Acknowledgement

Knowledge in itself is a continuous process. At this moment of our substantial enhancement, we rarely find words to express our gratitude towards those who were constantly involved with us. We take this opportunity to express our profound gratitude and deep regards to our guide Mr. Kamlesh Makvana for his exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by him time to time shall carry us a long way in the journey of life on which we are about to embark. The completion of any inter disciplinary project depends upon coordination, cooperation and combined efforts of several resources of knowledge, creativity, skill, energy and time. The work being accomplished now, we feel our most sincere urge to recall and knowledge through these lines, trying our best to give full credit wherever it deserves. It's our good fortune that we had support and well wishes of many. We are thankful to all and those names which have been forgotten to acknowledge here but contributions have not gone unnoticed.

"We may not achieve everything we dream, but we cannot achieve anything unless we dream."

With Sincere regards,
Hetul Sheth(17IT107)

Abstract

Serverless was the hottest cloud topic in 2017, and will continue to be in 2018. Some companies are opting to skip using cloud services like EC2, and moving straight to a completely serverless architecture. It pays to understand serverless design patterns.

So in this project we are creating a completely sereverless website using AWS, Auth0, Firebase and some basic coding in Node js which will allow user to watch videos uploaded on this platform and if a user wants to upload his/her own video need to login to authenticate the identity.

This is completely serverless so the aim for creating this is less latency, high throughput and cost Optimization.

List of tables

List of figures

CHAPTER 1

INTRODUCTION

1.1 Project Overview

So in this project we are creating a completely sereverless website using AWS, Auth0, Firebase and some basic coding in Node js which will allow user to watch videos uploaded on this platform and if a user wants to upload his/her own video need to login to authenticate the identity.

.

1.1.1 How to do

The project lies on the foundation of Cloud architecture for here used AWS. AWS stands for Amazon Web Services.

Auth0 authenticate the users by creating a temperory creadentials which is checked by AWS through API Gateway every time a existing user sign in or a new user sign up .

1.2 Scope

What it can do

- Provides user login
- Allow authenticate user to upload video files(recommended 1080p).
- Transcode the video uploaded to 720p, 360p, WebRip.
- Allow any user who visited the website to watch any video without signing in

What it can't do

- Upload a MP3 file or Image file.
- Profile page of user.
- Show trending videos or most seen videos or have a like or comment section(like youtube)

1.3 Objective

To create a completely Serverless website which does not require to instatiate the physical or cloud servers This will mainly lead to cost Optimization as servers will be running only the time user request to visit the site and will provide high throughput and less latency and we used here AWS Lambda which will build the infrastructure for the site on its own.

CHAPTER 2

SYSTEM ANALYSIS

2.1 User Characteristics

- Every system has their own characteristics but all gives different functionality from each other.
- Some web application required login or signup with authentication or email verifaction but this website require just login or registration process and no email verification till now because we want to keep it as simple as possible.
- User can just need to sign up nothing else and they easily get to watch any provided resolution of video and upload their own videos too.

2.2 Tools and Technology

1. **AWS Account.**
2. **Auth0 account for verfifcation of user entering correct and existing email address and password and for generating temperory token for user login**
3. **AWS Video Transcoder.**
4. **AWS Lambda.**
5. **AWS API Gateway.**
6. **Google Cloud for Firbase purpose only.**

2.2.1 Software requirement

1. Sublime or any text editor for coding in node js.

Programming Language

1. Node JS

About AWS Tools:

- **AWS Lambda**

  AWS Lambda lets you run code without provisioning or managing servers. You pay only for the compute time you consume - there is no charge when your code is not running.

  With Lambda, you can run code for virtually any type of application or backend service - all with zero administration. Just upload your code and Lambda takes care of everything required to run and scale your code with high availability. You can set up your code to automatically trigger from other AWS services or call it directly from any web or mobile app.

- **AWS Elastic Transcoder**

  Amazon Elastic Transcoder is media transcoding in the cloud. It is designed to be a highly scalable, easy to use and a cost effective way for developers and businesses to convert (or “transcode”) media files from their source format into versions that will playback on devices like smartphones, tablets and PCs.

