </div>
                      </div>


                      <div className="accordion" id="accordionExample" >
                         <div className="accordion-item">
```

```
                <h2 className="accordion-header">
                  <button className="accordion-button ChannelInfo" type="button" data-bs-toggle="collapse" data-bs-target="#collapseOne" aria-expanded="true" aria-controls="collapseOne">
                    <div className="ChannelPic"><img src={userData.currentVideoChannelPhoto} style={{ borderRadius: "50%" }} height="50px" width="50px" alt=".." /></div>
                    <div className="ChannelName">{userData.currentVideoChannelName}</div>
                  </button>
                </h2>
                <div id="collapseOne" className="accordion-collapse collapse" aria-labelledby="headingOne" data-bs-parent="#accordionExample">
                  <div className="accordion-body">
                    <strong>Description : </strong>
                    <br />
                    {userData.currentVideoDesc}
                    <br />
                    <br />
                    <strong> <a target="blank" href={userData.currentVideoNotes}>Access Notes File</a></strong>
                  </div>
                </div>
              </div>
            </div>

          </div>

          <div className="my-2 QuizSection">
            <h3 style={{ padding: "8px" }} >Add Your Notes Here : </h3>
            <div style={{ display: "flex", justifyContent: "center" }}>
              <Vedionote cuTime={currTimeSec} setTime={setTimeHere} />
            </div>
          </div>

        </div>
      </div>
```

```
            <div className='d-flex justify-content-center my-4'>
            <ReminderCard/>
            </div>
        </>
    )
}
```

5.1.3 ViewVideo.py

```
@csrf_exempt  # to avoid csrf forbiden verification error
def viewVideo(request):
    if request.method == "POST":

        json_data = request.body
        stream = io.BytesIO(json_data)
        parsed_data = JSONParser().parse(stream)
        video_id = parsed_data.get('sno')
        email = parsed_data.get('email')

        userObject = User.objects.get(email=email)
        LoginStatusObject = LoginStatus.objects.get(user=userObject)

        if(LoginStatusObject.is_loggedin == True):  # verifying if user is logged in

            viewedVideoObject = VideoData.objects.get(sno=video_id)
            viewedVideoObject.video_views += 1
            viewedVideoObject.save()

            responseObject = {
                "status": 200,
                "response": "Video view count increased Successfully"
            }

            json_data = JSONRenderer().render(responseObject)
            return HttpResponse(json_data, content_type='application/json')

        responseObject = {
            "status": 404,
            "response": "You're not logged in"
        }

        json_data = JSONRenderer().render(responseObject)
        return HttpResponse(json_data, content_type='application/json')

    responseObject = {
        "status": 404,
        "response": "POST request was expected"
    }

    json_data = JSONRenderer().render(responseObject)
   return HttpResponse(json_data, content_type='application/json')
```

5.1.4 Login.js

```
import React from 'react'
import '../Styles/Login.css';
import { useEffect } from 'react';
import { useContext } from 'react';
import { useState } from 'react';
// import { Link } from 'react-router-dom';

import Spinner from './Spinner';
// contexts
```

```
import LoggedInStatusContext from '../Context/LoggedInStatus/LoggedInStatusContext';
import UserDataContext from '../Context/UserData/UserDataContext';


export default function Login() {
    const is_loggedin = useContext(LoggedInStatusContext);
    const userData = useContext(UserDataContext);

    // FOR SHOWING SPINNER
    const [loading, setLoading] = useState(false)

    const handleUserLogin = (e) => {
        e.preventDefault();
        if (document.getElementById('loginUsername').value.length === 0) {
            alert('Username Cannot be empty');
        }
        else if (document.getElementById('loginEmail').value.length === 0) {
            alert('email field Cannot be empty');
        }
        else if (document.getElementById('loginPassword').value.length === 0) {
            alert('password field Cannot be empty');
        }
        else {
            loginUser();
        }
    }

    async function loginUser() {
        userData.setUserEmail(document.getElementById("loginEmail").value);
        let userObject = {
            "username": document.getElementById("loginUsername").value,
            "password": document.getElementById("loginPassword").value,
            "email": document.getElementById("loginEmail").value
        }
        setLoading(true);
        await fetch(`${userData.backendApi}/loginUser/`, {
            method: "POST",
            headers: {
                'Content-Type': 'application/json',
            },
            body: JSON.stringify(userObject),
        }).then(response => response.json()).then((data) => {
            setLoading(false);

            if (data.status === 200) {
                is_loggedin.setLoggedin(true);
                localStorage.setItem("userEmail", userObject.email);
                localStorage.setItem("username", userObject.username);
                document.getElementById("Application-logo").click();
                document.getElementById("GreetingAlert").style.display = "block";
            }
            else {
                alert(data.response);
            }
        });
    }

    useEffect(() => {
        let windowWidth = window.matchMedia("(min-width: 1077px)");
        if (windowWidth.matches) {
            document.getElementById("Login").className = "w-50 shadow p-3 mb-5 bg-body mx-auto
my-5 ";
        }
        else {
            document.getElementById("Login").className = "w-75 shadow p-3 mb-5 bg-body mx-auto
my-5 t";
        }
    },[]);
```

```
  return (
    <>
      {loading && <Spinner />}
      {!loading &&
        <div>
          <div id="logoutAlert" style={{ display: "none" }} className="alert alert-info alert-dismissible fade show" role="alert">
            Logged out Successfully :)
              <button type="button" className="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>
          </div>

