KOLEJ PROFESIONAL MARA

BERANANGDIPLOMA 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:
   ConsolasSize: 10
   pts
   Line Spacing: Single

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

Personal Details	
Name	NUR HADIRAH BINTI KHAIRIAFFENDI
I/D Number	BCS2207-054
Class	DCS4A
Lecturer	PUAN NINI ANIZA

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%**

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:
    a. Apply GUI elements that assist users in using application.
    b. The application's 'look and feel' is attractive and informative.
5. Produce a report on your application functionalities and features. Include the following:
    a. Overview of your application with a brief description.

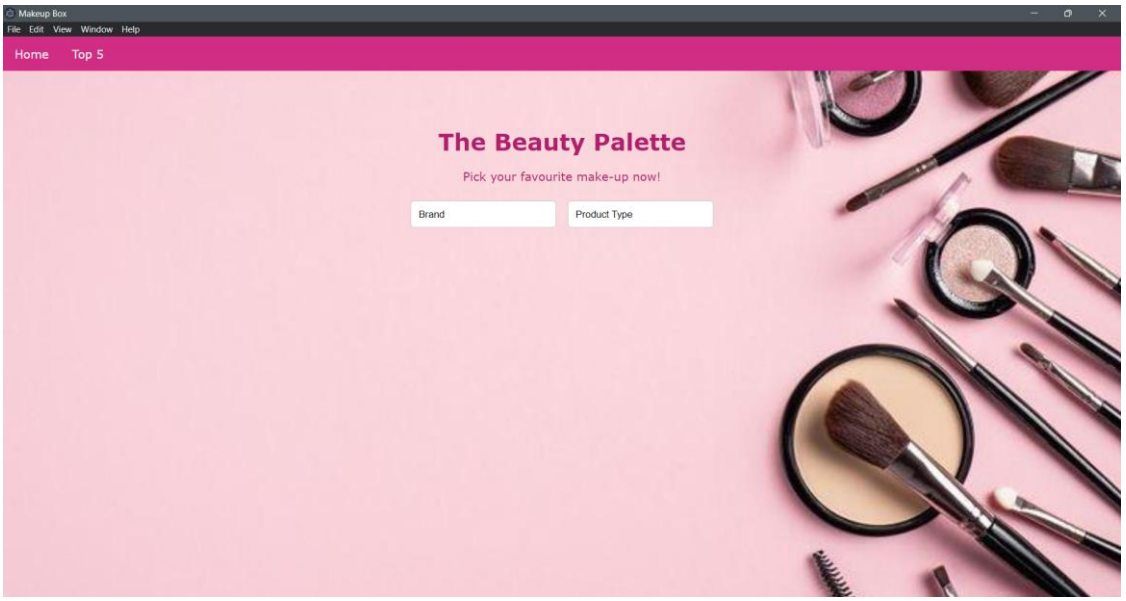
For those who enjoy cosmetics, the cosmetics Desktop Application is a feature-rich tool that offers convenient access to a large selection of makeup items. Users can get comprehensive data on a wide variety of beauty products through the application's integration with a robust API (http://makeup-api.herokuapp.com/api/v1/products.json). Users are able to peruse and investigate a vast assortment of makeup products that include different brands, kinds, and product categories.

This application's primary goal is to retrieve makeup products from the given API, enabling users to explore, search, and obtain details about various makeup products. Additionally, customers can keep a customised list of their top 5 preferred makeup companies and create, edit, delete, and modify it. This customised list provides a quick and easy method for customers to select their favourite cosmetics.

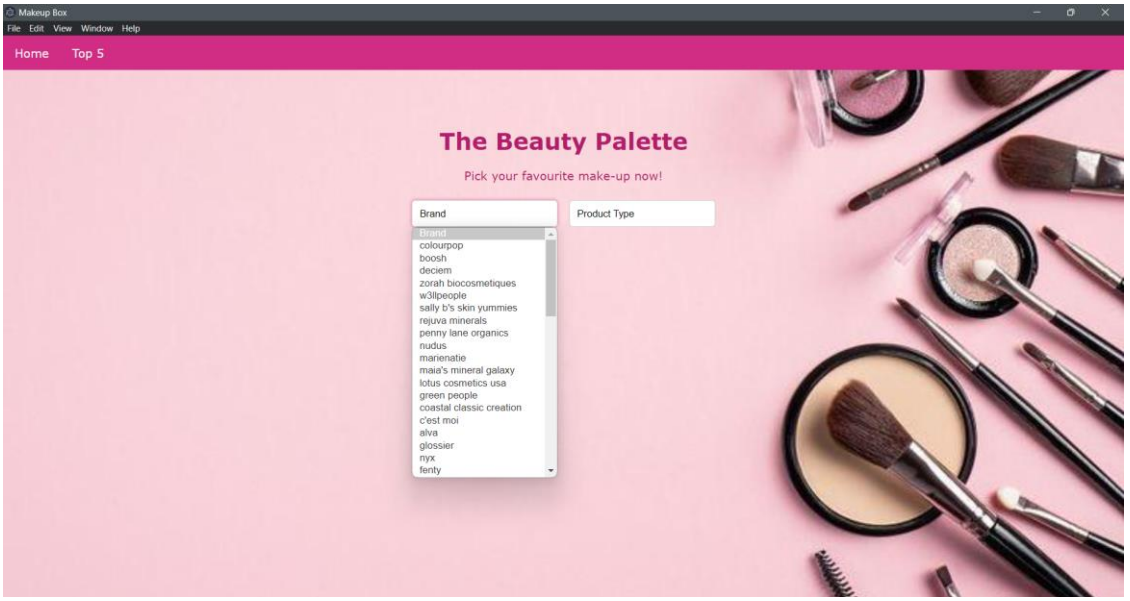
The application gives users the ability to efficiently manage their preferences by making it easier for them to Read, Create, Update, and Delete (CRUD) their top 5 makeup brands. The app's intuitive layout makes it easier for users to engage with the data on cosmetic products and offers a smooth way for them to discover, arrange, and keep track of their preferred makeup brands.

