KOLEJ PROFESIONAL MARA BERANANG

DIPLOMA IN COMPUTER SCIENCE

COURSE NAME	:	**OBJECT ORIENTED PROGRAMMING**
COURSE CODE	:	**CSC2744**
ACADEMIC SESSION	:	**SESSION 1 2023/2024**
TYPE OF ASSESSMENT	:	**FINAL ASSIGNMENT**
DURATION	:	**20/6/2023-10/07/2023**

CLO 3: Employ third party data in object oriented application development using graphical user interface (GUI) application framework

INSTRUCTION TO CANDIDATES:

1. Late submissions after given due date will not be accepted.
2. Report should be written using:
   Font type: Arial
   Size: 12 pts
   Line Spacing: 1.5
3. Coding format:
   Font type: Consolas
   Size: 10 pts
   Line Spacing: Single

Personal Details	
Name	HANIS NABIHAH BT MOHD JAINI
I/D Number	BCS2207-072
Class	DCS 4C
Lecturer	PUAN AKMAL

Section / Question No.	**Marks**
Total	**/ 50**

Question

Electron is a powerful framework that enables developers to create cross-platform applications using web technologies such as HTML, CSS, and JavaScript. You need to choose one of the applications below to develop a desktop application using Electron that integrates the given API. The application needs to be developed with specific requirements or functionalities.

Name of Application	Description	Requirements
Dictionary and Thesaurus	An app that lists words in groups of synonyms and related concept.	• Word search. • Information Output: give the meaning of the searched word for different part of speech (noun/adjective), antonyms and the example of word usage, sounds and related URL for the searched word. • CRUD words of the day https://api.dictionaryapi.dev/api/v2/entries/en/digital
Meal planner	An app that displays suggestion of recipe based on food item entered by user	• Suggest recipe based on food item. • Information Output: One recipe suggestion that comprises of ingredients, instruction on how to cook and URL of the related site for the food and the link on how to prepare the food. • CRUD meal planner https://www.themealdb.com/api/json/v1/1/search.php?s=shawarma
Makeup Box	An app that finds makeup products based on brand and category entered.	• Display the product info according to search criteria. • Information Output: product description based on brand and name, product image, product website and related link for the searched product. • CRUD makeup top 5 list http://makeup-api.herokuapp.com/api/v1/products.json?brand=maybelline

Tasks:

1. Create desktop app using electron and apply third party data fetched from API and the requirements given. Your application should have at least 2 pages and you may add extra functionality or features of your choice to the application.
2. Implement CRUD (create, read, update, delete) process to the application as given in the requirements.
3. Apply HTML and CSS for user interface and provide evidence for application.
4. GUI Elements:
    i. Apply GUI elements that assist users in using application.
    ii. The application's 'look and feel' is attractive and informative.
5. Produce a report on your application functionalities and features. Include the following:
    i. Overview of your application with a brief description.
    ii. Screenshots of the application with explanations on how to use it.
    iii. Program codes of your system
6. Submit files in GitHub.

Assessment Rubric

ATTRIBUTES	CRITERIA	POOR (1 mark)	FAIR (2 marks)	GOOD (3 marks)	VERY GOOD (4 marks)	EXCELLENT (5 marks)	Mark Obtained
Reproduce and Process Information	1. Create desktop app using electron and apply third party data fetched from API and the requirements given. You may add extra functionality or features of your choice to the application.	The application is an extensive collection and rehash of other people's ideas, products, and images. There is no evidence of new thought.	The application is somewhat a collection and rehash of other people's ideas, products, and images. There is little evidence of new thought or inventiveness.	The application is a minimal collection or rehash of other people's ideas, products, and images. There is a few evidence of new thought or inventiveness.	The application shows a lot of evidence of originality and inventiveness.	The application shows significant evidence of originality and inventiveness. Most of the content and many of the ideas are fresh,original, and inventive.	
		Able to display part of the data from the API and does not fulfill the requirements.	Able to display sufficient data from the API that meet with the requirements together with the description.	Able to display sufficient data from the API that meet with the requirements together with the description.	Able to display extra data from the API beyond the application requirements together with no description.	Able to display extra data from the API beyond the application requirements together with the description.	
		The data from the API does not reflect the whole purpose of the application developed.	The data from the API is sufficient but does not reflect the whole purpose of the application developed.	The data from the API is meaningful but does not reflect the purpose of the application developed.	The data from the API is meaningful and reflect the purpose of the application developed.	The data from the API is meaningful to come up with extra idea for the application developed.	
		The developed application does not fulfill the requirements stated for the chosen	The developed application fulfills all the requirements stated for the chosen application with	The developed application fulfills all the requirements stated for the chosen application.	The developed application fulfills all the requirements stated for the chosen application.	The developed application fulfills all the requirements stated for the chosen application that	

		application.	no extra functionalities.	Add extra functionality or features to the application.	Add extra functionality or features to the application that enhances the user experience or adds value to the application.	utilizes the data from the API Add extra functionality or features to the application that enhances the user experience or adds value to the application.	
	2. Implement CRUD (create, read, update, delete) process to the application as given in the requirements.	• Able to create only 2 of the CRUD processes according to the requirements. • No feedback for the CRUD process. • The design for data input is poor	• Able to create only 3 of the CRUD processes according to the requirements. • No feedback for the CRUD process. • The design for data input is good with some room for improvement	• Able to perform all the CRUD processes according to the requirements. • No feedback for the CRUD process. • Well-designed data input for CRUD process.	• Able to perform all the CRUD processes according to the requirements. • No feedback for the CRUD process. • Well-designed data input for CRUD process.	• Able to perform all the CRUD processes according to the requirements. • Appropriate feedback for the CRUD process. • Well-designed and user-friendly data input for CRUD process.	
	3. Apply HTML and CSS for user interface and provide evidence for application.	• **Text -** All text used is too small to view or the font type is wrongly chosen. • **Graphics** - Graphics seem randomly chosen, are of low quality, OR distract the reader.	• **Text** – Some of the text used is too small to view or the font type is wrongly chosen. • **Graphics -** Graphics seem randomly chosen, are	• **Text** - Most text used is clear but does not describe the content well. • **Graphics** - Graphics are related to the theme/purpose of the application	• **Text** - All text used is clear but does not describe the content well. • **Graphics -** Graphics are related to the theme/purpose of the	• **Text** - All text used is clear and able to describe the content well. • **Graphics** - Graphics are related to the theme/purpose of the application, are thoughtfully	