          <div id="Login" className="w-50 shadow p-3 mb-5 bg-body mx-auto my-5 " style={{ height: "30rem", display: "flex" }}>
            <img id="loginImage" src="Images/Login.gif" width="50%" alt="" />
            <div className='container mx-3 my-3'>
              <div className='fs-5 text-center'>Login</div>
              <hr />
              <form action="" onSubmit={handleUserLogin} >
                <div className="mb-2">
                    <label htmlFor="exampleFormControlInput1" className="form-label" id="label">Username<span className="mandatory_sign">*</span></label>
                  <input type="text" id="loginUsername" className="form-control input_tag" />
                </div>
                <div className="mb-2">
                    <label htmlFor="exampleFormControlInput1" className="form-label" id="label">Email<span className="mandatory_sign">*</span></label>
                  <input type="email" id="loginEmail" className="form-control input_tag" />
                </div>
                <div className="mb-2">
                    <label htmlFor="exampleFormControlInput1" className="form-label " id="label">Password<span className="mandatory_sign">*</span></label>
                    <input type="password" id="loginPassword" className="form-control input_tag" />
                </div>
```

```
                        <div className='text-center'>
                                <button type="submit" className=" loginBtn btn btn-info my-2 mx-2
custom_btn" style={{ "alignSelf": "center" }}>Login</button>
                        </div>
                    </form>
                </div>
            </div>
        </div>}
    </>

  )
}
```

5.1.5 Login.py

```
@csrf_exempt  # to avoid csrf forbiden verification error
def loginUser(request):
    if request.method == "POST":
        json_data = request.body
        stream = io.BytesIO(json_data)
        parsed_data = JSONParser().parse(stream)
        userName = parsed_data.get('username')
        password = parsed_data.get('password')
        emailFromFrontend = parsed_data.get('email')

        # Authenticating user
        user = authenticate(request, username=userName, password=password)

        if user is not None:
            # for successfull login
            userObject = User.objects.get(username=user)
            # verifying email
            emailFromBackend = User.objects.get(username=userName).email
            if(emailFromFrontend != emailFromBackend):
                responseObject = {
                    "status": 404,
                    "response": "email not recognized !"
                }
                json_data = JSONRenderer().render(responseObject)
                # returning response
                return HttpResponse(json_data, content_type='application/json')

            # for logging user in
            LoginStatusValue = LoginStatus.objects.get(user=userObject)
            LoginStatusValue.is_loggedin = True
            LoginStatusValue.save()

            responseObject = {
                "status": 200,
                "response": userName+", your login was successfull"
            }
            json_data = JSONRenderer().render(responseObject)
            # returning response
            return HttpResponse(json_data, content_type='application/json')
        else:

            responseObject = {
                "status": 404,
                "response": "Login Failed ! Bad Credentials"
            }

            json_data = JSONRenderer().render(responseObject)

            # Getting email from database , for sending security alert for bad credentials !
            userEmail = User.objects.get(username=userName).email
```

```
            email_mesg = "Security Alert !"+"\n\n"+"Dear "+userName + \
                ", someone just tried logging in your account with bad/wrong credentials! We hope that it
was you."

            send_mail(
                'Team LearnoScope - OTP FOR EMAIL VERIFICATION',
                email_mesg,
                'developerus.community@gmail.com',
                [userEmail],
                fail_silently=False,
            )

            # returning response
            return HttpResponse(json_data, content_type='application/json')

    return HttpResponse("Error : POST request Needed")
```

5.1.6 register.py

```
@csrf_exempt  # to avoid csrf forbideen verification error @
def registerUser(request):
    if request.method == "POST":
        # assuming that , this data is already verified
        userName = request.POST.get('username')
        phone = request.POST.get('phone')
        email = request.POST.get('email')
        password = request.POST.get('password')
        firstName = request.POST.get('firstName')
        lastName = request.POST.get('lastName')
        profile_pic = request.FILES['profile_pic']  # profile_picture
    # if user already exists return - "user already exists"
    # else create the account
     try:
         userObject = User.objects.get(email=email)

         responseObject = {
             "status": 404,
             "response": "The Email is already registered ! Please try logging in"
         }
         json_data = JSONRenderer().render(responseObject)
         return HttpResponse(json_data, content_type='application/json')
     except ObjectDoesNotExist:
         try:
             # userObject = User.objects.create_user(username,email,password)
             userObject = User.objects.create_user(
                 userName, email, password)
             userObject.first_name = firstName
             userObject.last_name = lastName

             # saving profile_photo to UserProfilePhoto Table
             profilePhoto = UserProfilePhoto(
                 profile_pic=profile_pic, user=userObject)
             profilePhoto.save()

             # saving phone number to PhoneNumber
             PhoneNumberObj = PhoneNumber(phone=phone, user=userObject)
             PhoneNumberObj.save()

             # Saving the User Object
             userObject.save()

             # For Sending OTP for Email Verification
             # it will generate random number of length-5
             random_num = random.randint(10000, 99000)
```

```
        otp = random_num

        # saving otp to otp Table
        otp_entry = OTP(otp=otp, user=userObject)
        otp_entry.save()

        # sending otp via mail
        otp_mesg = 'Dear '+userName + \
            ', your One Time Password for verifying the email ' + \
            email+', is '+str(otp)

        send_mail(
            'Team LearnoScope - OTP FOR EMAIL VERIFICATION',
            otp_mesg,
            'developerus.community@gmail.com',
            [email],
            fail_silently=False,
        )