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

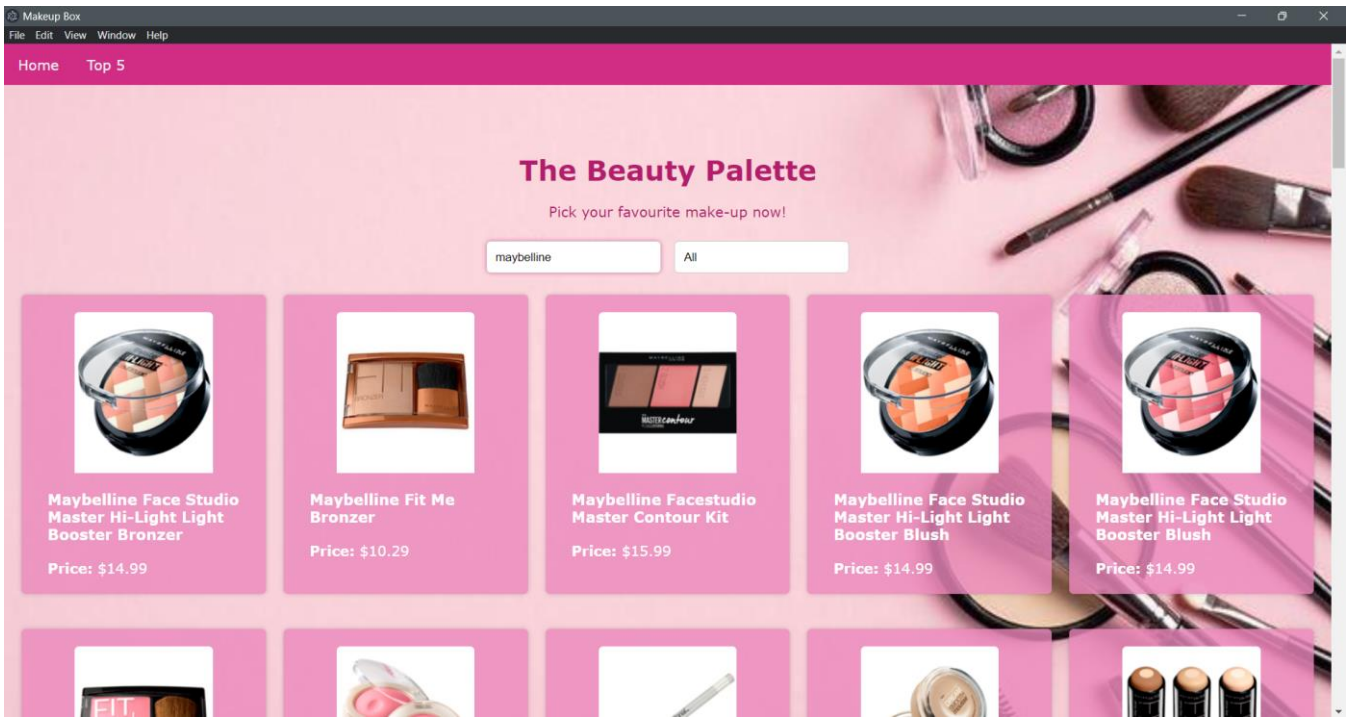
Index.html



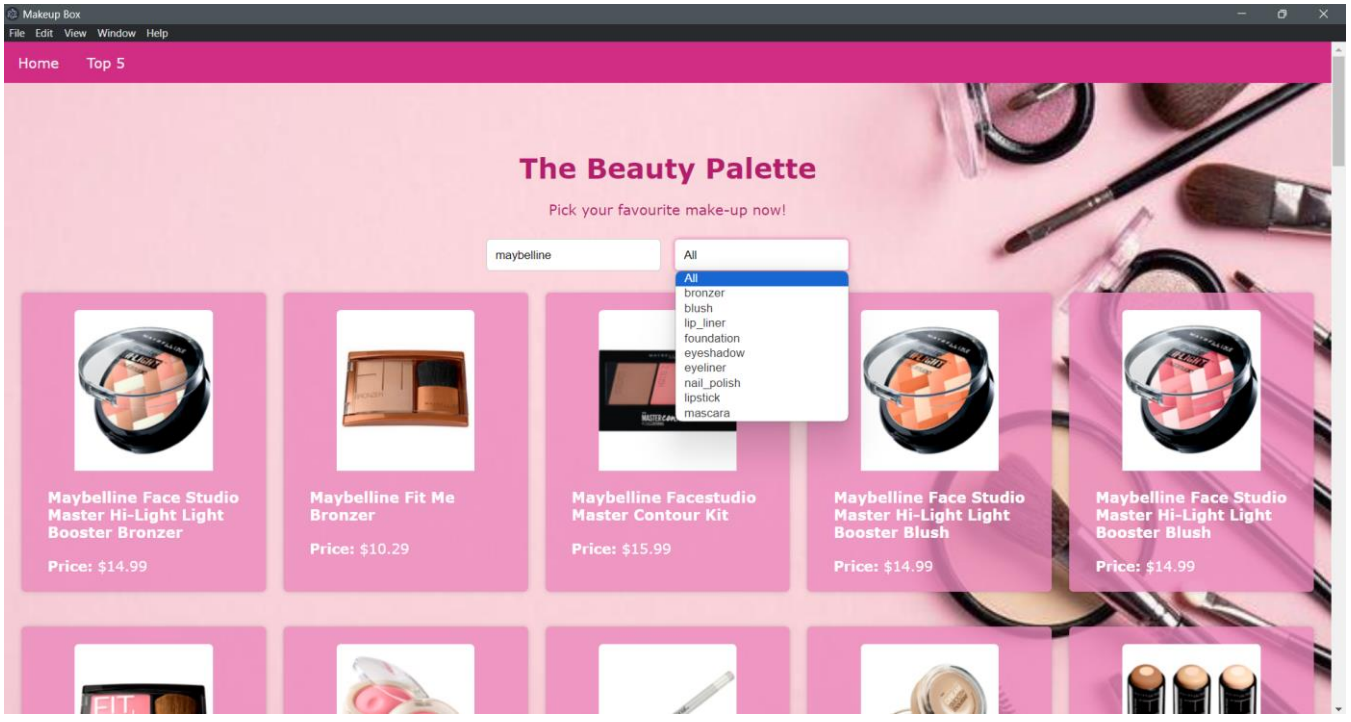
This is the homepage of my desktop application. At the header of my homepage, it has a navigation bar that is “Home” and “Top 5”. When user click “Home” this homepage will be displayed. When user click “Top 5” it will display (curd.html) page. The 2 drop-down function is to fetchAPI. The first drop-down is to fetch Brand from the API and the second drop-down is to fetch Product Type.