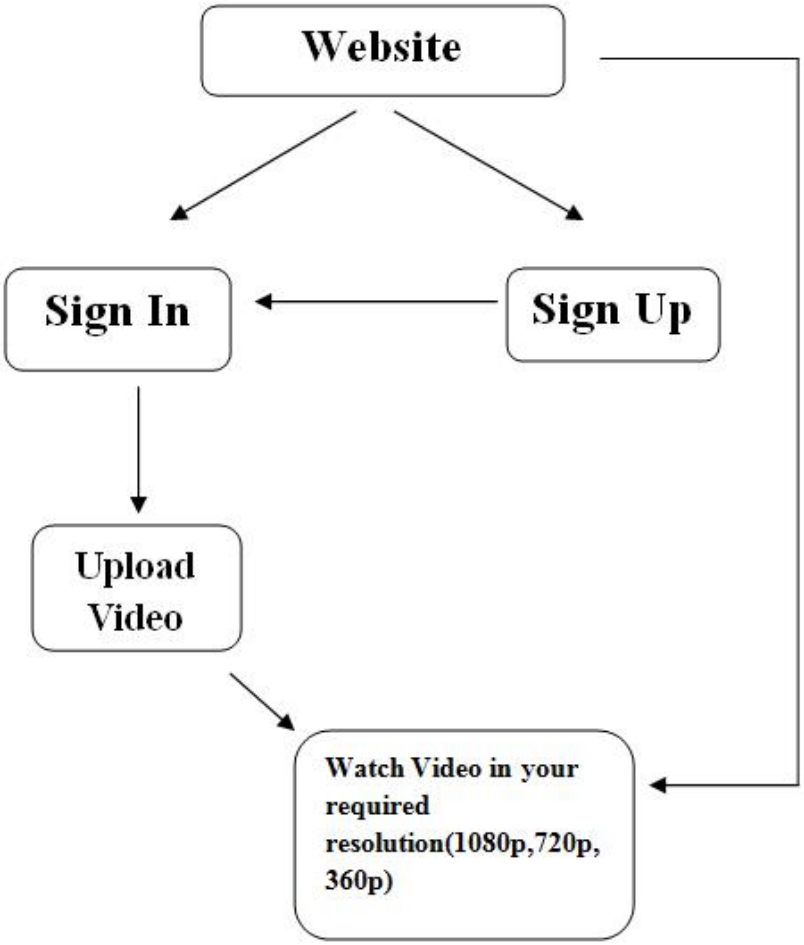
What is Auth0?

Auth0 provides authentication and authorization as a service. They give developers and companies the building blocks they need to secure their applications without having to become security experts.

CHAPTER 3

SYSTEM DESIGN

3.1 Data flow diagram:



3.2 Data Storage Facility in AWS S3:

Video Uploaded will be stored in one S3 bucket that will trigger a lambda function which will send the uploaded video to Elastic Transcoder and that transcoded video will be sent to other S3 destination bucket which will be stored in 3 different resolution.

Database Tool is FireBase:

The video once uploaded will trigger a lambda function of AWS which will tell the firebase to show concentric circle until transcoded video hits other S3 bucket. Once video gets transcoded the concentric circle evolving will be stopped and the video will be available to watch which is now shown on the website using Firebase.

Auth0 Service:

Auth0 will be again triggered by AWS Lambda through an API Gateway. This will tell when user logs in it will send the users credential to the Auth0 via API Gateway and will create a temporary token if user is authenticate which will then be sent back through API Gateway to the Lambda function which will tell our Web Server side code that user is authenticate and will allow to run Lambda function of user able to upload the code.

So how many Lambda functions are there in this project?

- **get-user-profile:** One which deals with Auth0 for getting user profile details**.**
- **custom-authorizer:** One which deals with Auth0 to get temperory token for authentication.
- **create-s3-upload-link:** Upload file to S3 when user click on Website and upload video on website side
- **push-transcoded-url-to-firebase:** Get video to firebase for using purpose in website
- **transcode-video:** When detected a video being uploaded will start transcoding that video.