			of low quality, OR distract the reader.	and are of excellent quality.	application, are of excellent quality and enhance reader interest or understanding	cropped, are of high quality and enhance reader interest or understanding.	
Curate	4.GUI Elements: i. Apply GUI elements that assist users in using application. i.	Not able to curate for required content. • **Layout** - The HTML elements in the application are cluttered looking or confusing. • **Navigation** Links do not take the reader to the sites/ pages described. User typically feels lost.	Limited curation for required content. • **Layout** - The HTML elements in the application is messy, may appear busy or boring. • **Navigation** Links seem to be missing and don't allow the user to easily navigate.	Satisfactory curation for required content. • **Layout** - The HTML elements are suitable. • **Navigation** Links allow the reader to move from page to page, but some links seem to be missing.	Good curation for required content. • **Layout** - The HTML elements are suitable and usable. • **Navigation** Links are labelled and allow the user to easily move from page to page.	Excellent curation for required content. • **Layout** - The HTML elements are well structured, attractive, and usable layout. • **Navigation** Links are clearly labelled, consistently placed, and allow the user to easily move from page to page.	
	ii. The application's 'look and feel' is attractive and informative.	• The application is in need of polish in its visual design and is not appropriate for the target audience. • **Color**	• The application is in need of polish in its visual design, but it is still appropriate for the target audience. • **Color**	• The application mostly follows good visual design principles (e.g.: alignment, contrast,	• The application demonstrates good visual design principles (e.g.: alignment, contrast,	• The application clearly demonstrates good visual design principles (e.g.: alignment,	

		Choice of colors and combinations are not suitable.	Choice of colors and combinations do not match the concept of the application.	easily read text) and is appropriate for the target audience. • **Color** Choice of colors and combinations match the concept of the application.	easily read text) and is appropriate for the target audience. • **Color** Appropriate colors used to produce an atmosphere that expresses the concept of the application.	contrast, easily read text) and is appropriate for the target audience. • **Color** Appropriate colors used to produce an atmosphere that expresses the concept of the application.	
Convey	5. Produce a report on your application functionalities and features that includes: i. Overview of the application. ii. Screenshots of the application with explanations on how to use it.	The overview of the application is vague. The user guide is incomplete and cannot be recognized as a user guide.	The overview of the application is very brief and does not describe the whole functionalities of the application. The user guide provides limited information with no screenshots of the application.	The overview of the application is clearly described the whole application and its functionalities. The user guide provides basic information with limited screenshots of the application.	The overview of the application is clearly described the whole application and its functionalities. The user guide provides adequate information with complete screenshots of the application.	The overview of the application is clearly described the whole application and its functionalities. The user guide provides extensive information with complete screenshots and labelling of the application.	
	iii. Program codes of the system	HTML, CSS and JavaScript codes attached are not complete. The codes are hardly read.	HTML, CSS and JavaScript codes attached are complete. The codes are hardly read.	HTML, CSS and JavaScript codes attached are complete. The codes are readable but not organized.	HTML, CSS and JavaScript codes attached are complete. The codes are readable and	HTML, CSS and JavaScript codes attached are complete and include comments for the important parts of the codes.	

		Does not submit complete electron files in GitHub	Completely submit all the electron file in GitHub.	Completely submit all the electron file in GitHub.	organized. Completely submit all the electron file in GitHub.	The codes are readable and organized. Completely submit all the electron file in GitHub.	
Total Marks Earned							**/50**
Total Percentage (40%)							**/40%**

i. Overview of your application with a brief description.

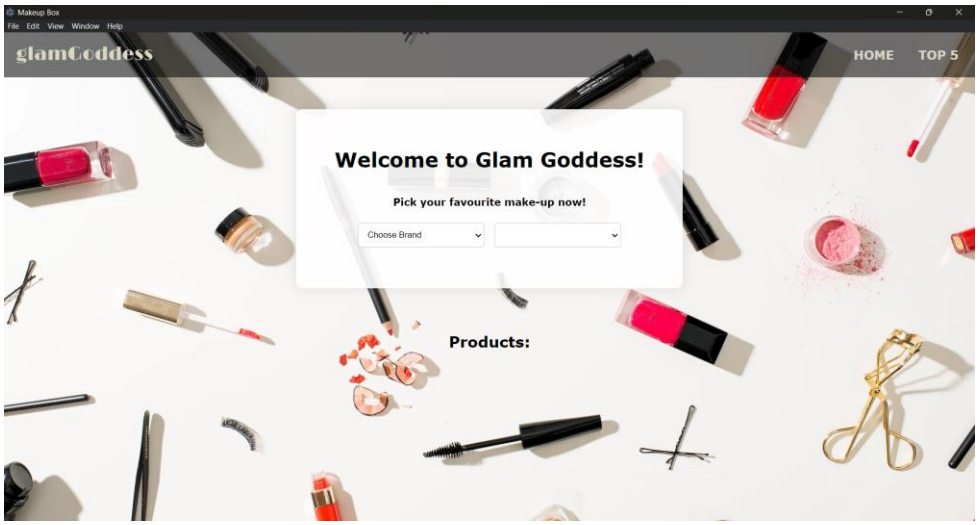
Glam Goddess is a user-friendly application designed to help users discover and organize their favorite makeup products. Two essential features are offered by this versatile platform. First, it lets users browse a wide range of makeup products that are labeled with brand and type, like "lipstick" or "eyeshadow." Once a product type has been chosen, the system presents a selected list of recommended products, including product details, images, official product website links, and related resources. Second, Glam Goddess empowers users to manage their top 5 makeup product lists using CRUD operations. With this system, users can create, read, update, and delete makeup products from their personalized top 5 lists. This organized and flexible approach considers their various preferences and to help users discover their favorite makeup products. Glam Goddess serves as a convenient tool for both exploring new makeup products and efficiently organizing personalized top 5 makeup products.

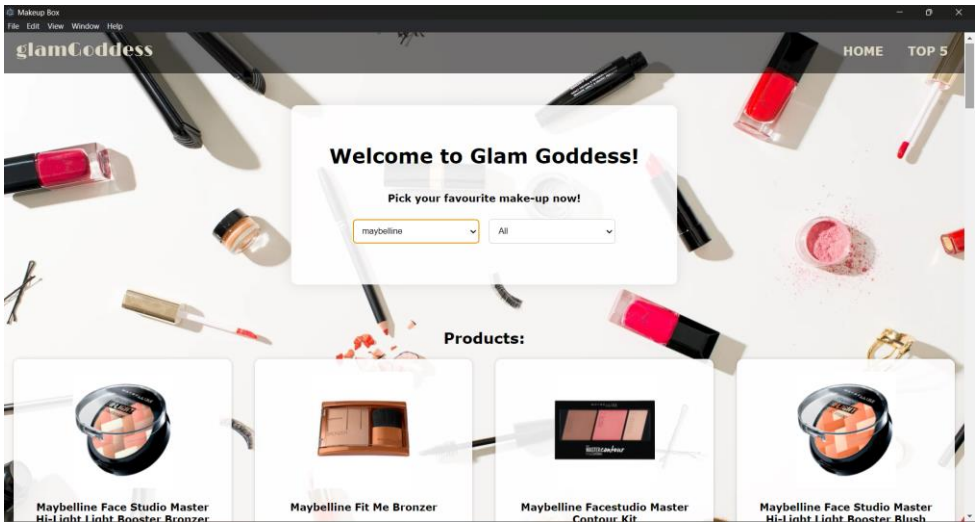
ii. Screenshots of the application with explanations on how to use it.