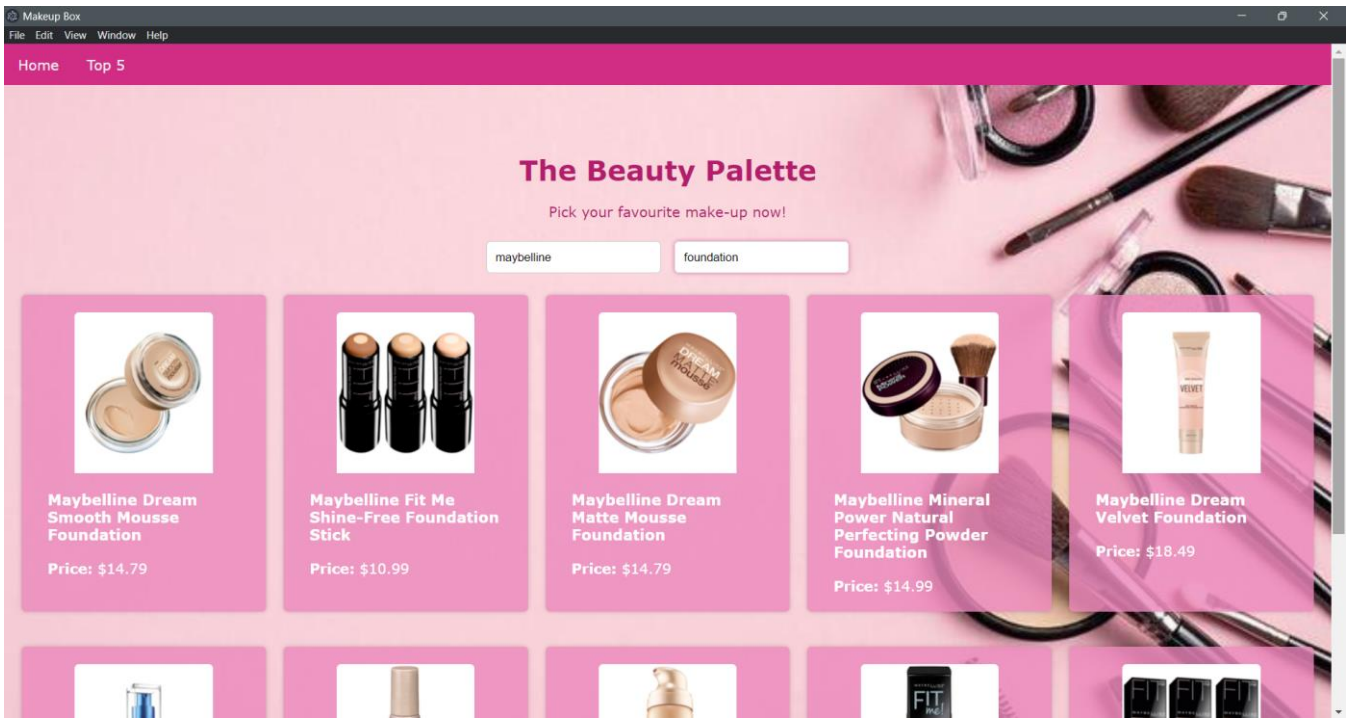
When user click the Brand drop-down, it will displayed all the Brand that already fetch from API. User need to select Brand.



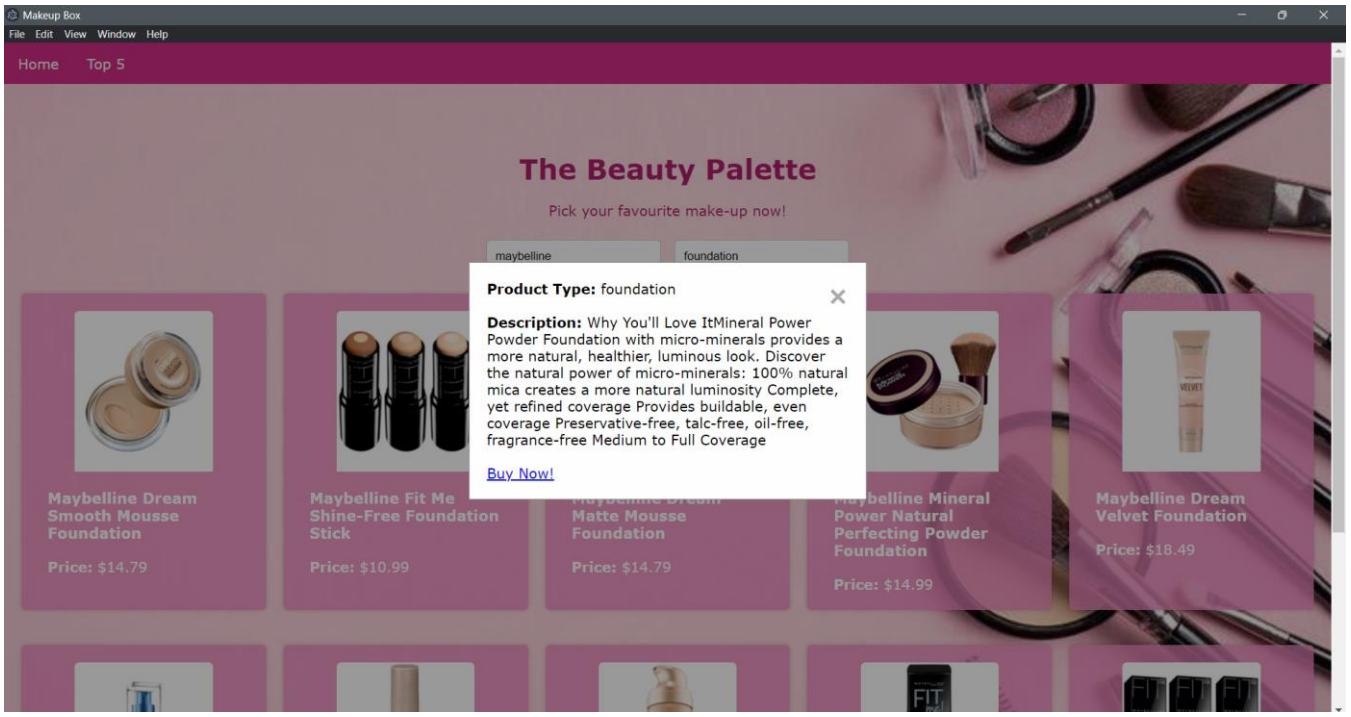
User choose brand Maybelline. Then the page will fetch all the make-up product from Maybelline.