        # responseObject = {
        #     "status": 200,
        #     "response": "User Registered Successfully<br><a href=`https://facebook.com`>Verify
Email</a>"
        # }
        #IF USER SUCCESSFULLY GETS REGISTERED, HE/SHE IS REDIRECTED TO OTP
PAGE
        # json_data = JSONRenderer().render(responseObject)
        # return HttpResponse(json_data, content_type='application/json')
        return redirect("http://localhost:3000/otp")


    except IntegrityError:
        responseObject = {
            "status": 404,
            "response": "username :  '"+userName+"'  is already taken. " + "please try another one"
        }
```

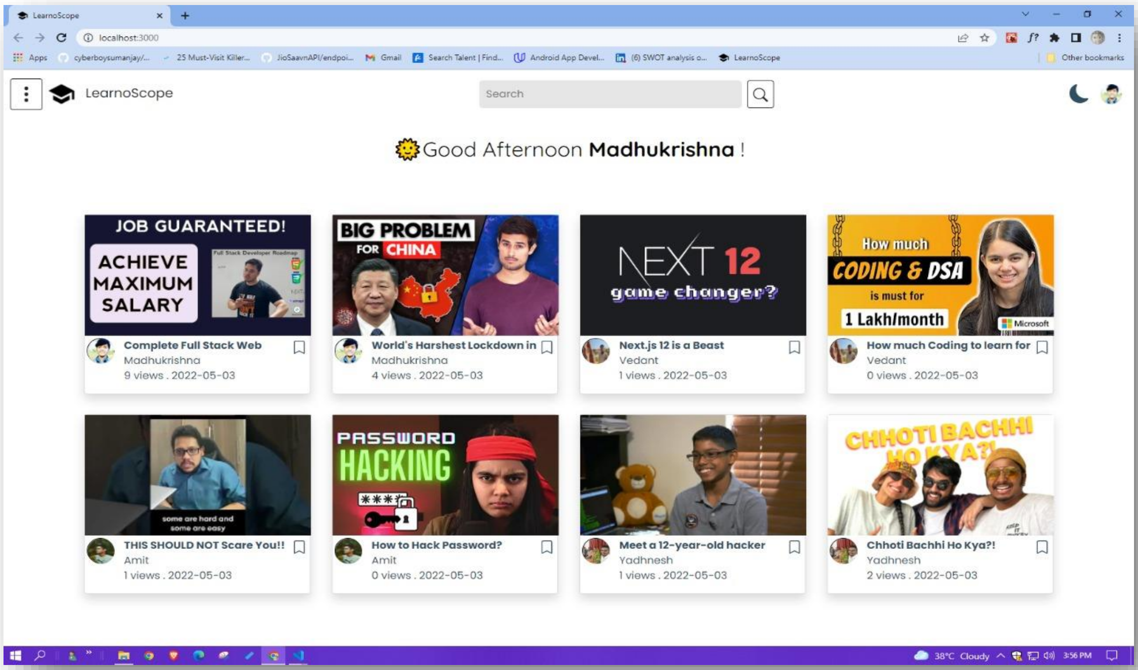
```
            json_data = JSONRenderer().render(responseObject)
            return HttpResponse(json_data, content_type='application/json')

    responseObject = {
        "status": 404,
        "response": "Error: Post request needed"
    }
    json_data = JSONRenderer().render(responseObject)
    return HttpResponse(json_data, content_type='application/json')

# For Email-OTP Verification @
```

5.1.7 logout.py

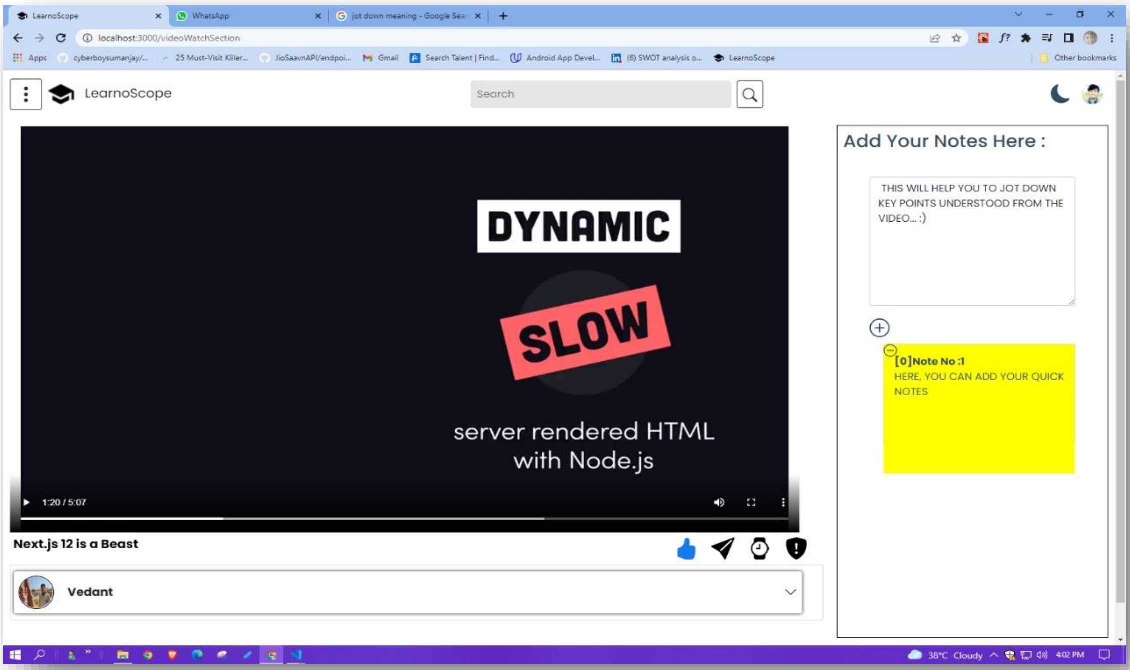
```
def logoutUser(request):
    if request.method == "POST":
        json_data = request.body
        stream = io.BytesIO(json_data)
        parsed_data = JSONParser().parse(stream)
        email = parsed_data.get('email')

        userObject = User.objects.get(email=email)
        # for logging user out
        LoginStatusObject = LoginStatus.objects.get(user=userObject)
        LoginStatusObject.is_loggedin = False
        LoginStatusObject.save()

        responseObject = {
            "status": 200,
            "response": "Logged out Successfully"
        }
        json_data = JSONRenderer().render(responseObject)
        return HttpResponse(json_data, content_type='application/json')