CHAPTER 4
IMPLEMENTATION

4.1 Implementation Environment

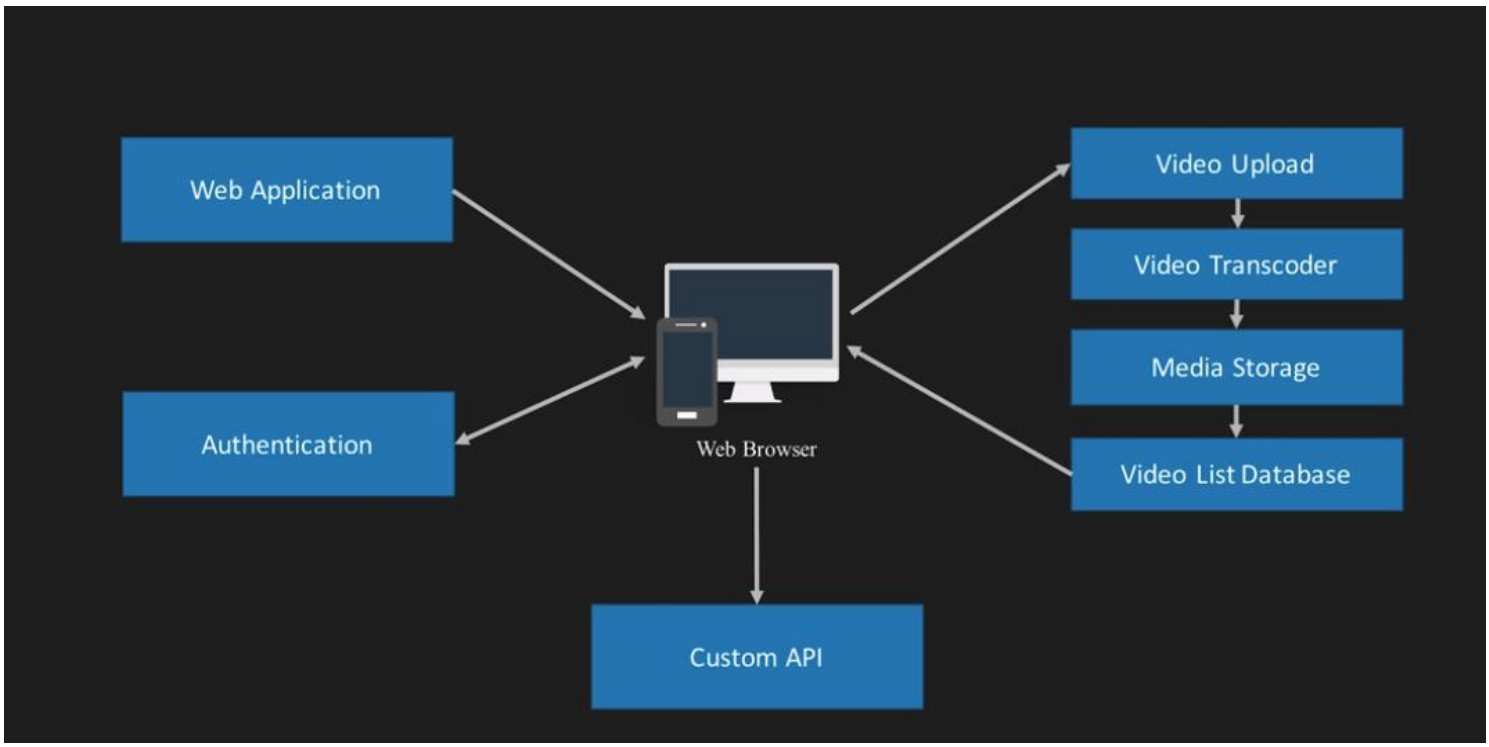


Figure 1 Implementation Environments

4.2 Coding Standard:

INDEX.HTML

```
<!doctype html>
<!--[if lt IE 7]>
<html class="no-js lt-ie9 lt-ie8 lt-ie7" lang=""> <![endif]-->
<!--[if IE 7]>
<html class="no-js lt-ie9 lt-ie8" lang=""> <![endif]-->
<!--[if IE 8]>
<html class="no-js lt-ie9" lang=""> <![endif]-->
<!--[if gt IE 8]><!-->
<html class="no-js" lang="">
<!--<![endif]-->

<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
  <title></title>
  <meta name="description" content="">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="apple-touch-icon" href="images/apple-touch-icon.png">

  <link rel="stylesheet" href="css/bootstrap.min.css">
  <style>
    body {
      padding-top: 50px;
      padding-bottom: 20px;
    }
  </style>
  <link rel="stylesheet" href="css/bootstrap-theme.min.css">
  <link rel="stylesheet" href="css/main.css">

  <script src="js/vendor/modernizr-2.8.3-respond-1.4.2.min.js"></script>
</head>

<body>
  <!--[if lt IE 8]>
<p class="browserupgrade">You are using an <strong>outdated</strong> browser. Please <a href="http://browsehappy.com/">upgrade
  your browser</a> to improve your experience.</p>
<![endif]-->
  <nav class="navbar navbar-inverse navbar-fixed-top" role="navigation">
    <div class="container">
      <div class="navbar-header">
        <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#navbar"
          aria-expanded="false" aria-controls="navbar">
          <span class="sr-only">Toggle navigation</span>
          <span class="icon-bar"></span>
          <span class="icon-bar"></span>
          <span class="icon-bar"></span>
        </button>
        <a class="navbar-brand" href="#">24 Hour Video</a>
      </div>
      <div id="navbar" class="navbar-collapse collapse">
        <div class="navbar-form navbar-right">
          <button id="user-profile" class="btn btn-default" onclick="alert('Not implemented yet')" <img
            id="profilepicture" />
          <span id="profilename"></span>
          </button>
```

```
                <button id="auth0-login" class="btn btn-success">Sign in</button>
                <button id="auth0-logout" class="btn btn-success">Sign Out</button>

            </div>
          </div>
          <!--/.navbar-collapse -->
        </div>
    </nav>

    <!-- Main jumbotron for a primary marketing message or call to action -->
    <div class="jumbotron">
        <div id="hero" class="container">
            <img src="images/popcorn.png" />
            <h1>All videos. All the time.</h1>

            <p>Guaranteed 100% server free.</p>
        </div>
    </div>

    <script src="//ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
    <script>window.jQuery || document.write('<script src="js/vendor/jquery-1.11.2.min.js"><\/script>')</script>

    <script src="js/vendor/bootstrap.min.js"></script>
    <script src="https://cdn.auth0.com/js/lock/11.16.0/lock.min.js"></script>

    <script src="js/user-controller.js"></script>
    <script src="js/config.js"></script>
    <script src="js/main.js"></script>

    <!-- Google Analytics: change UA-XXXXX-X to be your site's ID. -->
    <script>
        (function (b, o, i, l, e, r) {
            b.GoogleAnalyticsObject = l;
            b[l] || (b[l] =
                function () {
                    (b[l].q = b[l].q || []).push(arguments)
                });
            b[l].l = +new Date;
            e = o.createElement(i);
            r = o.getElementsByTagName(i)[0];
            e.src = '//www.google-analytics.com/analytics.js';
            r.parentNode.insertBefore(e, r)
        }(window, document, 'script', 'ga'));
        ga('create', 'UA-XXXXX-X', 'auto');
        ga('send', 'pageview');
    </script>
  </body>

  </html>
```