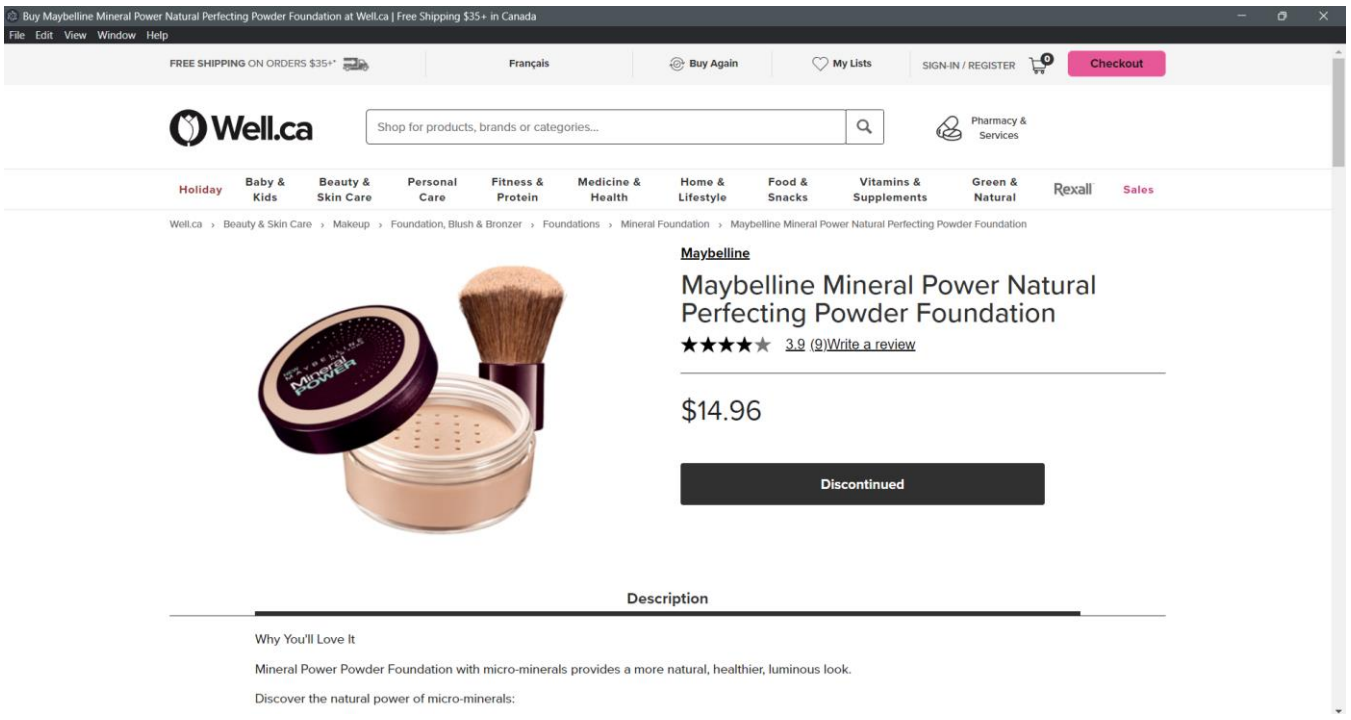
At the product type drop-down, user can fetch the product type based on Maybelline brand. User need to choose what type of product they want to fetch.



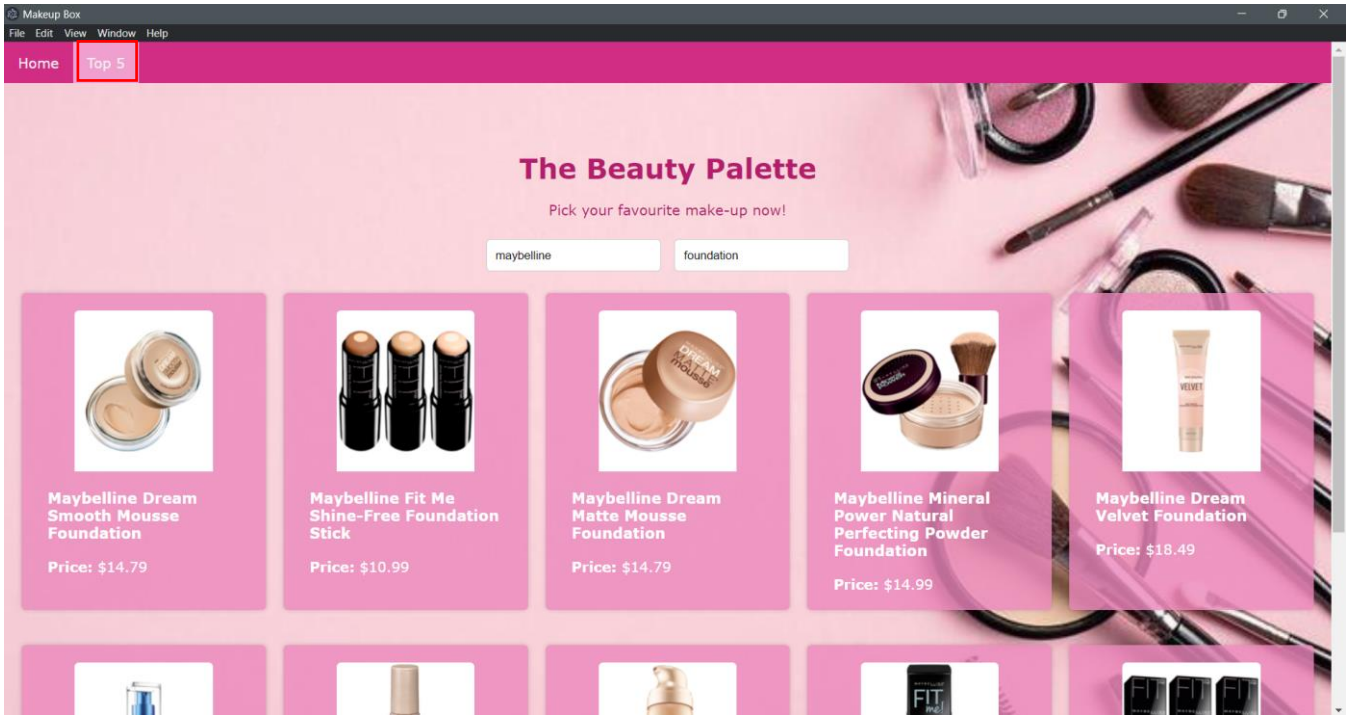
User choose product type foundation. This page will display all the foundation from brand Maybelline. The container only displayed product name and product price. But, when user click the image, it will display product type, product description and product link.