    responseObject = {
        "status": 404,
        "response": "POST request needed"
    }
    json_data = JSONRenderer().render(responseObject)
    return HttpResponse(json_data, content_type='application/json')
```

5.2 Output: -

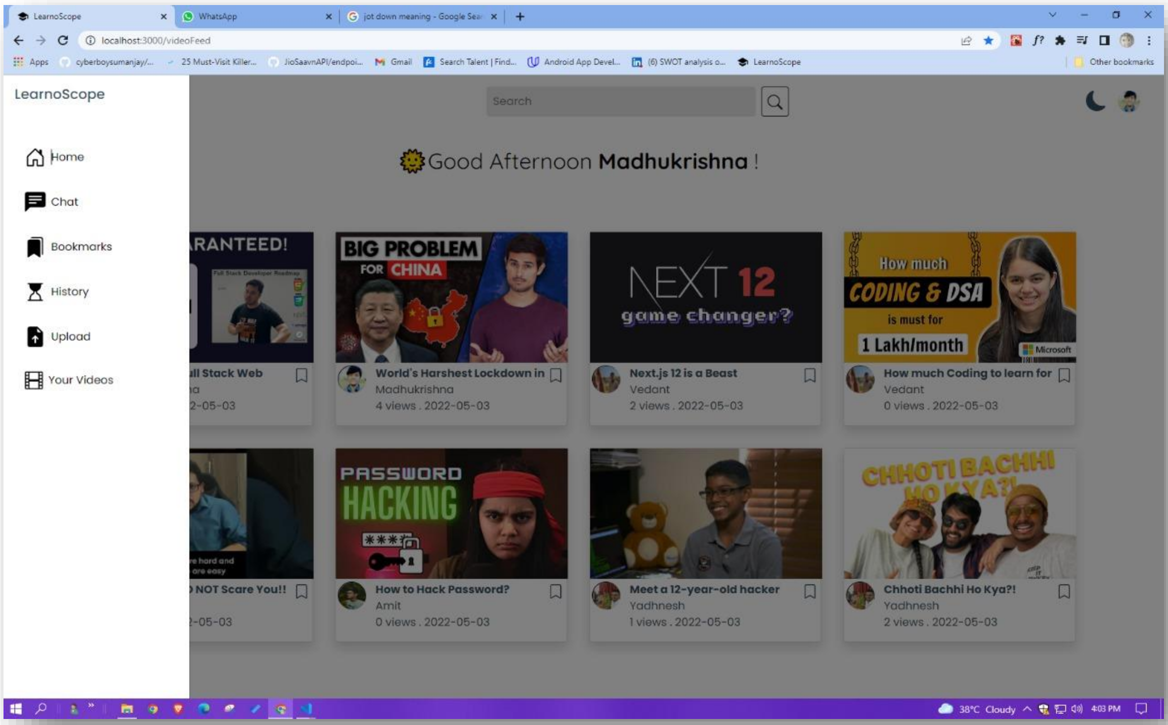
Video Feed Page After Login: -



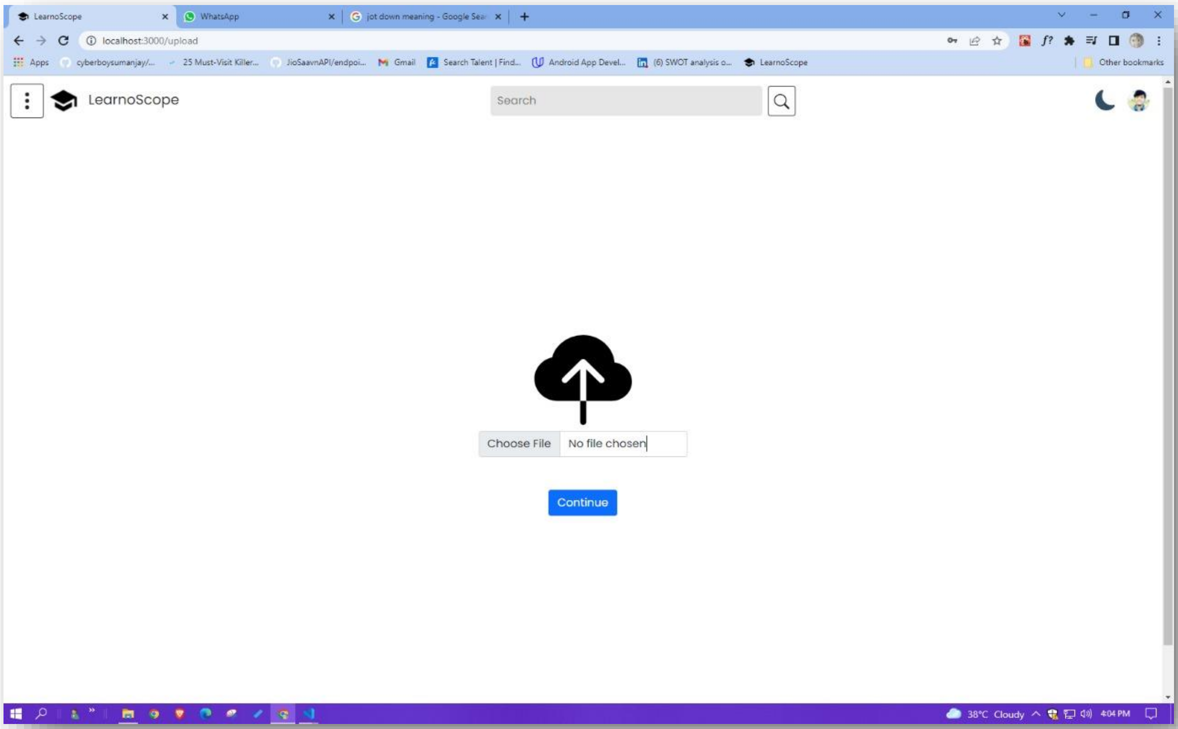
Video Watch Section: -


LearnoScope
Search
Add Your Notes Here :
THIS WILL HELP YOU TO JOT DOWN KEY POINTS UNDERSTOOD FROM THE VIDEO... :)
[0]Note No :1
HERE, YOU CAN ADD YOUR QUICK NOTES
DYNAMIC
SLOW
server rendered HTML with Node.js
1:20 / 5:07
Next.js 12 is a Beast
Vedant

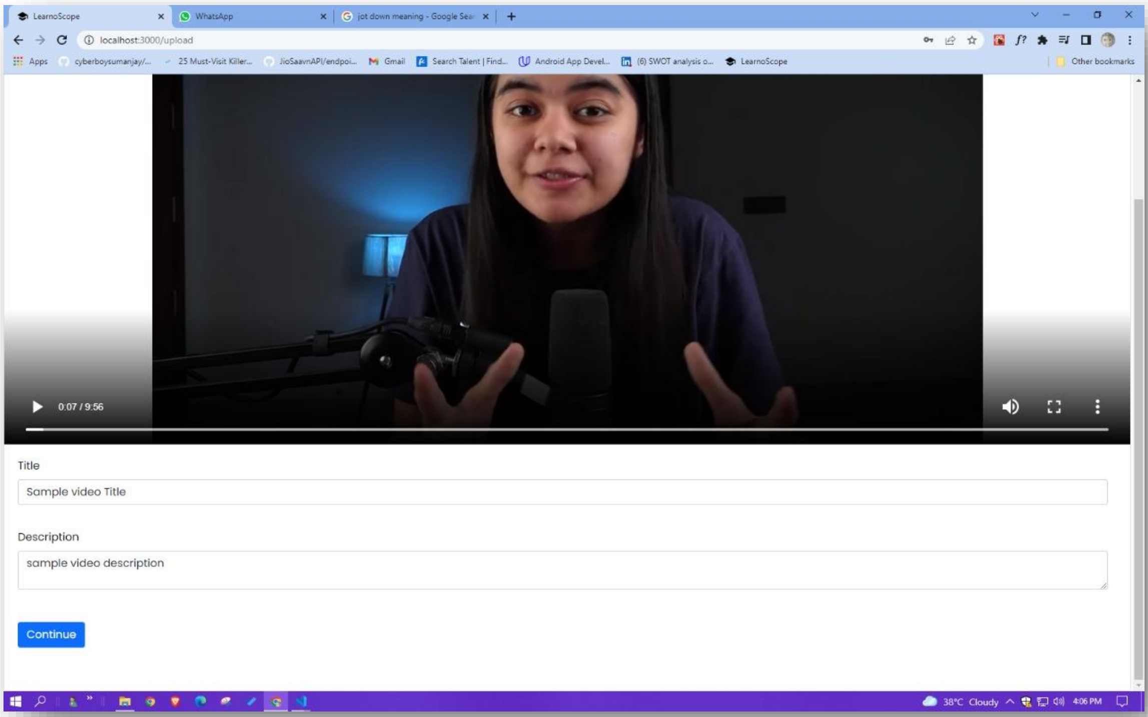
Demonstration of LeftBar :-



Upload Section: -



Filling of MetaData while Uploading Video :-



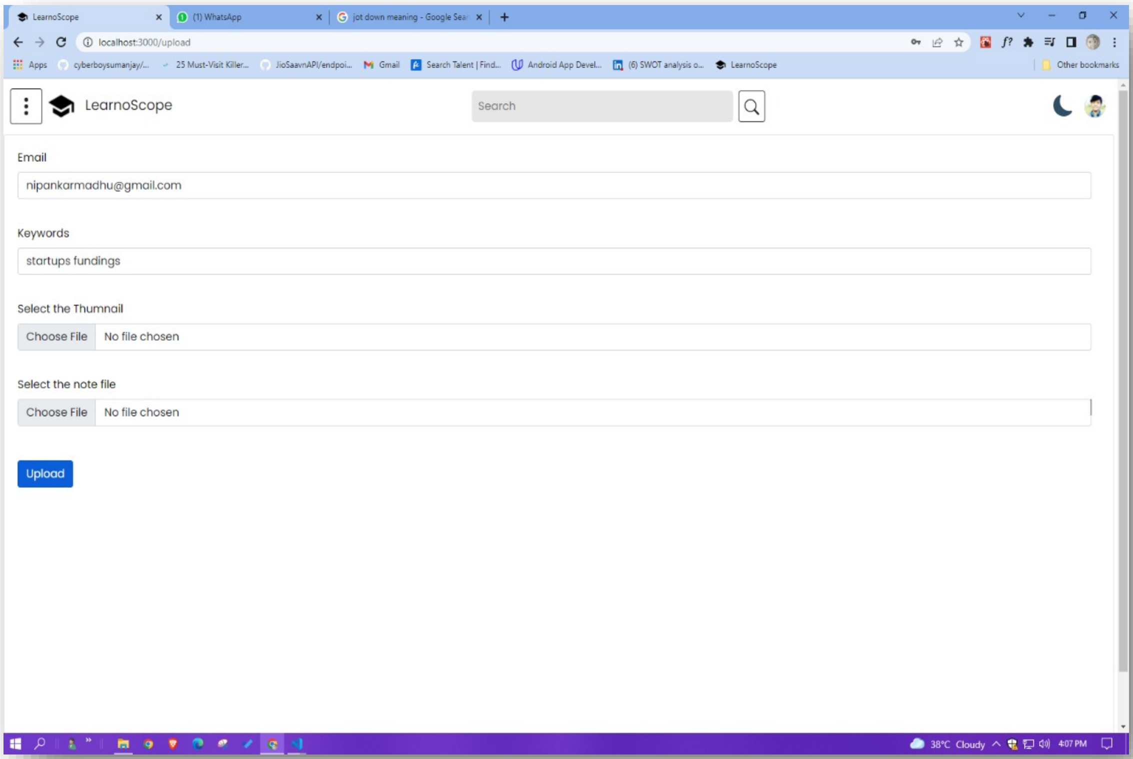
Title

Sample video Title

Description

sample video description

Continue



Email

nipankarmadhu@gmail.com

Keywords

startups fundings

Select the Thumnail

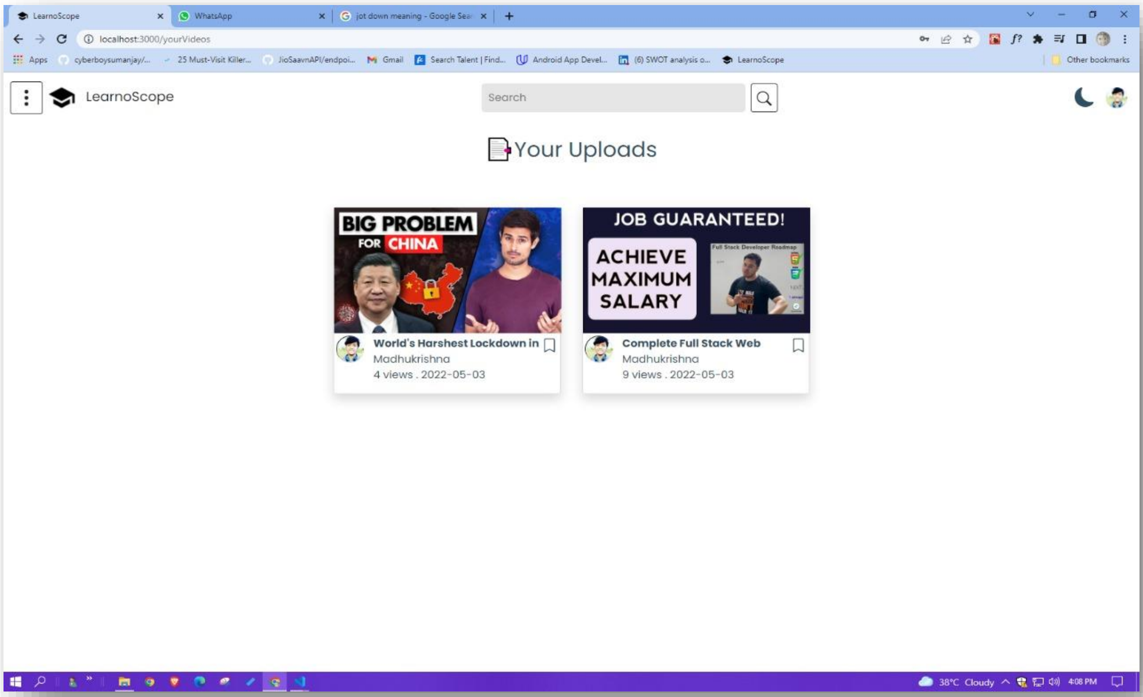
Choose File No file chosen

Select the note file

Choose File No file chosen