Config.js

```
var configConstants = {
    auth0: {
        domain: 'hetul-sheth.auth0.com',
        clientId: 'JjfRz7rKkDrPnNFPiaLJPQIiKUVH5X29'
    },
    apiBaseUrl: 'https://hal1zlcqm8.execute-api.us-east-1.amazonaws.com/dev'
};
```

main.js

```
(function () {
   $(document).ready(function () {
      userController.init(configConstants);
   });
}());
```

user-controller.js

```
var userController = {
   data: {
      auth0Lock: null,
      config: null
   },
   uiElements: {
      loginButton: null,
      logoutButton: null,
      profileButton: null,
      profileNameLabel: null,
      profileImage: null
   },
   init: function (config) {
      this.uiElements.loginButton = $('#auth0-login');
      this.uiElements.logoutButton = $('#auth0-logout');
      this.uiElements.profileButton = $('#user-profile');
      this.uiElements.profileNameLabel = $('#profilename');
      this.uiElements.profileImage = $('#profilepicture');

      this.data.config = config;
      var params = {
         autoclose: true,
         auth: {
            params: {
               scope: 'openid profile email user_metadata picture'
            },
            responseType: 'token'
         }
      };
      this.data.auth0Lock = new Auth0Lock(config.auth0.clientId, config.auth0.domain, params);

      // check to see if the user has previously logged in
      var accessToken = localStorage.getItem('accessToken');
      var idToken = localStorage.getItem('idToken');

      this.wireEvents();

      if (accessToken && idToken) {
         this.getUserProfile(accessToken, idToken);
      }

   },
   configureAuthenticatedRequests: function () {
      $.ajaxSetup({
         'beforeSend': function (xhr) {
            xhr.setRequestHeader('Authorization', 'Bearer ' + localStorage.getItem('idToken'));
         }
      });
   },
```

```
  getUserProfile: function (accessToken, idToken) {
    var that = this;
    this.data.auth0Lock.getUserInfo(accessToken, function (error, profile) {

      if (error) {
        return alert('There was an error getting the profile: ' + error.message);
      }

      that.configureAuthenticatedRequests();
      that.showUserAuthenticationDetails(profile);

    });
  },
  showUserAuthenticationDetails: function (profile) {
    var showAuthenticationElements = !!profile;

    if (showAuthenticationElements) {
      this.uiElements.profileNameLabel.text(profile.nickname || profile.email);
      this.uiElements.profileImage.attr('src', profile.picture);
    }

    this.uiElements.loginButton.toggle(!showAuthenticationElements);
    this.uiElements.logoutButton.toggle(showAuthenticationElements);
    this.uiElements.profileButton.toggle(showAuthenticationElements);
  },
  wireEvents: function () {
    var that = this;

    this.uiElements.loginButton.click(function (e) {
      that.data.auth0Lock.show();
    });

    this.uiElements.logoutButton.click(function (e) {
      localStorage.removeItem('accessToken');
      localStorage.removeItem('idToken');

      that.uiElements.logoutButton.hide();
      that.uiElements.profileButton.hide();
      that.uiElements.loginButton.show();
    });

    this.data.auth0Lock.on('authenticated', function (authResult) {
      localStorage.setItem('accessToken', authResult.accessToken);
      localStorage.setItem('idToken', authResult.idToken);
      that.getUserProfile(authResult.accessToken, authResult.idToken);
    });
  }
};
```