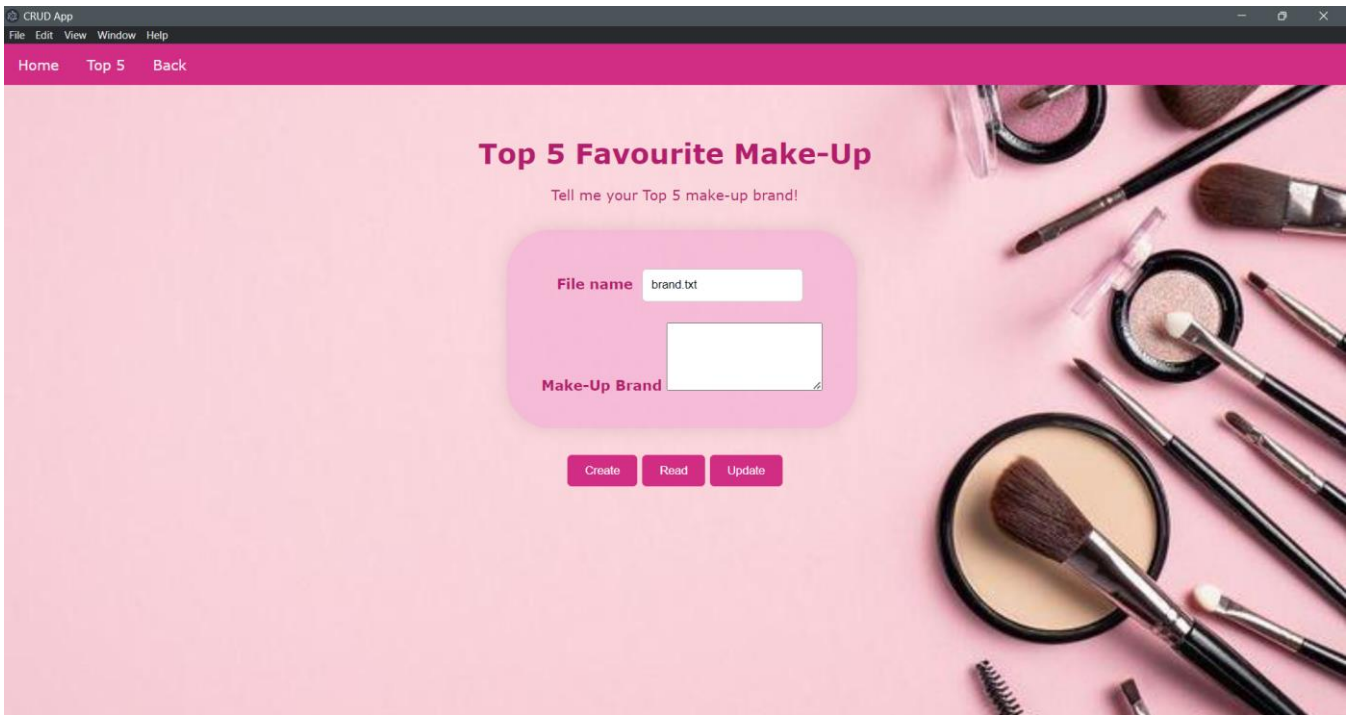
User click image at product ‘Maybelline Mineral Power Natural Perfecting Powder Foundation’. The product type, product description and product price already displayed. User can click to the ‘But Now!’ button if user want to buy what product.



