Steps to runt the application:

1. Pull the repository from git: <https://github.com/dipaksingbhukwal/photo_album>
2. Extract the ‘photo\_album-master.zip’ file and go to photo\_album-master-> photo\_album-master directory.
3. Edit ‘.env’ file for CLOUDINARY\_URL to set value:

CLOUDINARY\_URL = “your\_cloudinary\_url”

1. Open command prompt in the directory and run ‘npm install’.
2. Run the program through command prompt using ‘npm start’
3. Open browser and open link <http://localhost:9000/>
4. Upload the images using all three upload options to check the successful upload and overlay upon image retrieval.

To fulfil the requirement of assignment below changes were made:

1. **Automatically limit image size to 500x500 pixels on upload.**

* Change the ‘cloudinary.uploader.upload’ method and add two new parameters ‘height=500’ and ‘width=500’

  cloudinary.uploader

.upload(imageFile, {

tags: tag === "" ? "Default Tag" : tag,

width: 500,

height: 500

    }) // Set tag to default if now user tag is provided

1. **Tag uploaded images (doesn’t matter which tag)**

* New ‘input’ tag was added to ‘views/photos/add/ejs’

    <!-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* -->

    <!-- Below line of code changed to '2.  Tag uploaded images (doesn’t matter which tag)' -->

    <div *class*="form\_line">

      <label *for*="photo\_title">Title:</label>

      <div *class*="form\_controls">

        <input

*id*="photo\_tag"

*name*="tag"

*size*="30"

*type*="text"

*placeholder*="Enter your own image tag"

        />

      </div>

    </div>

    <!-- \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* -->

* Tag assignment code was changed in ‘controllers/photos\_controllers.js’

/\*

  Below line of code changed to 2.  Tag uploaded images (doesn’t matter which tag)'

  It fetches the user provided tag from req.body

  \*/

  // Get image tag

  var tag = req.body.tag;

1. **One with the Cloudinary logo as an overlay (watermark).**

* Overlay was added both on ‘views/photos/index.ejs’ and ‘views/photos/create\_through\_server.ejs’ pages to show cloudinary icon watermark.

**create\_through\_server.ejs**

<div *class*="photo">

  <h2><%= photo.title %></h2>

  <a *href*="<%= photo.image.url %>" *target*="\_blank">

    <%-cloudinary.image(photo.image.public\_id, {transformation: [ {width: 150,

    height: 150, quality: 80,crop:'fill',format:'jpg',class:'thumbnail inline'},

    {effect: "saturation:50"}, {overlay:

    {url:"https://res.cloudinary.com/dbhukwal/image/upload/v1568674498/samples/cloudinary-icon.png"},

    width: "0.25", flags: "relative", gravity: "north\_east", y: 10, x: 10}]})%>

  </a>

</div>

index.ejs

          <a *href*="<%= cloudinary.url(photo.image.public\_id)%>" *target*="\_blank">

            <%-cloudinary.image(photo.image.public\_id, {transformation: [

              {width: 150, height: 150, quality: 80,crop:'fill',format:'jpg',class:'thumbnail inline'},

              {effect: "saturation:50"},

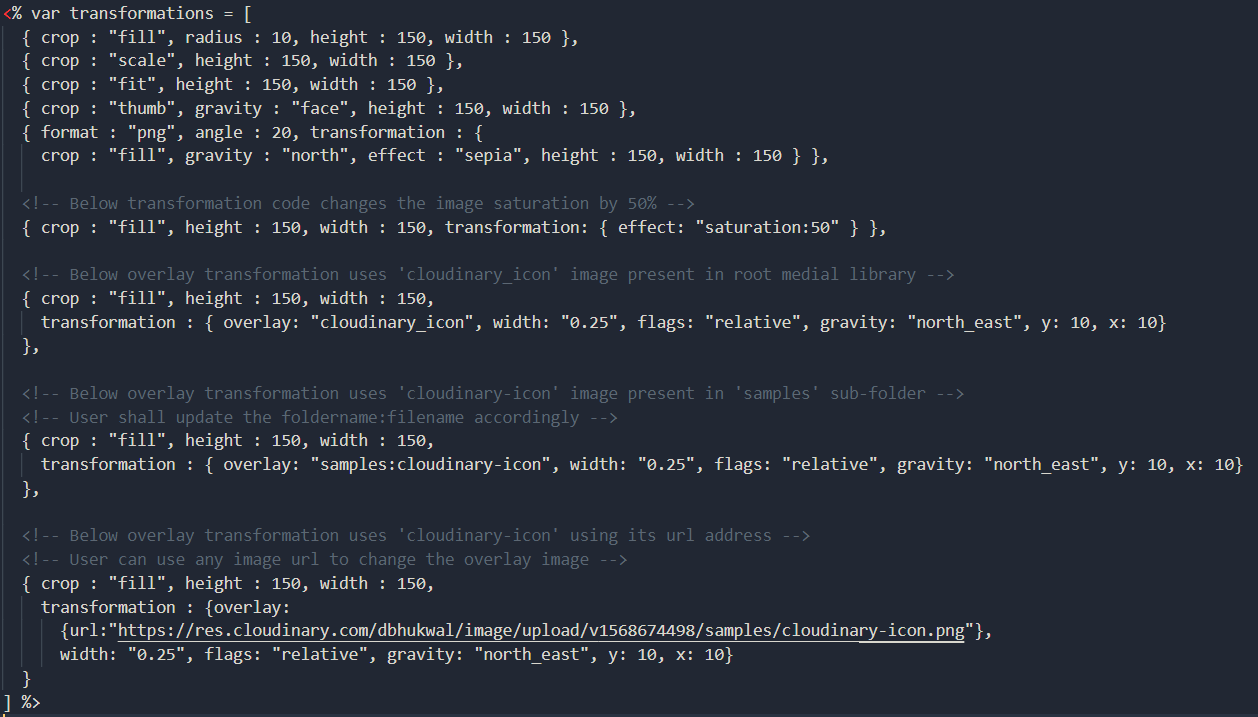
              {overlay:

              {url:"https://res.cloudinary.com/dbhukwal/image/upload/v1568674498/samples/cloudinary-icon.png"},

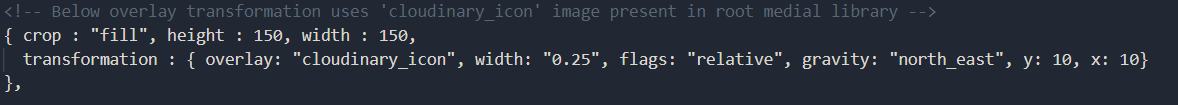
              width: "0.25", flags: "relative", gravity: "north\_east", y: 10, x: 10}]} )%>

          </a>

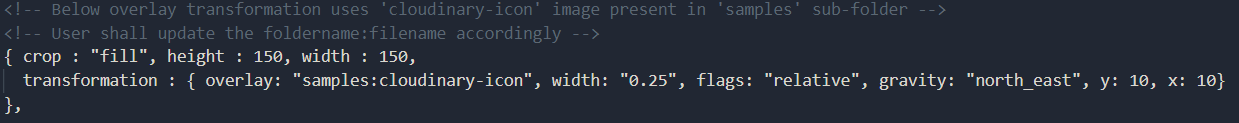
* In addition to above changes, under Hide transformations tab, three new transformation properties were added to add overlay image to uploaded image.



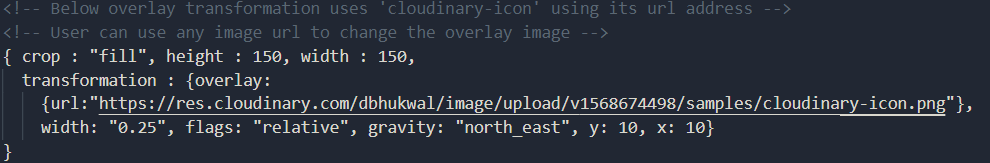
* The first overlay transformation uses the ‘cloudinary\_icon’ image present in root folder of media library. User can change the name of the images and use any image present in root library.



* The second overlay transformation uses the ‘cloudinary-icon’ image present in ‘sample’ sub-folder of root media library. User can change the name of the images and use any image present in ‘sample’ folder. The overlay url uses ‘:’ to specify the folder hierarchy while loading image.



* The third overlay transformation uses the ‘cloudinary-icon’ image present on internet. User can use any image present out there (internet) by using the address/url of that image.



1. **Second with the image saturation increased to 50%.**

* Saturation of the image was changed by adding saturation transformation object.
* Object was added both on ‘views/photos/index.ejs’ and ‘views/photos/create\_through\_server.ejs’ pages to increase image color saturation.

**create\_through\_server.ejs**

<div *class*="photo">

  <h2><%= photo.title %></h2>

  <a *href*="<%= photo.image.url %>" *target*="\_blank">

    <%-cloudinary.image(photo.image.public\_id, {transformation: [ {width: 150,

    height: 150, quality: 80,crop:'fill',format:'jpg',class:'thumbnail inline'},

    {effect: "saturation:50"}, {overlay:

    {url:"https://res.cloudinary.com/dbhukwal/image/upload/v1568674498/samples/cloudinary-icon.png"},

    width: "0.25", flags: "relative", gravity: "north\_east", y: 10, x: 10}]})%>

  </a>

</div>

**index.ejs**

          <a *href*="<%= cloudinary.url(photo.image.public\_id)%>" *target*="\_blank">

            <%-cloudinary.image(photo.image.public\_id, {transformation: [

              {width: 150, height: 150, quality: 80,crop:'fill',format:'jpg',class:'thumbnail inline'},

              {effect: "saturation:50"},

              {overlay:

              {url:"https://res.cloudinary.com/dbhukwal/image/upload/v1568674498/samples/cloudinary-icon.png"},

              width: "0.25", flags: "relative", gravity: "north\_east", y: 10, x: 10}]} )%>

          </a>

* With above code changes, the saturation transformation was added under ‘hide-transformation’ tab on <http://localhost:9000/photos>.

