**HTML CODE:**  
  
<template> <lightning-card title="Profile Picture" icon-name="utility:image">

  <div class="slds-box custom-container slds-grid slds-gutters custom-fixed-height">

    <!-- Div 1 -->

    <template if:true={showDiv1}>

      <div class="slds-col slds-size\_3-of-3">

        <lightning-card >

          <div>

            <lightning-input label="" accept="image/png,image/jpeg" name="file uploader" onchange={handleFilesChange} type="file">

            </lightning-input>

          </div>

          <br />

          <div class="slds-text-body\_small slds-text-color\_error">{fileName}

            <template if:true={showLoadingSpinner}>

              <lightning-spinner alternative-text="Uploading......" size="medium"></lightning-spinner>

            </template>

          </div>

          <br />

          <div>

            <template if:false={data}>

              <lightning-button class="slds-m-top--medium" label="Upload Image" onclick={handleSave}

                variant="brand" disabled={isTrue}></lightning-button>

            </template>

            <template if:true={data}>

              <lightning-button class="slds-m-top--medium" label="Upload Image" onclick={handleReplace}

                variant="brand" disabled={isTrue}></lightning-button>

            </template>

          </div>

          <br /><br />

        </lightning-card>

      </div>

    </template>

    <!-- Div 2 -->

    <template if:true={showDiv2}>

      <div class="slds-col slds-size\_3-of-3">

          <div class="slds-border\_left slds-border\_right slds-border\_bottom" style="height: 100%; padding: 5px;">

            <template if:false={data}>

              <i class="slds-text-color\_error">No image uploaded!</i>

            </template>

            <template if:true={imageUrl}>

              <img src={imageUrl} style="width: 100%; height: 95%;" alt="Uploaded Image">

            </template>

          </div>

      </div>

    </template>

  </div>

  <lightning-button label="Remove Picture" onclick={handleRemovePicture} variant="destructive" class="custom-button-padding"></lightning-button>

</lightning-card>

</template>

**Js. CODE:**  
  
import { LightningElement, track, api } from 'lwc';

import saveFile from '@salesforce/apex/ImageUploaderHandler.saveFile';

import setImageUrl from '@salesforce/apex/ImageUploaderHandler.setImageUrl';

import deleteFiles from '@salesforce/apex/ImageUploaderHandler.deleteFiles';

import { ShowToastEvent } from 'lightning/platformShowToastEvent';

export default class ImageUploaderLWC extends LightningElement {

  @api recordId;

  @track data;

  @track fileName = '';

  @track showLoadingSpinner = false;

  @track isTrue = false;

  filesUploaded = [];

  file;

  fileContents;

  fileReader;

  content;

  MAX\_FILE\_SIZE = 1500000;

  @track showDiv1 = true;

  @track showDiv2 = false;

  @track imageUrl;

  connectedCallback() {

    this.populateImageUrl();

  }

  handleFilesChange(event) {

    this.fileName = '';

    if (event.target.files.length > 0) {

      if (event.target.files[0].type != 'image/jpeg' && event.target.files[0].type != 'image/png') {

        this.fileName = 'Invalid file type!!';

        this.isTrue = true;

      } else {

        this.isTrue = false;

        this.filesUploaded = event.target.files;

        this.fileName = event.target.files[0].name;

      }

    }

  }

  handleSave() {

    if (this.filesUploaded.length > 0) {

      this.uploadHelper();

    } else {

      this.fileName = 'Please select a file to upload!!';

    }

  }

  handleReplace() {

    if (this.filesUploaded.length > 0) {

      deleteFiles({ recordId: this.recordId })

        .then(data => {

          console.log(data);

          this.uploadHelper();

        })

        .catch(error => {

          this.dispatchEvent(

            new ShowToastEvent({

              title: 'Error!!',

              message: error.message,

              variant: 'error',

            }),

          );

          this.fileName = '';

          this.isTrue = false;

          this.showLoadingSpinner = false;

          this.filesUploaded = [];

        });

    } else {

      this.fileName = 'Please select a file to upload!!';

    }

  }

  uploadHelper() {

    this.file = this.filesUploaded[0];

    if (this.file.size > this.MAX\_FILE\_SIZE) {

      this.dispatchEvent(

        new ShowToastEvent({

          title: 'Error',

          message: 'File Size is too large',

          variant: 'error',

        }),

      );

      return;

    }

    this.isTrue = true;

    this.showLoadingSpinner = true;

    this.fileReader = new FileReader();

    this.fileReader.onloadend = (() => {

      this.fileContents = this.fileReader.result;

      let base64 = 'base64,';

      this.content = this.fileContents.indexOf(base64) + base64.length;

      this.fileContents = this.fileContents.substring(this.content);

      this.saveToFile();