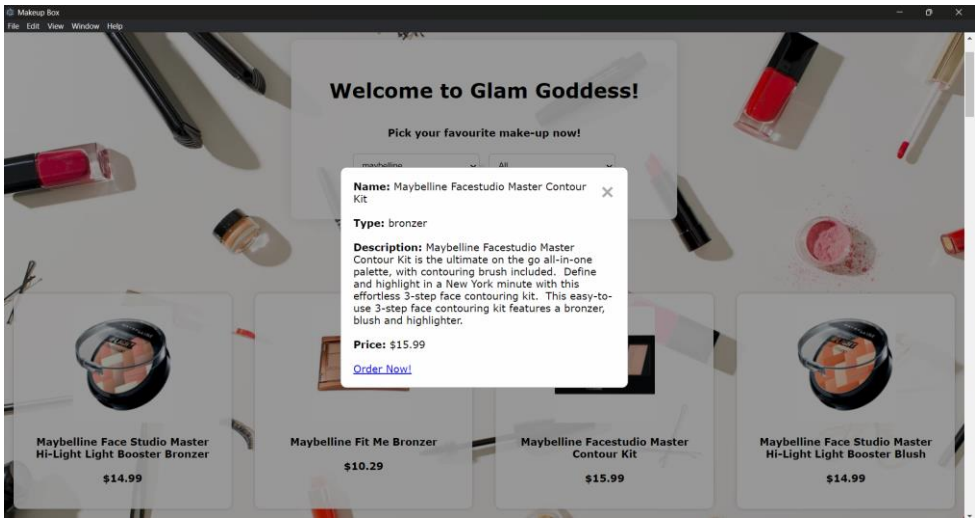
User guide

- File Name: home.html

  This is the file "home.html," which my system refers to as the "Home" page. When a user views my application, they will first see this page. There is a navigation menu with choices like "Home" and "Top 5" on this page. The user can access the top 5 list by clicking on "Top5". Users can search for any brand and product type depending on their preferences using the search options available on this page. Following brand selection, a dropdown menu with a list of available options for product type appears, and the system shows all makeup brands. Any suggested product's box can be clicked by users to view product details such as its name, type, description, price, and a link to the official product website. Users can also access the 'Top 5' page ('crud.html') by clicking on the 'Top 5' link in the navigation bar.

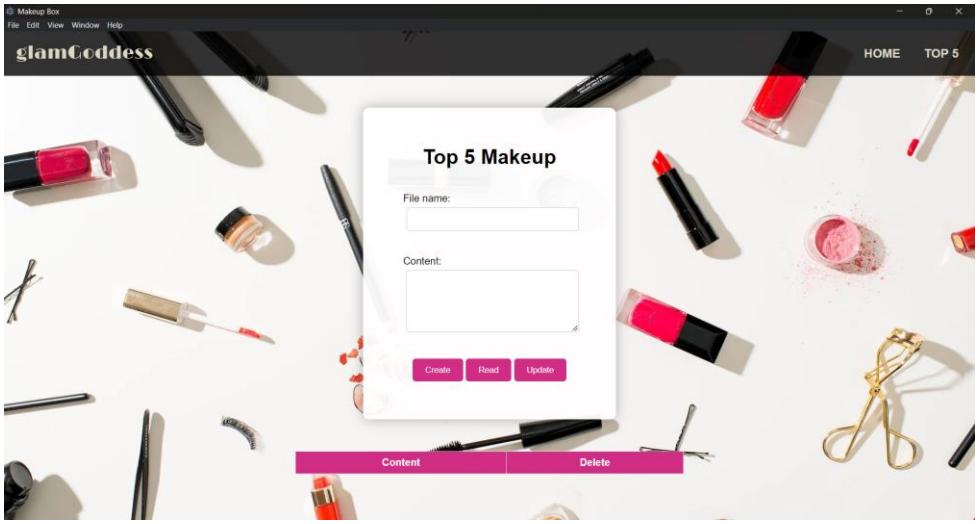

Makeup Box
File Edit View Window Help
glamGoddess
HOME
TOP 5
Welcome to Glam Goddess!
Pick your favourite make-up now!
Choose Brand
Products:

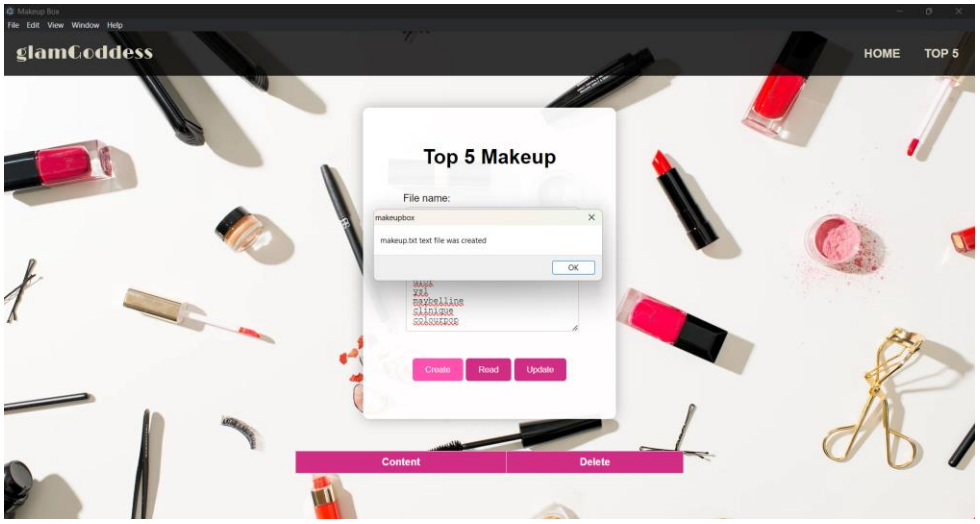

Makeup Box
File Edit View Window Help
glamGoddess
HOME
TOP 5
Welcome to Glam Goddess!
Pick your favourite make-up now!
maybelline
All
Products:
Maybelline Face Studio Master Hi-Light Light Booster Bronzer
Maybelline Fit Me Bronzer
Maybelline Facestudio Master Contour Kit
Maybelline Face Studio Master Hi-Light Light Booster Blush