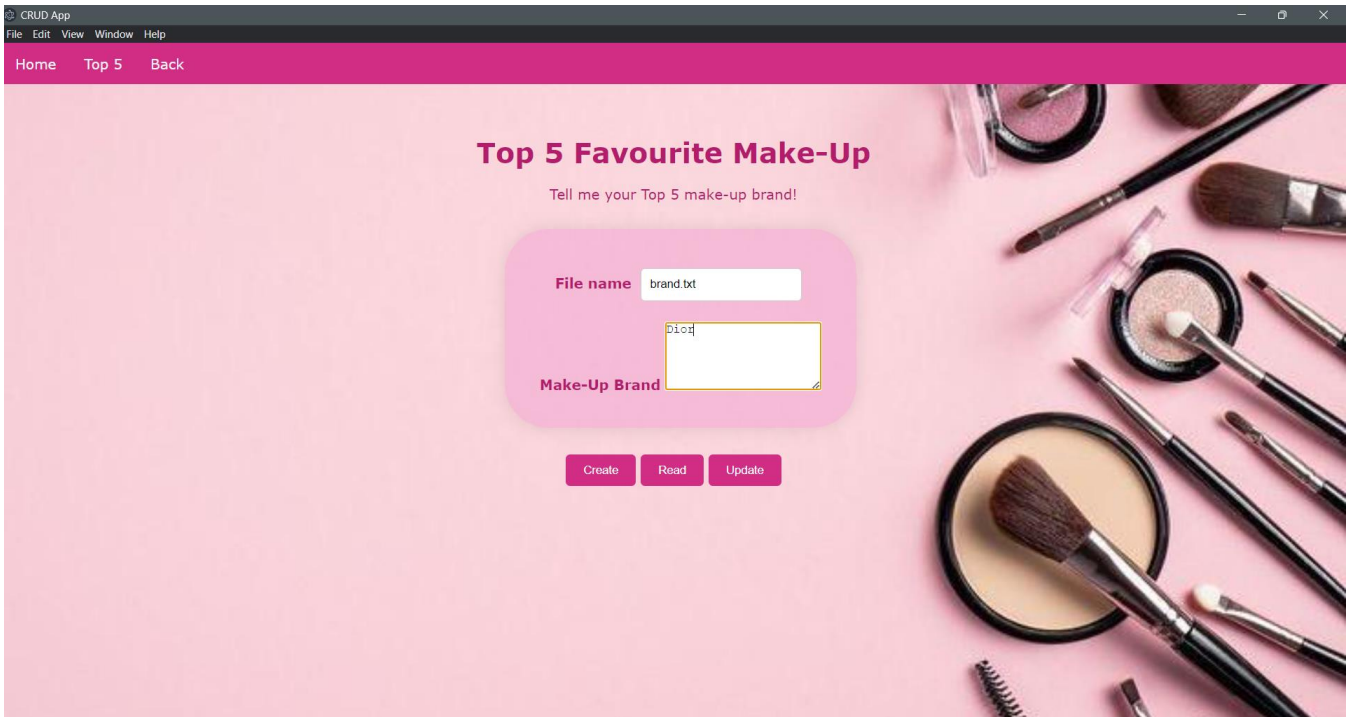
When user click ‘Buy Now!’ link, it will displayed the Maybelline website and user can buy the product they have selected.



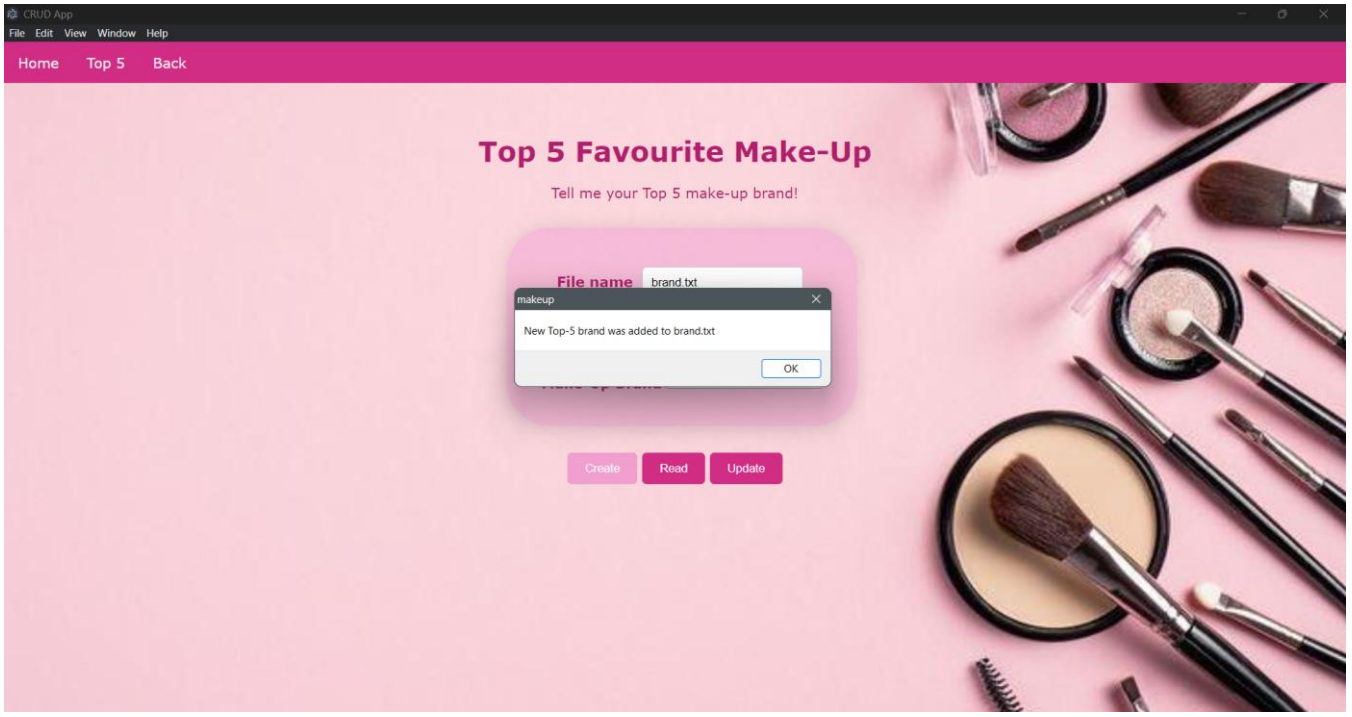
User click the “Top 5” on the navigation bar.