4.3 Snapshots:

User Page:

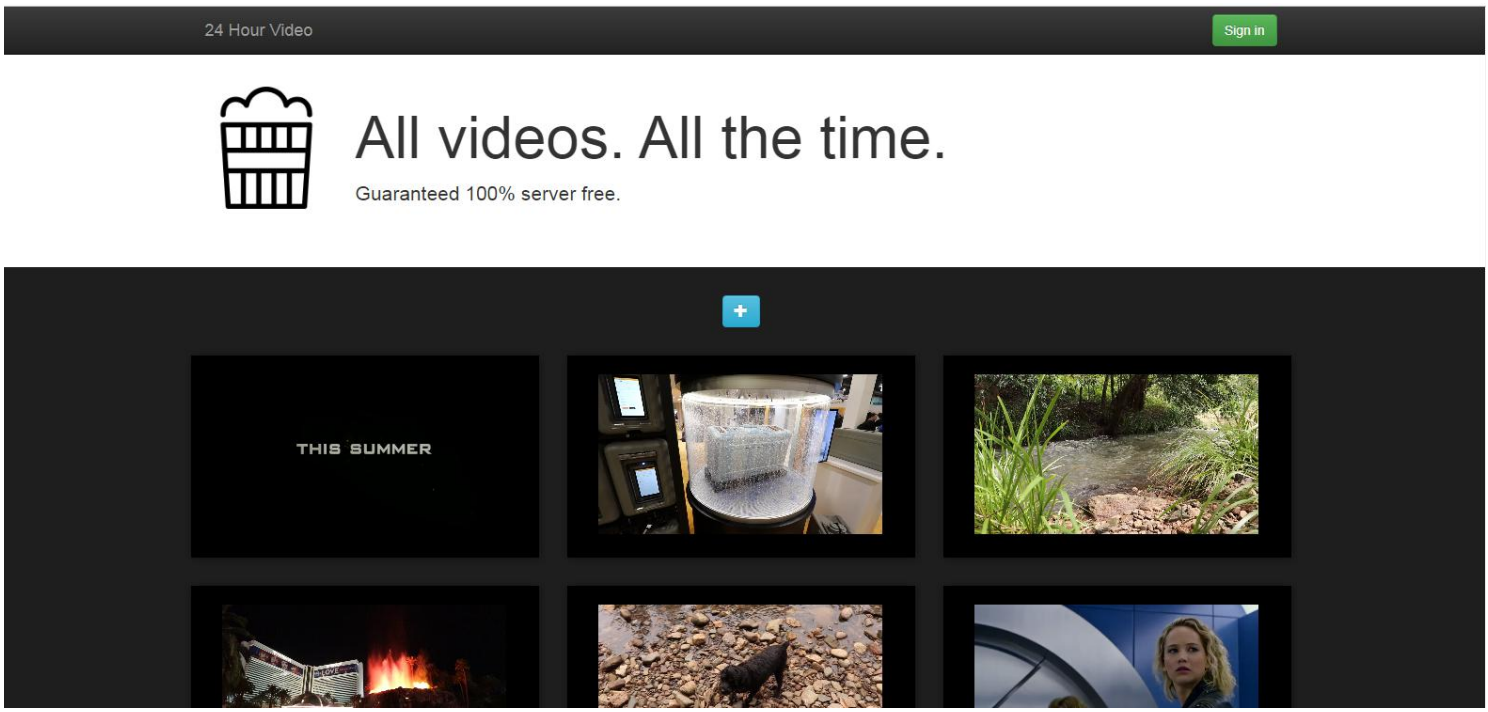