Makeup Box
File Edit View Window Help
Welcome to Glam Goddess!
Pick your favourite make-up now!
Name: Maybelline Facestudio Master Contour Kit
Type: bronzer
Description: Maybelline Facestudio Master Contour Kit is the ultimate on the go all-in-one palette, with contouring brush included. Define and highlight in a New York minute with this effortless 3-step face contouring kit. This easy-to-use 3-step face contouring kit features a bronzer, blush and highlighter.
Price: $15.99
Order Now!
Maybelline Face Studio Master Hi-Light Light Booster Bronzer
$14.99
Maybelline Fit Me Bronzer
$10.29
Maybelline Facestudio Master Contour Kit
$15.99
Maybelline Face Studio Master Hi-Light Light Booster Blush
$14.99

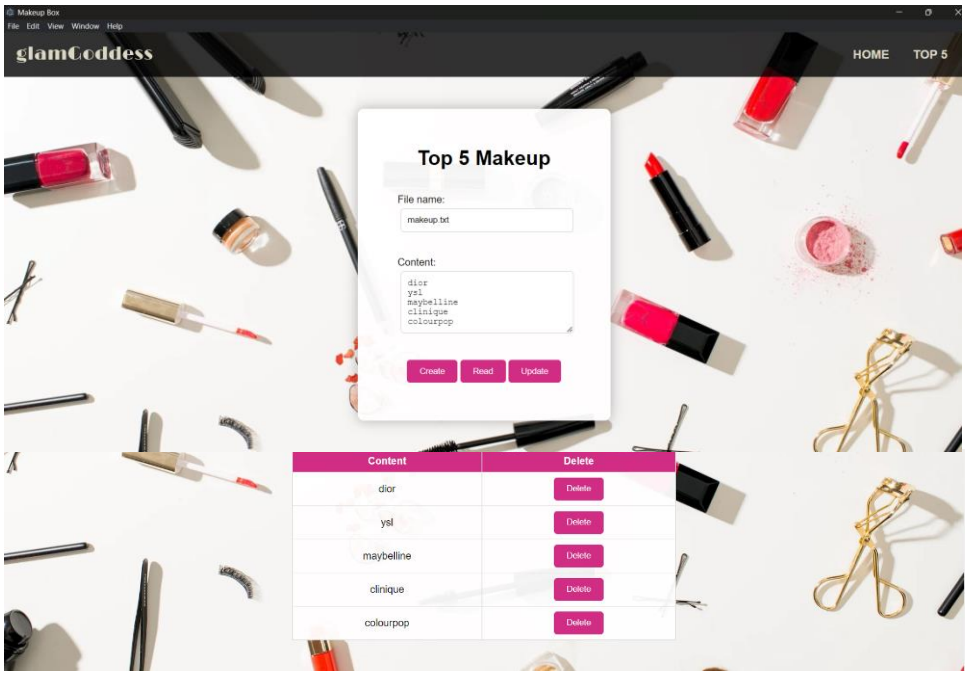
- File Name: crud.html

This is the "crud.html" file, which my system refers to as the "Top 5 Makeup" page. Users are given a form to fill out with their top 5 favorite makeup products, and they must create a new file on this page for all the content they enter to be saved in that file. When a user clicks the "Create" button after completing the form, the system will notify them that the content has been successfully created and added to the file. The alert will say "file has been created!". Additionally, a form that enables users to view and edit file they previously created and added to the file. Before the user selects the "Read" or "Update" button, the image following is shown:

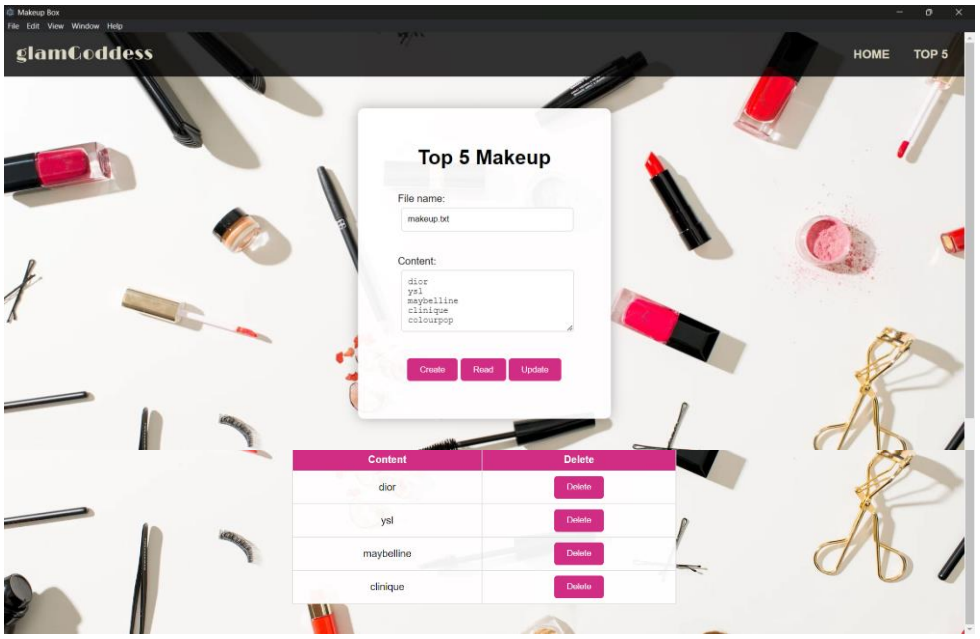




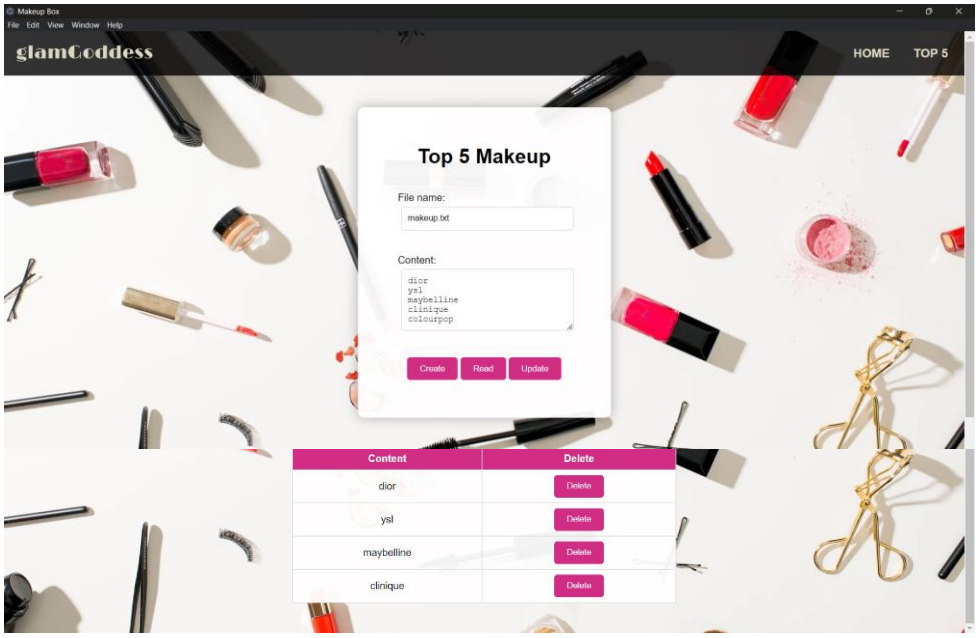
The user can read the content that they previously created and added to the file by clicking the "Read" button, which will cause the content from the file to be displayed in the table below. If a user wants to remove any content from the file, they can click the "Delete" button in the table, as seen below:



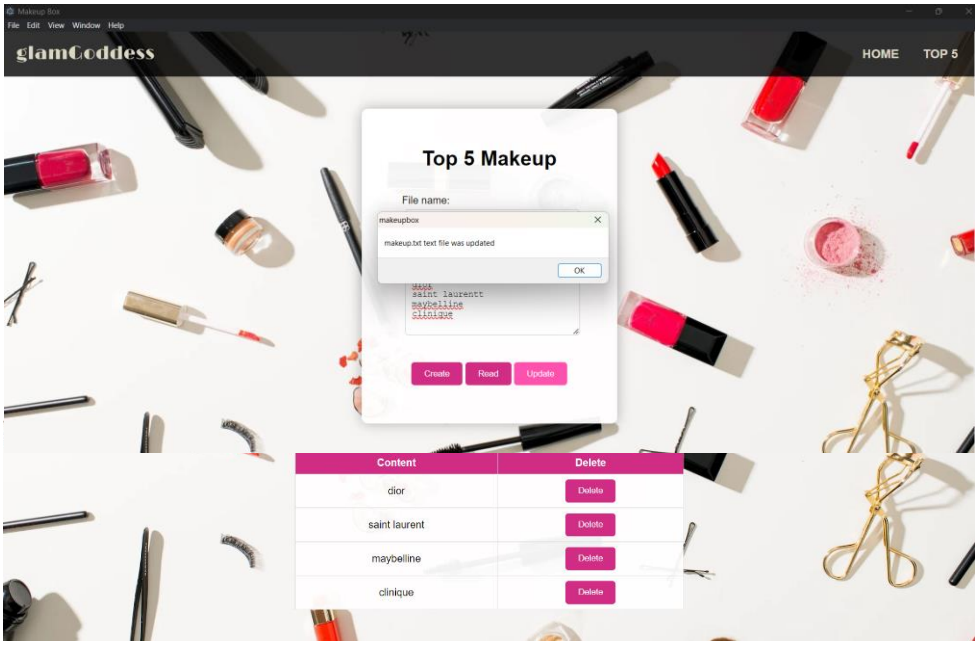
after clicking the “Delete” button:

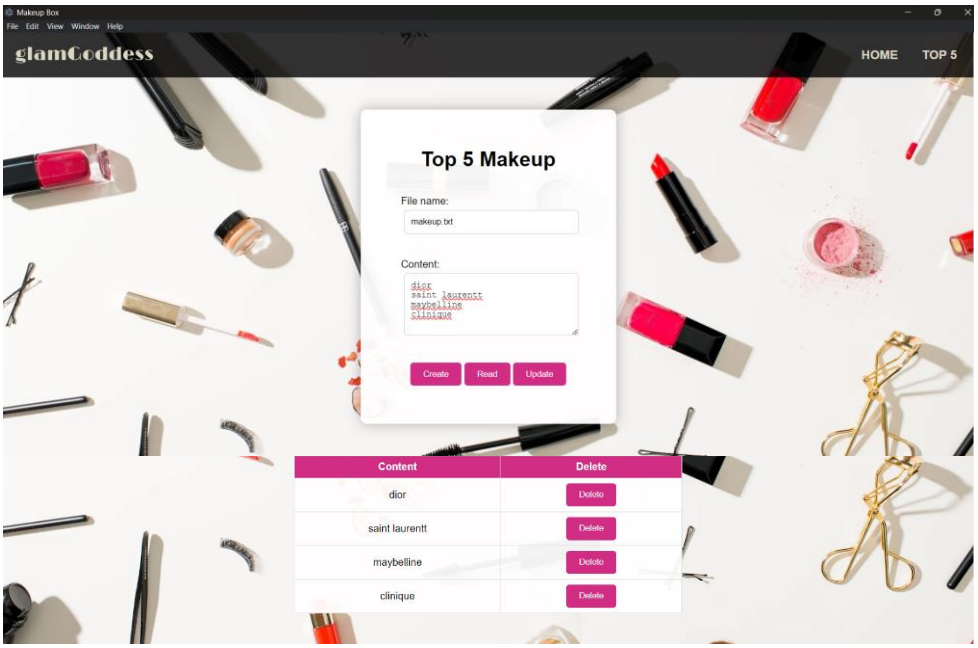


The user must first click the "Read" button to update the file. The content user entered in the form will be displayed by the system after user click "Read." User can edit the content within the form after it has been displayed, and then click the "Update" button. The user will be notified that the file has been successfully updated after clicking "Update". The alert will say "File was updated!". Here's an example of updated content in the form:



After clicking the "Update" button:





iii. Program codes of your system

- **home.css**

```
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

/* Body styles */
body {
  font-family: Verdana, Geneva, Tahoma, sans-serif;
  margin: 0;
  padding: 0;
  display: flex;
  flex-direction: column;
  background: #f9c1e2 url('bf.png');
  background-size: cover;
  background-position: center;
  background-attachment: fixed;
}

/*dropdowns*/
select {
  padding: 10px;
```

```
  border: 1px solid #ccc;
  border-radius: 5px;
  margin: 5px;
  width: 200px;
}

/*navigation bar*/
.navbar {
  background-color: rgba(0, 0, 0, 0.546);
  overflow: hidden;
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 10px 10px;
}

.navbar .logo {
  width: 215px;
  cursor: pointer;
  padding-top: 1px;
  padding-bottom: 1px;
  margin-left: 10px;
}

.nav-content {
  text-align: right;
  font-size: larger;
  font-weight: bold;
}

.nav-content a {
  display: inline-block;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
  color: #DDD8C3;
}

.nav-content a:hover {
  color: #daca8c;
```

```
}

.button-container {
  position: absolute;
  margin-top: 20px;
  top: 10px;
  right: 10px;
}

.top-right-button {
  padding: 10px 20px;
  background-color: #d22d85;
  color: white;
  border: none;
  border-radius: 5px;
  cursor: pointer;
}

/*product list*/
ul#productList {
  list-style: none;
  padding: 0;
  display: flex;
  flex-wrap: wrap;
  justify-content: space-between;
}

/*product container*/
ul#productList li {
  flex-basis: calc(24% - 20px);
  margin-left: 1%;
  margin-right: 1%;
  background-color: rgba(255, 255, 255, 0.897);
  border-radius: 10px;
  box-shadow: 0 0 5px rgba(0, 0, 0, 0.2);
  display: flex;
  flex-direction: column;
  transition: transform 0.3s ease;
  margin-bottom: 20px;
  color: black;
```

```
  padding: 20px;
  text-align: center;
  box-sizing: border-box;
}

ul#productList li:hover {
  transform: scale(1.05);
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.3);
}

/*product links*/
.product-container a {
  color: #d22d85;
  text-decoration: none;
  transition: color 0.3s ease;
}

.product-container a:hover {
  color: #f19fcf;
}

/*form container*/
.form-container {
  display: flex;
  align-items: center;
}

.product-container:hover {
  transform: scale(1.05);
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.3);
}

.product-container img {
  transition: transform 0.3s ease;
}

.product-container img:hover {
  transform: scale(1.1);
}
```

```
/*product links*/
.product-container a {
  color: #d22d85;
  text-decoration: none;
  transition: color 0.3s ease;
}

.product-container a:hover {
  color: #f19fcf;
}

/*modal display*/
.modal {
  display: none;
  position: fixed;
  z-index: 1;
  left: 0;
  top: 0;
  width: 100%;
  height: 100%;
  overflow: auto;
  background-color: rgba(0,0,0,0.4);
}

.modal-content {
  background-color: #fefefe;
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  padding: 20px;
  border: 1px solid #888;
  border-radius: 10px;
  width: 30%;
  color: grey;
}

.close {
  color: #aaaaaa;
  float: right;
```

```
  font-size: 28px;
  font-weight: bold;
}

.close:hover,
.close:focus {
  color: #000;
  text-decoration: none;
  cursor: pointer;
}

/* Container settings */
.container {
  padding: 60px;
  border-radius: 10px;
  box-shadow: 0 0 20px rgba(0, 0, 0, 0.1);
  background-color: rgba(255, 255, 255, 0.9);
  margin: 50px auto;
  max-width: 700px;
}

body .container {
  display: flex;
  flex-direction: column;
  align-items: center;
}

/*input field and dropdown*/
input[type="text"],
select {
  padding: 10px;
  border: 1px solid #ccc;
  border-radius: 5px;
  margin: 5px;
}

/*heading*/
h1 {
  color: black;
  text-align: center;
```

```
}

h2 {
  color: black;
  font-weight: bold;
  text-align: center;
}

h4{
  color: black;
}

p{
  color: #000000;
}

/*image */
img {
  max-width: 100%;
  border-radius: 5px 5px 0 0;
  display: block;
  margin: auto;
  padding-bottom: 20px;
}
```

- **home.html**

```
<!DOCTYPE html>

<html>

  <head>

    <meta charset="UTF-8" />

    <!-- PAGE TITLE -->

    <title>Makeup Box</title>

    <!--STYLE AND JAVASCRIPT-->

    <link rel="stylesheet" href="home.css">


  </head>
```

```
<body>

  <!-- NAVIGATION BAR -->
  <div class="navbar">
    <!-- LOGO -->
    <img src="logo.png" class="logo">
    <div class="nav-content">
        <a href="home.html">HOME</a>
        <a href="crud.html">TOP 5</a>
    </div>
  </div>

  <!-- MAIN CONTENT CONTAINER -->
  <div class="container">
    <center>
      <h1>Welcome to Glam Goddess!</h1>
      <br><br>
      <h4>Pick your favourite make-up now!</h4>
      <br>
      <!-- DROPDOWN FOR BRAND -->
      <select id="brandDropdown" onchange="getProductsByBrand()"></select>
      <!-- DROPDOWN FOR PRODUCT TYPE -->
      <select id="productTypeDropdown" onchange="getProductsByType()"></select>
    </center>
  </div>

  <!-- MAKEUP DETAILS MODAL -->
  <div id="makeupModal" class="modal">
    <div class="modal-content">
```

```
            <!-- CLOSE BUTTON -->
            <span class="close" onclick="closeModal()">&times;</span>
            <!-- PRODUCT DETAILS -->
            <p><b>Name:</b> <span id="productNameSpan"></span></p>
            <br>
            <p><b>Type:</b> <span id="productTypeSpan"></span></p>
            <br>
            <p><b>Description:</b> <span id="descriptionSpan"></span></p>
            <br>
            <p><b>Price:</b> <span id="priceSpan"></span></p>
            <br>
            <p><b>Link:</b> <span id="productLinkSpan"></span></p>
          </div>
        </div>

        <!-- LIST OF PRODUCTS -->
        <br>
        <h2>Products:</h2>
        <br>
        <ul id="productList"></ul>

        <!--JAVASCRIPT -->
        <script src="home.js"></script>
      </body>
        </html>
```