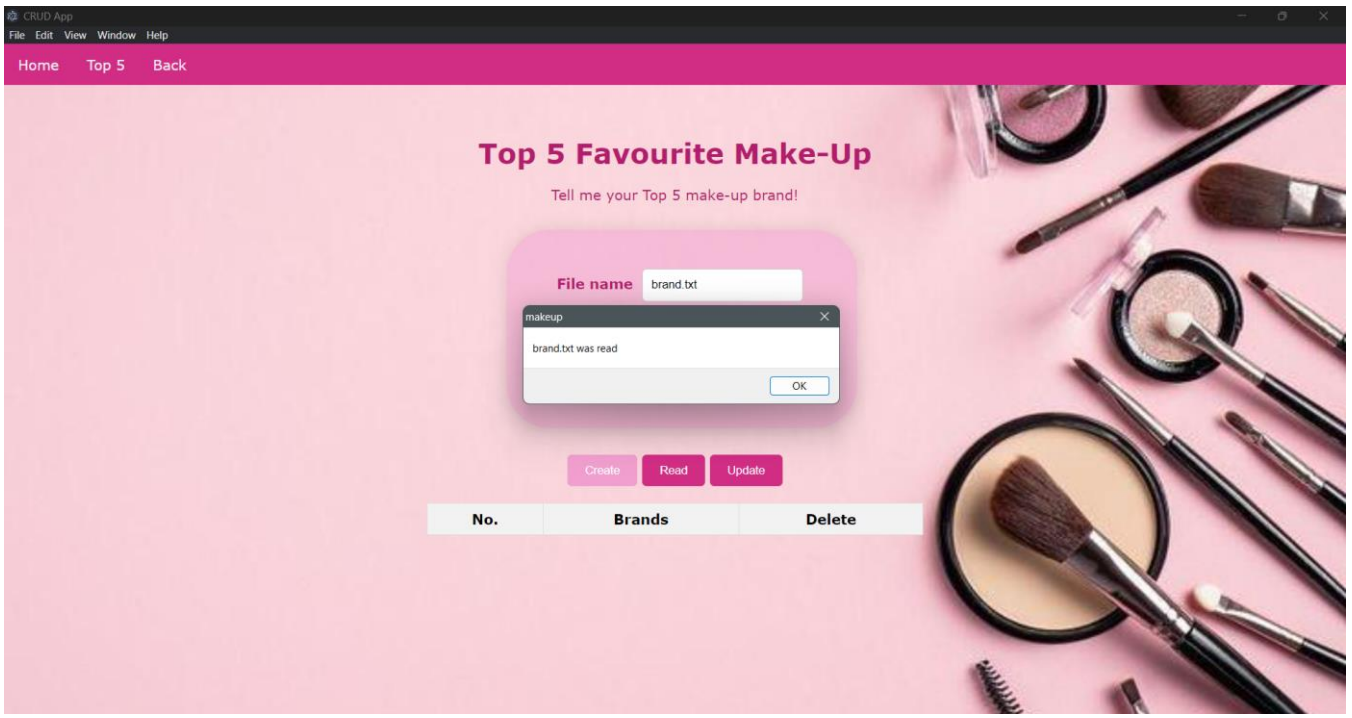
This page will be displayed. On this page, user is able to create, read, update, and delete their Top 5 make-up that they like. The file name already fixed so user no need to fill in the form. At the make-up brand form, user can enter their Top 5 brand. When user click “Back” button, it will return user to the homepage.



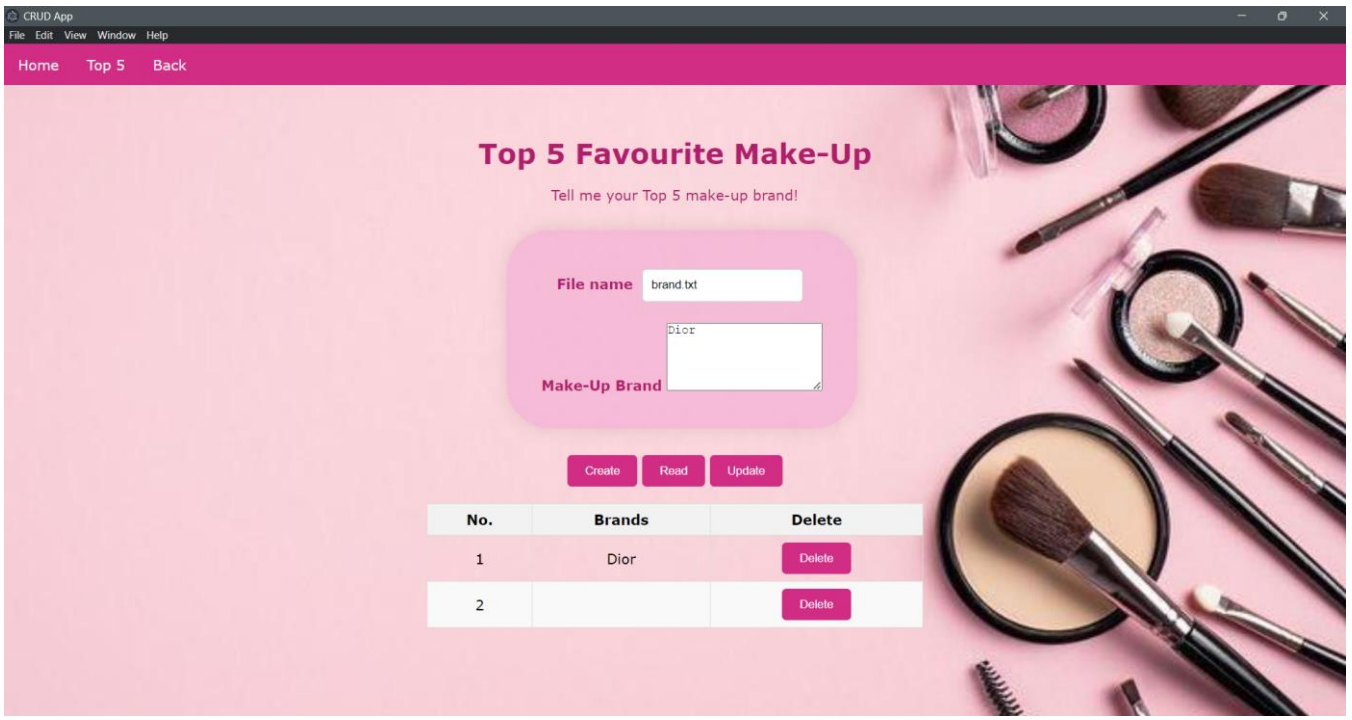
User need to enter the brand. Example, user enter ‘Dior’. Then, user need to click Create button to add the make-up brand in the brand.txt file.