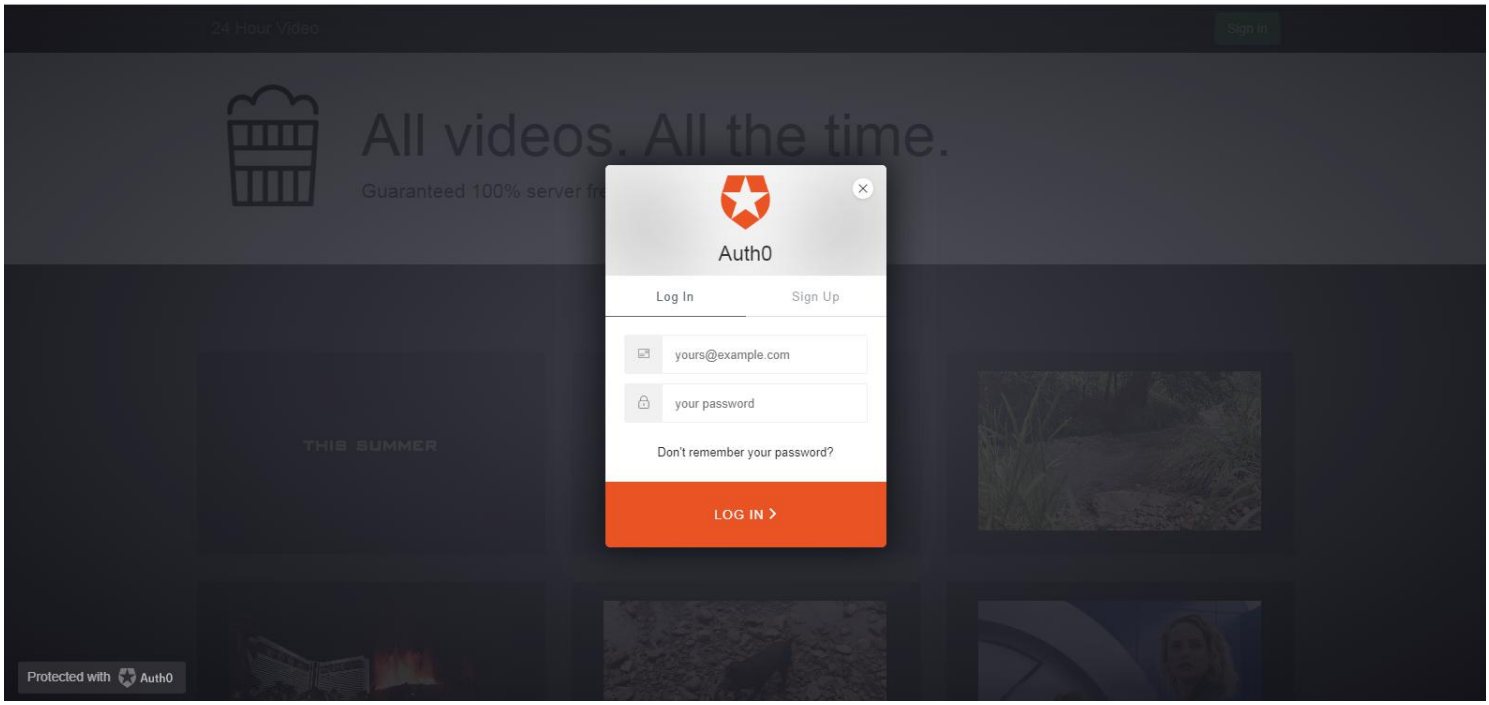
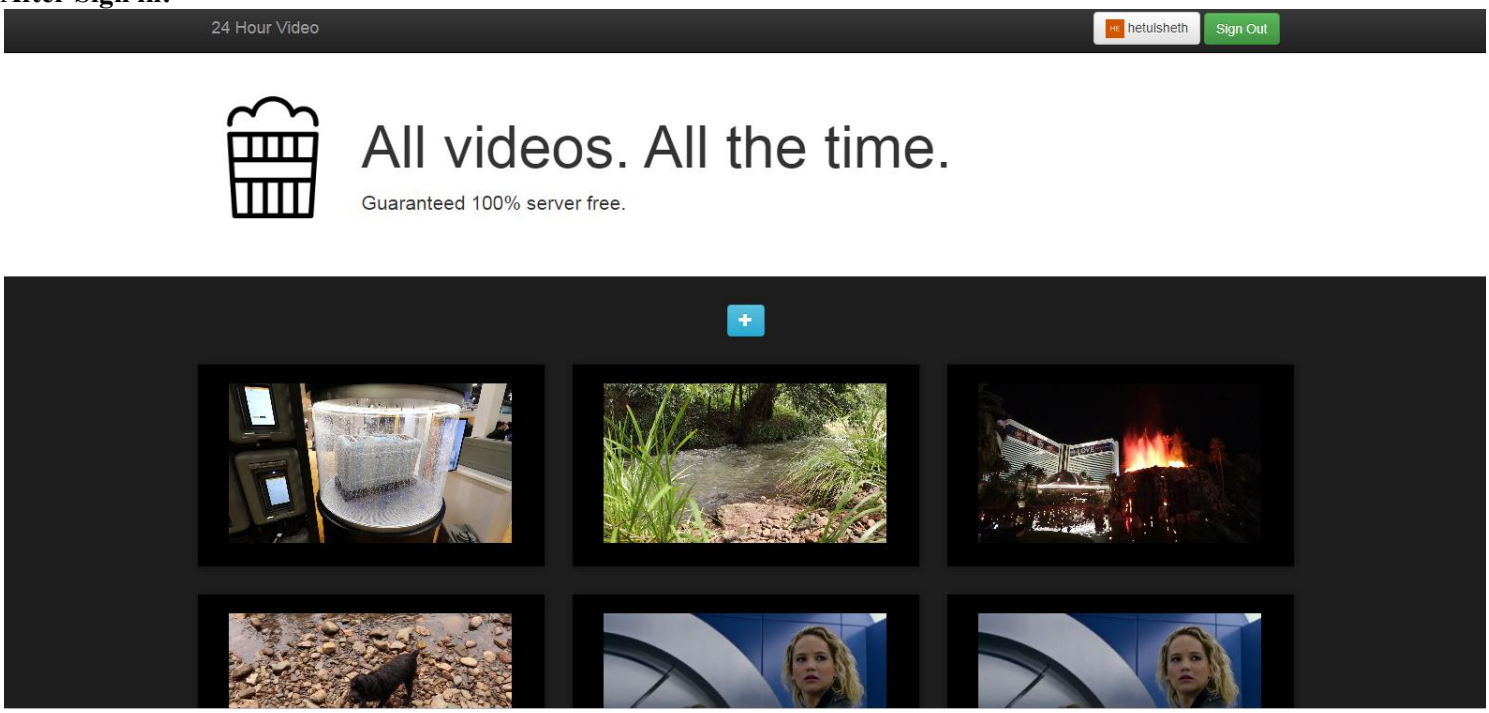