- **home.js**

```
// FETCH BRANDS IN THE DROPDOWN
function fetchBrands() {
```

```
  // Fetch data from API to get makeup products
  fetch('http://makeup-api.herokuapp.com/api/v1/products.json')
      .then(response => response.json())
      .then(data => {

          const brands = data.map(item => item.brand).filter((value, index, self) =>
   self.indexOf(value) === index);

          const brandDropdown = document.getElementById('brandDropdown');
          brandDropdown.innerHTML = '<option value="">Choose Brand</option>';

          brands.forEach(brand => {
              const option = document.createElement('option');
              option.value = brand;
              option.text = brand;
              brandDropdown.appendChild(option);
          });
      })
      .catch(error => {
          console.error("Error fetching brands:", error);
      });
}

// FETCH AND DISPLAY PRODUCTS BY SELECTED BRAND
function getProductsByBrand() {
  const selectedBrand = document.getElementById('brandDropdown').value;

  // Fetch products based on the selected brand
  fetch(`http://makeup-api.herokuapp.com/api/v1/products.json?brand=${selectedBrand}`)
```

```
        .then(response => response.json())
        .then(data => {
            populateProductTypes(data);
            displayProducts(data);
        })
        .catch(error => {
            console.error("Error fetching products:", error);
            document.getElementById("productList").innerHTML = "Error fetching
    products.";
        });
}

// POPULATE PRODUCT TYPES DROPDOWN
function populateProductTypes(data) {
  const productTypeDropdown = document.getElementById('productTypeDropdown');
  productTypeDropdown.innerHTML = '<option value="">All</option>';

  const productTypes = data.map(item => item.product_type).filter((value, index, self)
   => self.indexOf(value) === index);

  productTypes.forEach(type => {
      const option = document.createElement('option');
      option.value = type;
      option.text = type;
      productTypeDropdown.appendChild(option);
  });
}

// FETCH AND DISPLAY PRODUCTS BY SELECTED TYPE
```

```
function getProductsByType() {
  const selectedBrand = document.getElementById('brandDropdown').value;
  const selectedType = document.getElementById('productTypeDropdown').value;

  let url = `http://makeup-
   api.herokuapp.com/api/v1/products.json?brand=${selectedBrand}`;
  if (selectedType) {
      url += `&product_type=${selectedType}`;
  }

  // Fetch products based on the selected brand and type
  fetch(url)
      .then(response => response.json())
      .then(data => {
          displayProducts(data);
      })
      .catch(error => {
          console.error("Error fetching products:", error);
          document.getElementById("productList").innerHTML = "Error fetching
   products.";
      });
}

// DISPLAY PRODUCTS IN THE LIST
function displayProducts(products) {
  const productList = document.getElementById("productList");
  productList.innerHTML = '';

  products.forEach(product => {
```

```
    const li = document.createElement('li');
    const detailsDiv = document.createElement('div');

    li.addEventListener('click', function () {
        displayProductDetails(product);
    });

    if (product.image_link) {
        const img = document.createElement('img');
        img.src = product.image_link;
        img.dataset.productType = product.product_type || '';
        img.dataset.description = product.description || '';
        img.dataset.productLink = product.product_link || '';
        img.style.maxWidth = '200px';
        img.classList.add('productImage');
        detailsDiv.appendChild(img);
    }

    if (product.name) {
        detailsDiv.innerHTML += `<b>${product.name}</b> <br><br>`;
    } else {
        detailsDiv.innerHTML += `<b>Product Name:</b> Product Name not available
<br><br>`;
    }

    if (product.price) {
        detailsDiv.innerHTML += `<b>$${product.price}</b> <br><br>`;
    } else {
        detailsDiv.innerHTML += `<b>Price:</b> Price not available <br><br>`;
```

```
      }

      li.appendChild(detailsDiv);
      productList.appendChild(li);
  });
}

// DISPLAY PRODUCT DETAILS
function displayProductDetails(product) {
  const name = product.name || 'Product name not available';
  const productType = product.product_type || 'Product type not available';
  const description = product.description || 'Description not available';
  const price = `$${product.price}` || 'Price not available';
  const productLink = product.product_link || 'Product link not available';

  displayModal(name, productType, description, price, productLink);
}

// DISPLAY MODAL WITH PRODUCT DETAILS
function displayModal(name, productType, description, price, productLink) {
  const modal = document.getElementById('makeupModal');
  const productNameSpan = document.getElementById('productNameSpan');
  const productTypeSpan = document.getElementById('productTypeSpan');
  const descriptionSpan = document.getElementById('descriptionSpan');
  const priceSpan = document.getElementById('priceSpan');
  const productLinkSpan = document.getElementById('productLinkSpan');

  productNameSpan.textContent = name;
  productTypeSpan.textContent = productType;
```

```
    descriptionSpan.textContent = description;
    priceSpan.textContent = price;
    productLinkSpan.innerHTML = productLink !== 'Product link not available' ?
      `<a href="${productLink}" target="_blank">Order Now!</a>` : 'Product link not
     available';

    modal.style.display = 'block';
  }

  // CLOSE THE MODAL
  function closeModal() {
    document.getElementById('makeupModal').style.display = 'none';
  }

  // FETCH BRANDS WHEN THE PAGE LOADS
  fetchBrands();
```

- **crud.css**

```
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

/* Body styles */
body {
  font-family: 'Arial', sans-serif;
  margin: 0;
  padding: 0;
```

```
  display: flex;
  flex-direction: column;
  background: #f9c1e2 url('bf.png');
  background-size: cover;
  background-position: center;
  background-attachment: fixed;
}

/*dropdowns*/
select {
  padding: 10px;
  border: 1px solid #ccc;
  border-radius: 5px;
  margin: 5px;
  width: 200px;
}

/*navigation bar*/
.navbar {
  background-color: rgba(0, 0, 0, 0.8);
  overflow: hidden;
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 10px;
}

.navbar .logo {
  width: 215px;
```

```
  cursor: pointer;
  padding-top: 1px;
  padding-bottom: 1px;
  margin-left: 10px;
}

.nav-content {
  text-align: right;
  font-size: larger;
  font-weight: bold;
}

.nav-content a {
  display: inline-block;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
  color: #DDD8C3;
}

.nav-content a:hover {
  color: #daca8c;
}

/* Container settings */
.container {
  padding: 60px;
  border-radius: 10px;
  box-shadow: 0 0 20px rgba(0, 0, 0, 0.3);
```

```
  background-color: rgba(255, 255, 255, 0.95);
  margin: 50px auto;
  max-width: 400px;
}

body .container {
  display: flex;
  flex-direction: column;
  align-items: center;
}

/*Main heading*/
h1 {
  color: black;
  text-align: center;
  margin-top: 0px;
  margin-bottom: 20px;
}

/*Form*/
form {
  display: flex;
  flex-direction: column;
}

.form-group {
  margin-bottom: 15px;
}
```

```
label {
  margin-bottom: 5px;
}

input[type="text"],
textarea {
  padding: 10px;
  border: 1px solid #ccc;
  border-radius: 5px;
  margin: 5px;
  width: 100%;
  box-sizing: border-box;
}

/* Button*/
button {
  background-color: #d22d85;
  color: white;
  padding: 10px 20px;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  transition: background-color 0.3s ease;
}

button:hover {
  background-color: #ff51ae;
}
```

```
/* Table*/

table {

  width: 40%;

  border-collapse: collapse;

  margin-top: 20px;

  margin-bottom: 50px;

}


th,

td {

  border: 1px solid #ddd;

  padding: 8px;

  text-align: center;

  background-color: rgba(255, 255, 255, 0.95);

}


th {

  background-color: #d22d85;

  color: white;

     }
```