    });

    this.fileReader.readAsDataURL(this.file);

  }

  saveToFile() {

    saveFile({ recordId: this.recordId, strFileName: this.file.name, base64Data: encodeURIComponent(this.fileContents) })

      .then(result => {

        this.imageUrl = this.data;

        this.fileName = this.fileName + ' - Uploaded Successfully';

        this.isTrue = false;

        this.showLoadingSpinner = false;

        this.filesUploaded = [];

        this.dispatchEvent(

          new ShowToastEvent({

            title: 'Success!!',

            message: this.file.name + ' - Uploaded Successfully!!!',

            variant: 'success',

          }),

        );

        // Refresh the data after successful upload

        this.populateImageUrl();

      })

      .catch(error => {

        this.dispatchEvent(

          new ShowToastEvent({

            title: 'Error while uploading File',

            message: error.message,

            variant: 'error',

          }),

        );

        this.fileName = '';

        this.isTrue = false;

        this.showLoadingSpinner = false;

        this.filesUploaded = [];

      });

  }

  populateImageUrl() {

    setImageUrl({ recordId: this.recordId }).then(data => {

      this.data = data;

      console.log(data);

      if (data) {

        this.imageUrl = data;

        this.showDiv1 = false;

        this.showDiv2 = true;

      }

    }).catch(error => {

      this.dispatchEvent(

        new ShowToastEvent({

          title: 'Error!!',

          message: error.message,

          variant: 'error',

        }),

      );

    });

  }

  handleRemovePicture() {

    deleteFiles({ recordId: this.recordId })

      .then(data => {

        console.log(data);

        this.imageUrl = null;

        this.showDiv1 = true;

        this.showDiv2 = false;

      })

      .catch(error => {

        this.dispatchEvent(

          new ShowToastEvent({

            title: 'Error!!',

            message: error.message,

            variant: 'error',

          }),

        );

      });

  }

  // ... (any other methods or lifecycle hooks)

}

**CSS CODE:**  
  
/\* customComponent.css \*/

.custom-fixed-height {

  height: 350px; /\* Set the desired fixed height for the component \*/

}

/\* customComponent.css \*/

.custom-fixed-height .slds-border\_left {

  height: calc(80% - 40px); /\* Set the height to fit within the fixed container minus padding \*/

}

/\* customComponent.css \*/

.custom-button-padding {

  padding: 17px;

}

**APEX CLASS CODE:**  
  
public inherited sharing class ImageUploaderHandler {

    @AuraEnabled

    public static ContentVersion saveFile(Id recordId, String strFileName, String base64Data) {

        base64Data = EncodingUtil.urlDecode(base64Data, 'UTF-8');

        ContentVersion cv = new ContentVersion();

        cv.Title = strFileName;

        cv.PathOnClient = '/' + strFileName;

        cv.FirstPublishLocationId = recordId;

        cv.VersionData = EncodingUtil.base64Decode(base64Data);

        cv.IsMajorVersion = true;

        insert cv;

        return cv;

    }

    @AuraEnabled

    public static String setImageUrl(Id recordId){

        list<id> lstConDocs = new list<id>();

        for(ContentDocumentLink cntLink : [Select Id, ContentDocumentId From ContentDocumentLink Where LinkedEntityId =:recordId]) {

            lstConDocs.add(cntLink.ContentDocumentId);

        }

        if(!lstConDocs.isEmpty()) {

            ContentDistribution cdl = new ContentDistribution();

            cdl.ContentVersionId = [SELECT Id, Title, ContentDocumentId FROM ContentVersion WHERE ContentDocumentId IN :lstConDocs LIMIT 1].Id;

            cdl.Name = 'PublicShare';

            insert cdl;

            return [SELECT DistributionPublicUrl, ContentDownloadUrl FROM ContentDistribution WHERE Id = :cdl.Id LIMIT 1].ContentDownloadUrl;

        }

        else {

            return null;

        }

    }

    @AuraEnabled

    public static void deleteFiles(Id recordId){

        list<id> lstConDocs = new list<id>();

        for(ContentDocumentLink cntLink : [Select Id, ContentDocumentId From ContentDocumentLink Where LinkedEntityId =:recordId]) {

            lstConDocs.add(cntLink.ContentDocumentId);

        }

        if(!lstConDocs.isEmpty()) {

            delete [SELECT Id FROM ContentDocument WHERE id IN:lstConDocs];

        }

    }

 }