Figure 2 User Details

Sign up :



After Sign in:



File Upload Module:

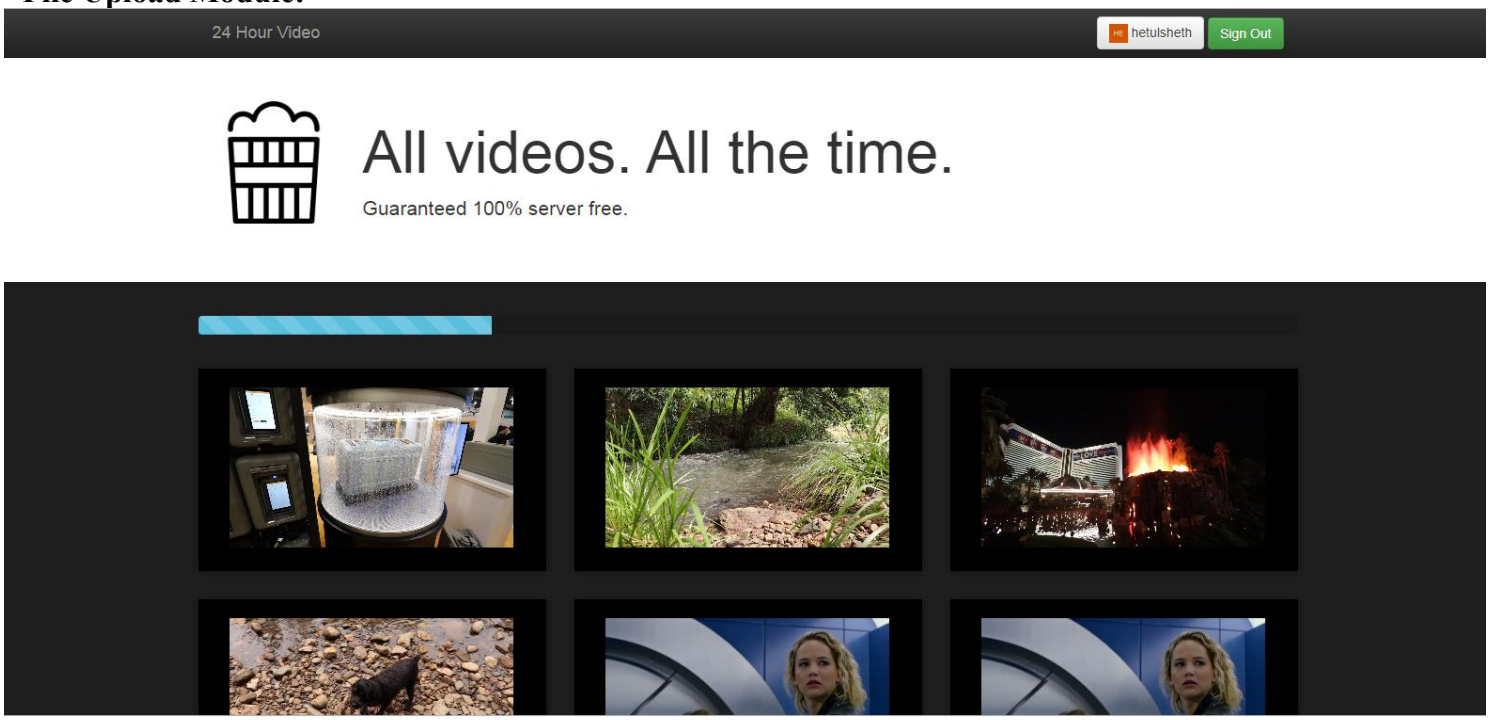


Figure 4 File Upload Module

Showing Transcoding Output:

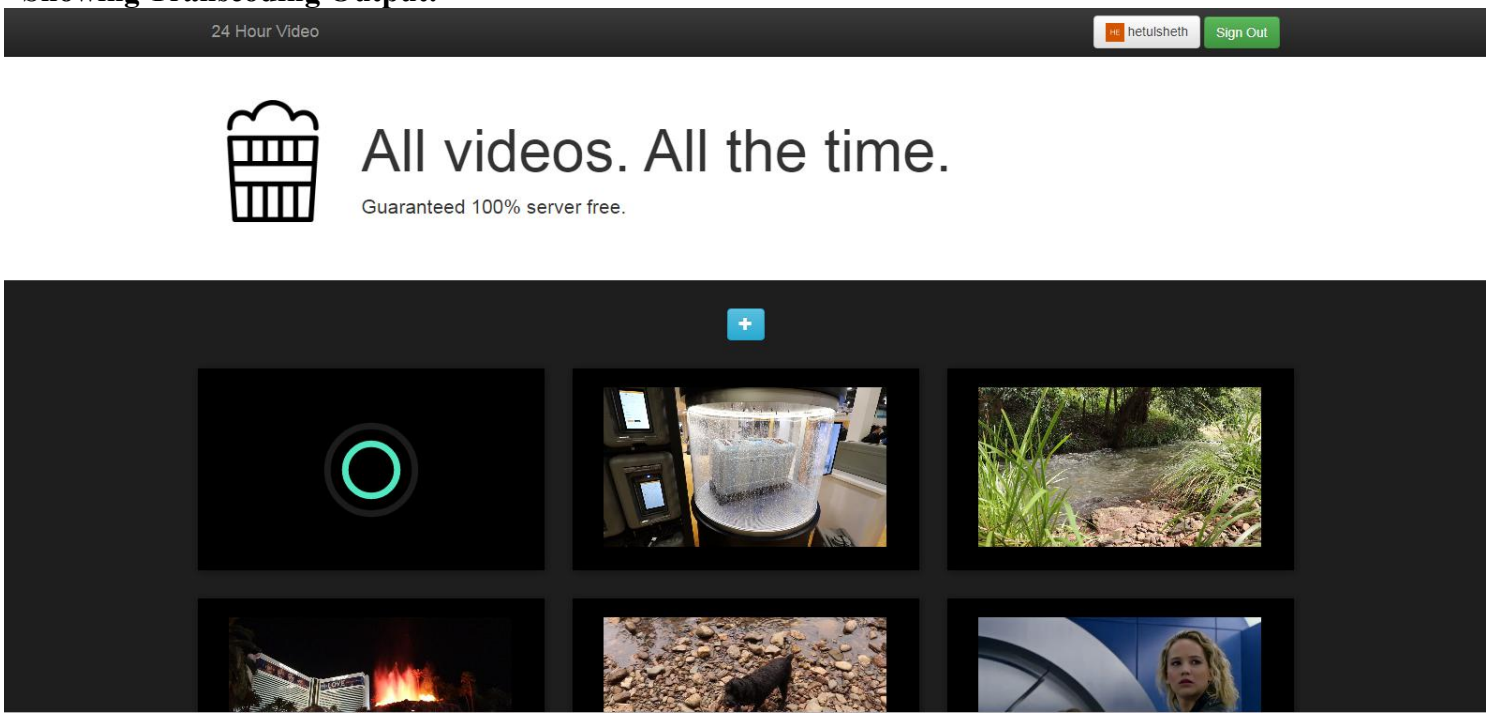


Figure 5 Output

CHAPTER 5

CONSTRAINTS

5.1 Constraints:

1. Once video gets uploaded there is no option to delete it.

2. Can transcode only video files.

3. No search option for videos if large list of video is available.

4. User cannot select the resolution right now. It will adjust to resolution as per your network speed. We will work on it soon.

CHAPTER 6

6.1 Conclusion:

Serverless enables you to build modern applications with increased agility and lower total cost of ownership. Building **serverless** applications means that your developers can focus on their core product instead of worrying about managing and operating servers or runtimes, either in the cloud or on-premises.

Serverless is the future.

6.2 References:

- https://aws.amazon.com/getting-started/projects/build-serverless-web-app-lambda-apigateway-s3-dynamodb-cognito/?trk=gs_card
- https://stackoverflow.com/questions/48304178/serverless-web-app-architecture