- **crud.html**

```
<!DOCTYPE html>

<html lang="en">

<head>

    <!-- TITLE -->

    <title>Makeup Box</title>

    <!-- CSS -->

    <link rel="stylesheet" type="text/css" href="crud.css">
```

```
</head>
<body>

    <!-- NAVIGATION -->
    <header class="navbar">
        <img src="logo.png" class="logo">
        <div class="nav-content">
          <a href="home.html">HOME</a>
          <a href="crud.html">TOP 5</a>
      </div>
    </header>
    <main>
              <div class="container">
                <div class="form-group">
                  <h1>Top 5 Makeup</h1><br>
                    <!--FILE FORM-->
                    <label>File name:</label>
                    <input id="fileName" type="text" class="form-control">
                </div>
                <br>

                <div class="form-group">
                    <!--CONTENT FORM-->
                    <label for="fileContents">Content:</label>
                    <textarea id="fileContents" class="form-control" rows="5"></textarea>
                </div>
                <br>
                <!--BUTTON-->
                <center>
```

```
                <button id="btnCreate" class="btn btn-default">Create</button>
                <button id="btnRead" class="btn btn-default">Read</button>
                <button id="btnUpdate" class="btn btn-default">Update</button>
              </div>

        <!--TABLE-->
        <center>
        <table id="fileTable">
            <thead>
                <tr>
                    <th>Content</th>
                    <th>Delete</th>
                </tr>
            </thead>
            <tbody id="fileTableBody"></tbody>
        </table>
        </center>
    </main>

    <!-- JAVASCRIPT -->
    <script src="crud.js"></script>
</body>
     </html>
```

- **crud.js**

```
const { app, BrowserWindow } = require('electron');
const fs = require('fs');
const path = require('path');
```

```
// DOM elements
var btnAppend = document.getElementById('btnCreate');
var btnRead = document.getElementById('btnRead');
var btnUpdate = document.getElementById('btnUpdate');
var btnDelete = document.getElementById('btnDelete');
var fileName = document.getElementById('fileName');
var fileContents = document.getElementById('fileContents');

let pathName = path.join(__dirname, 'Files');

//Create button
btnCreate.addEventListener('click', function(){

    // CreatE text file
    let file = path.join(pathName, fileName.value);
    let contents = fileContents.value;
    fs.writeFile(file, contents, function(err){
        if(err){
            return console.log(err);
        }
        var txtfile = document.getElementById("fileName").value;
        alert(txtfile + " text file was created");
        console.log("The file was created");
    });
});

//Read button
btnRead.addEventListener('click', function() {
    let file = path.join(pathName, fileName.value);
```

```
    fs.readFile(file, 'utf8', function(err, data) {
        if (err) {
            console.error(err);
            return console.log(err);
        }
        fileContents.value = data;

        const lines = data.split('\n');
        const tableBody = document.getElementById('fileTableBody');
        tableBody.innerHTML = '';

        lines.forEach(function(line) {
            addContentToTable(line);
        });

        console.log("File has been Read!");
    });
});

//Update button
btnUpdate.addEventListener('click', function () {
    let file = path.join(pathName, fileName.value);
    let contents = fileContents.value;

    fs.writeFile(file, contents, function (err) {
        if (err) {
            return console.log(err);
        }
```

```
        var txtfile = document.getElementById("fileName").value;
        console.log("The file was updated!");
        alert(txtfile + " text file was updated");
    });
});

//Delete button
btnDelete.addEventListener('click', function(){
    let file = path.join(pathName, fileName.value);

    fs.unlink(file, function(err){
        if(err){
            return console.log(err);
        }
        fileName.value = "";
        fileContents.value = "";
        console.log("The file was deleted!");
    });
});

//add content to the table
function addContentToTable(content) {
    const newRow = document.createElement('tr');

    const contentCell = document.createElement('td');
    contentCell.textContent = content;

    const deleteCell = document.createElement('td');
    const deleteButton = document.createElement('button');
```

```
deleteButton.textContent = 'Delete';

deleteButton.addEventListener('click', function() {
    const row = this.parentNode.parentNode;
    const rowIndex = row.rowIndex;

    row.remove();

    const file = path.join(pathName, fileName.value);
    fs.readFile(file, 'utf8', function(err, data) {
        if (err) {
            console.error(err);
            return console.log(err);
        }
        const lines = data.split('\n');
        lines.splice(rowIndex - 1, 1);
        const updatedData = lines.join('\n');
        fs.writeFile(file, updatedData, 'utf8', function(err) {
            if (err) {
                console.error(err);
            }
        });
    });
});

deleteCell.appendChild(deleteButton);

newRow.appendChild(contentCell);
newRow.appendChild(deleteCell);
```

```
    const tableBody = document.getElementById('fileTableBody');
    tableBody.appendChild(newRow);
     }
```

- **index.js**

```
const { app, BrowserWindow } = require('electron');
const fs = require('fs')
const path = require('path');

// Handle creating/removing shortcuts on Windows when installing/uninstalling.
if (require('electron-squirrel-startup')) {
  app.quit();
}

const createWindow = () => {
  // Create the browser window.
  const mainWindow = new BrowserWindow({
    width: 800,
    height: 600,
    webPreferences: {
      nodeIntegration: true,
      contextIsolation: false,
      preload: path.join(__dirname, 'preload.js'),
    },
  });

  // and load the index.html of the app.
  mainWindow.loadFile(path.join(__dirname, 'home.html'));
```

```
  // Open the DevTools.
  //mainWindow.webContents.openDevTools();
};

// This method will be called when Electron has finished
// initialization and is ready to create browser windows.
// Some APIs can only be used after this event occurs.
app.on('ready', createWindow);

// Quit when all windows are closed, except on macOS. There, it's common
// for applications and their menu bar to stay active until the user quits
// explicitly with Cmd + Q.
app.on('window-all-closed', () => {
  if (process.platform !== 'darwin') {
    app.quit();
  }
});

app.on('activate', () => {
  // On OS X it's common to re-create a window in the app when the
  // dock icon is clicked and there are no other windows open.
  if (BrowserWindow.getAllWindows().length === 0) {
    createWindow();
  }
});

// In this file you can include the rest of your app's specific main process
     // code. You can also put them in separate files and import them here.
```

- **preload.js**

```
// See the Electron documentation for details on how to use preload scripts:
// https://www.electronjs.org/docs/latest/tutorial/process-model#preload-scripts
```

- **Link:** **https://github.com/hanisnbhh/makeupBox.git**