Upload

Uploads of a particular user: -

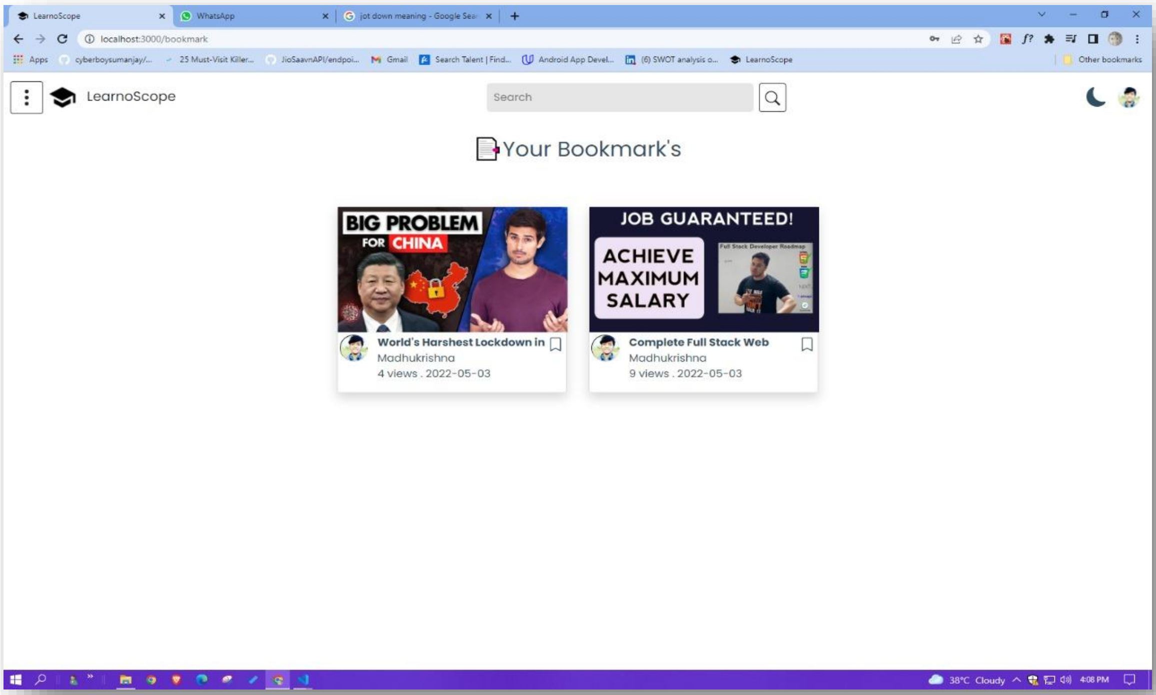


localhost:3000/yourVideos

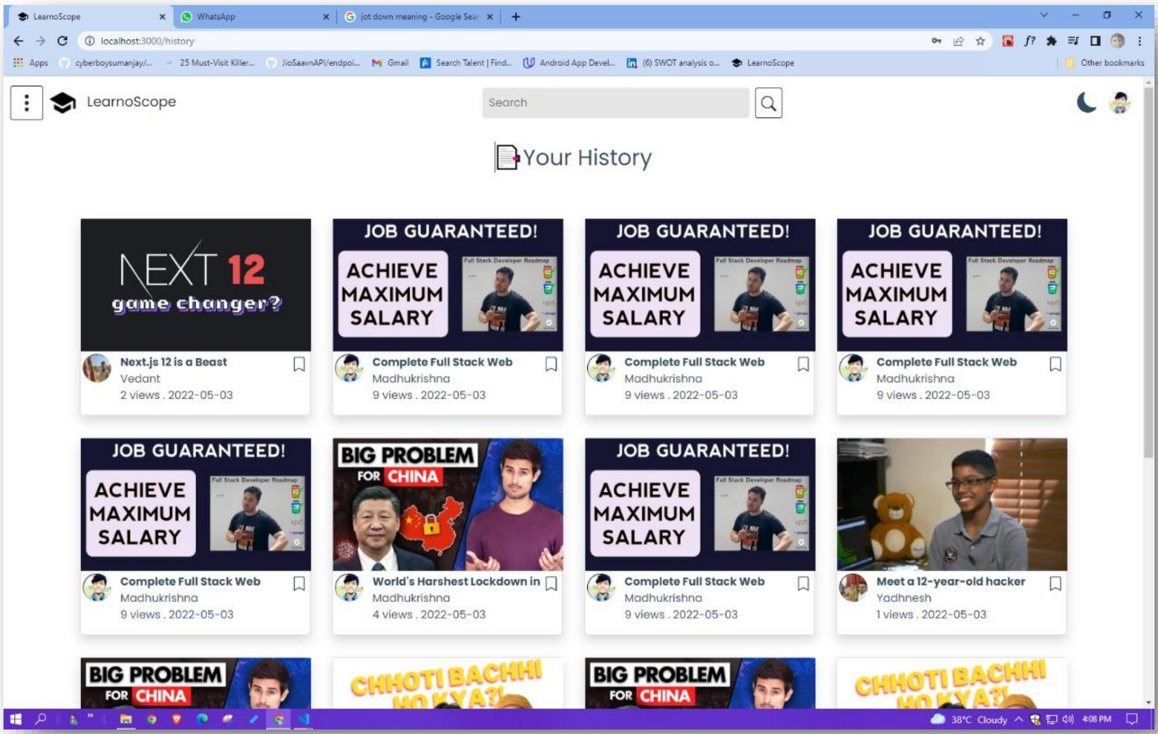
World's Harshest Lockdown in
Madhukrishna
4 views . 2022-05-03

Complete Full Stack Web
Madhukrishna
9 views . 2022-05-03

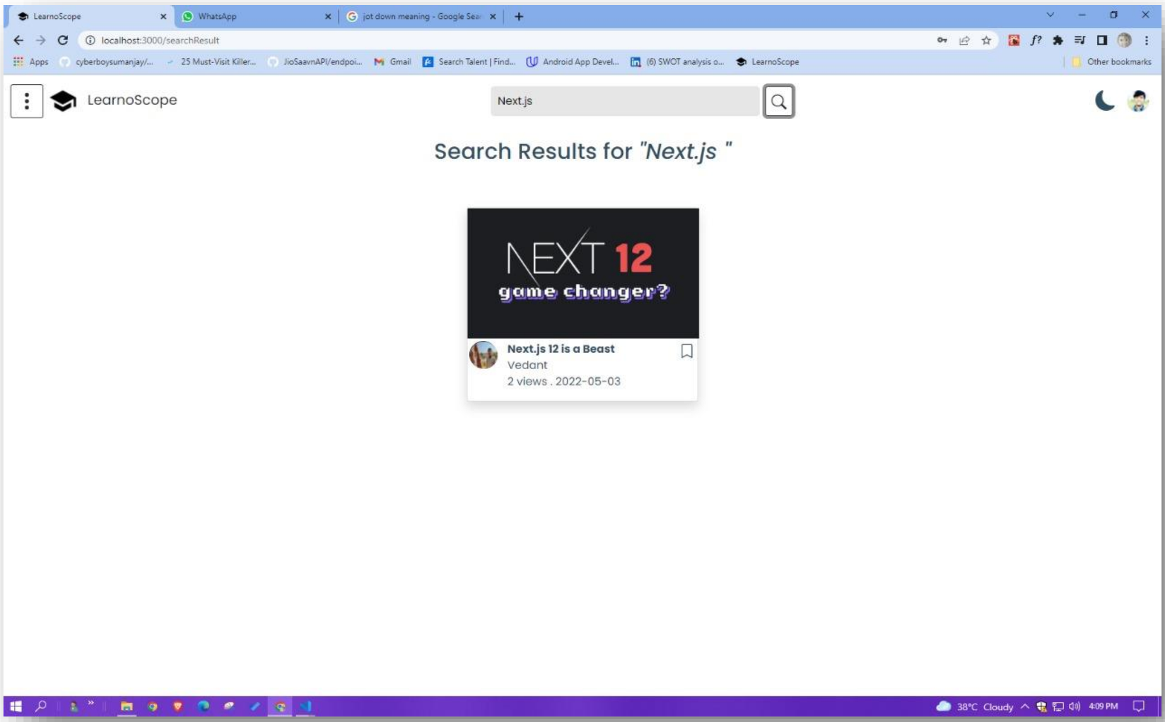
Bookmarked Videos are seen here: -



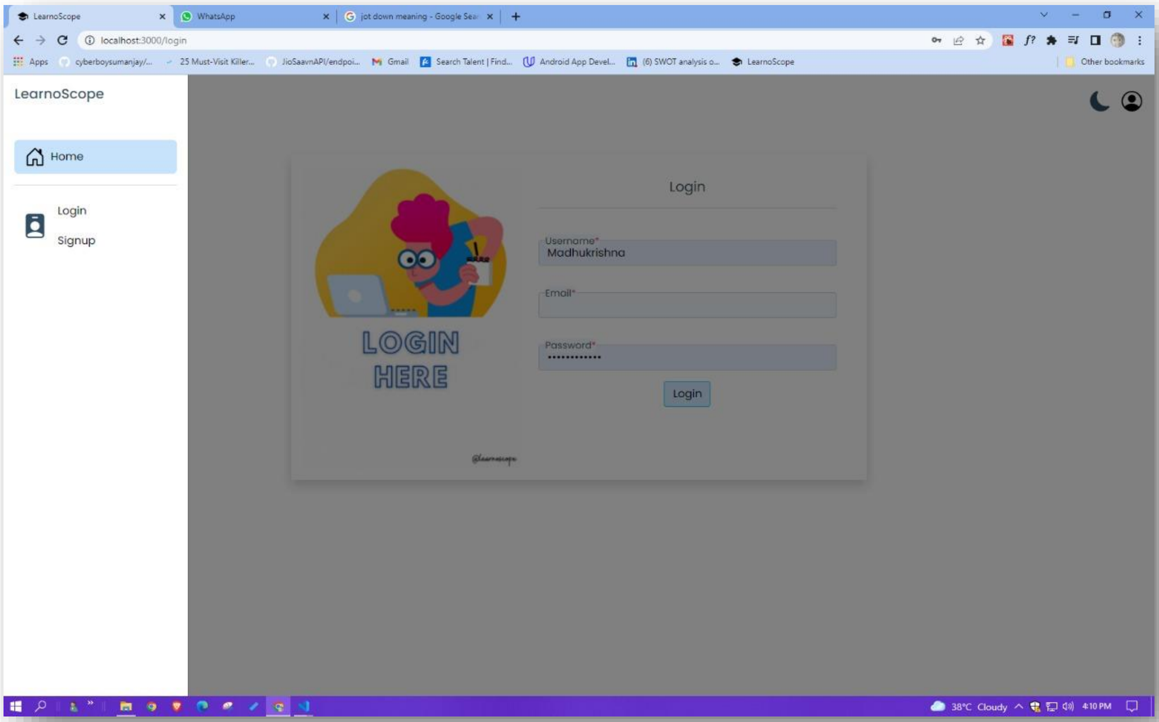
User History Page: -



Search Results are shown here: -



State of application when user is not logged in: -



Chapter: -6 TESTING

No.	Testcase Name	Description	Expected Result	Actual Result	Remark
1	Login Authentication	Entering wrong email and password	Should show bad credentials alert/message	A Bad Credential Alert is seen	PASS
2	Upload section Video Input	Click on Choose File Button	Only .mp4 files must be visible to select	Only .mp4 files are visible to select	PASS
3	History generation	Click on any video card	The clicked/opened video must be added in history section	The clicked/opened video gets added in history section	PASS
4	View Increment	Click on any video card	The view count of the clicked/opened video must be incremented by one	The view count of the clicked/opened video is getting incremented by one	PASS
5	Like Increment	Click on like button in VideoWatchSection	The like of the current video must be incremented by one	The like of the current video is getting incremented by one	PASS
6	Access Notes File	Click on “Access Notes File” in video description	A Notes file must get open in new tab	A Notes file is getting open in new tab	PASS
7	Mode Switching	Click on Moon Shaped button at the Navbar	Theme must change to Dark	Theme got changed to Dark	PASS
8	Logout	Click on profile photo & then click on Logout	LeftBar options must change & user must be redirected to login page	LeftBar options got changed & user got redirected to login page	PASS

Chapter: -7 COST ESTIMATION

7.1 Effort Estimation Table

Task	**Effort Weeks**	**Deliverables**	**Milestones**
Analysis of existing system and comparing with the proposed one	**1.3 weeks**		
Planning and Design 1. System Flow 2. Design Modules and its Deliverables	**1 week**	Modules,Design document	
Implementation	**5 weeks**	Primary System	
Testing	**1 week**	Test Reports	Formal
Documentation	**0.3 week**	Complete Project Report	Formal

7.2 Estimation of KLOC

Sr.No.	**Modules**	**KLOC**
1.	Web Designing	0.90
2.	Routing Code	0.10
3.	Client-Side Logic	2.59
4.	State Management Code	0.10
5.	Server-Side Logic	1.43
6.	Database Schema	0.10

Thus, the Total number of lines required is approximately 5.22 KLOC

Chapter: -8 Future Scope

Our Project technologies can be extended in the following future scope.

- We can include a feature of recognizing the content of the video using the machine learning algorithm so that we can provide complete educational video content to the audience.
- GUI can be enhanced for greater need.
- Feature of reminding users to watch videos on their scheduled time.
- Multiple technologies other than one language can be incorporated.
- Feedback Poles
- Community Posts
- Comment Feature
- Mobile application versions can be launched.

Chapter: -9 CONCLUSION

This project is taking into consideration various techniques that can be used to make a successful “Educational Video Sharing Platform” for the web.

We also learned the languages and technologies used to obtain the goal of our project. We have successfully designed the application using the technologies such as

-React JS (Frontend)

-Django (Backend)

We are making the use of different techniques and try to minimize the distraction so that users can experience the distraction free Educational Video sharing platform. Also in this project we are trying to include the important features that make the learning journey of the user easy and enjoyable.

So here we conclude that our project will help us to achieve the s

Chapter :- 10 REFERENCE

1. https://reactjs.org/docs/getting-started.html
2. https://docs.djangoproject.com/en/4.0/
3. https://stackoverflow.com/
4. https://www.javatpoint.com/react-props
5. https://www.geeksforgeeks.org/reactjs-state-react/
6. https://www.youtube.com/watch?v=tiLWCNFzThE
7. https://getbootstrap.com/
8. https://www.sqlite.org/docs.html
9. https://codewithharry.com/videos/python-django-tutorials-hindi-0
10. https://www.netlify.com/