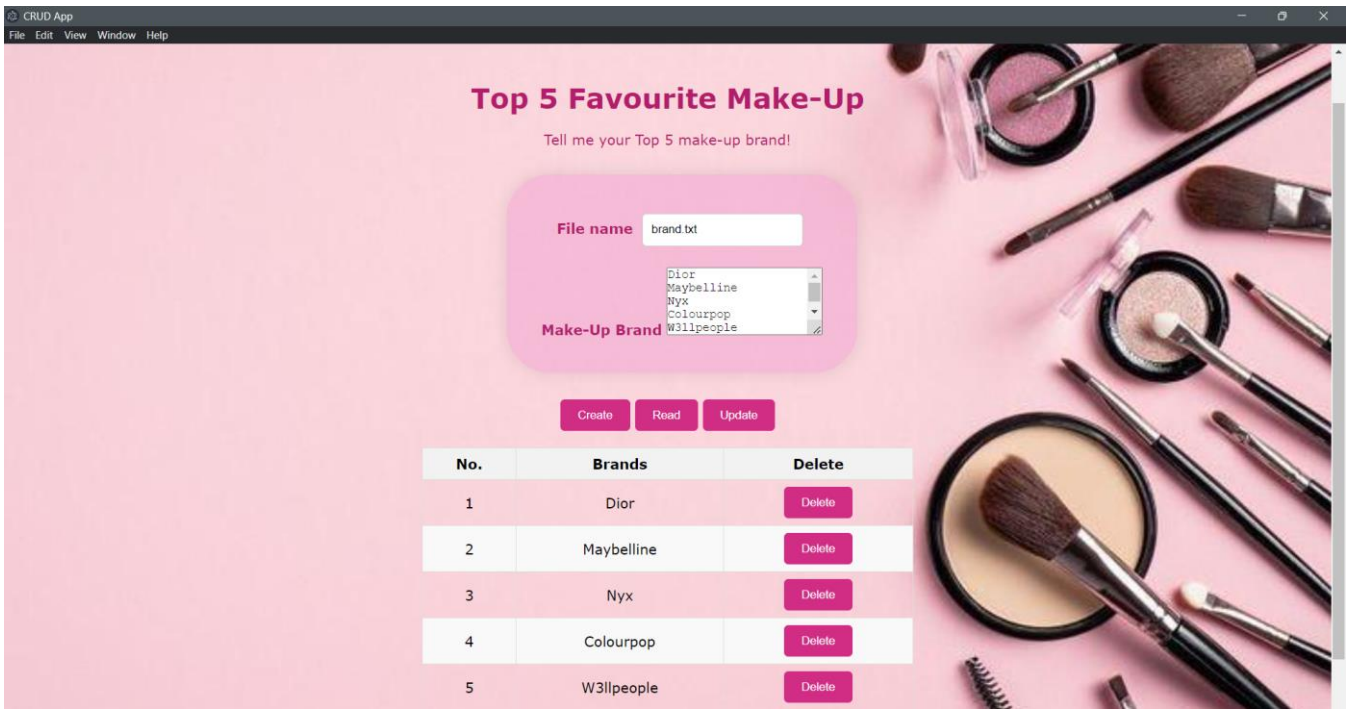
When user click create button, this pop up will displayed automatically. User just need to click OK button.



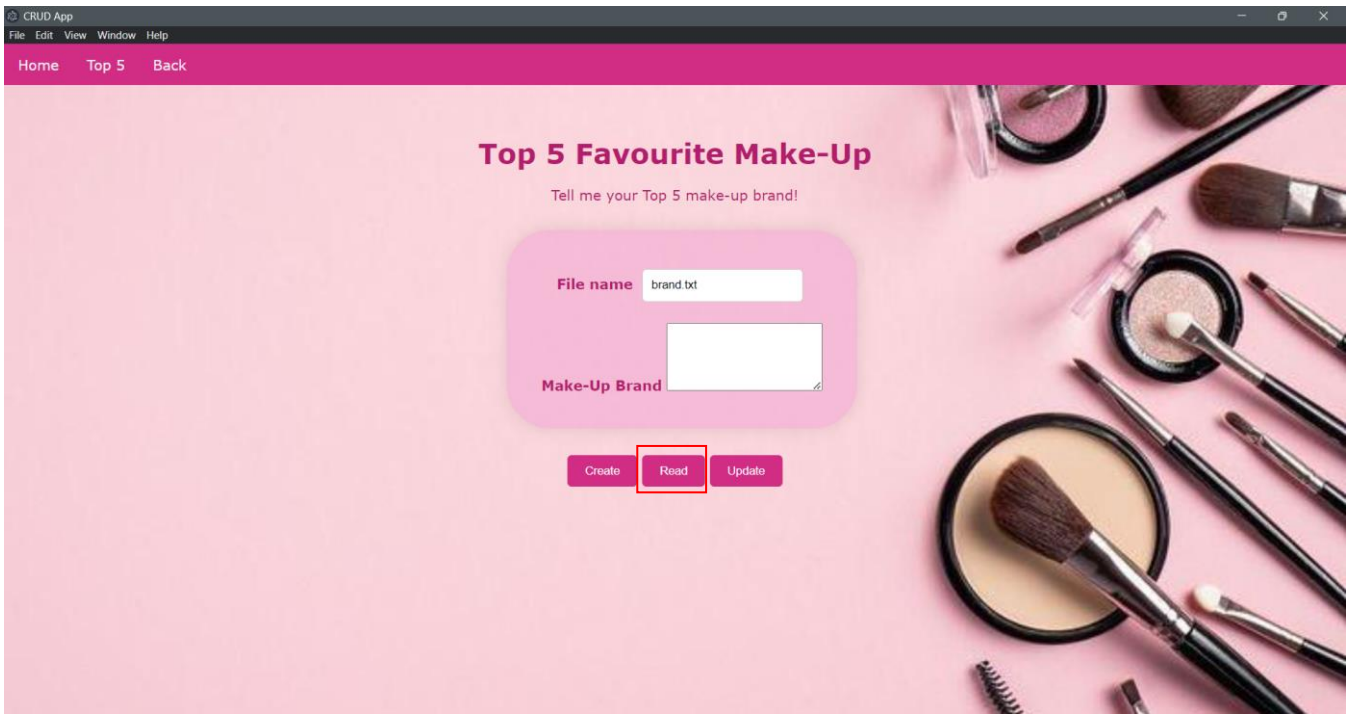
When user click OK, this pop up will be displayed too.



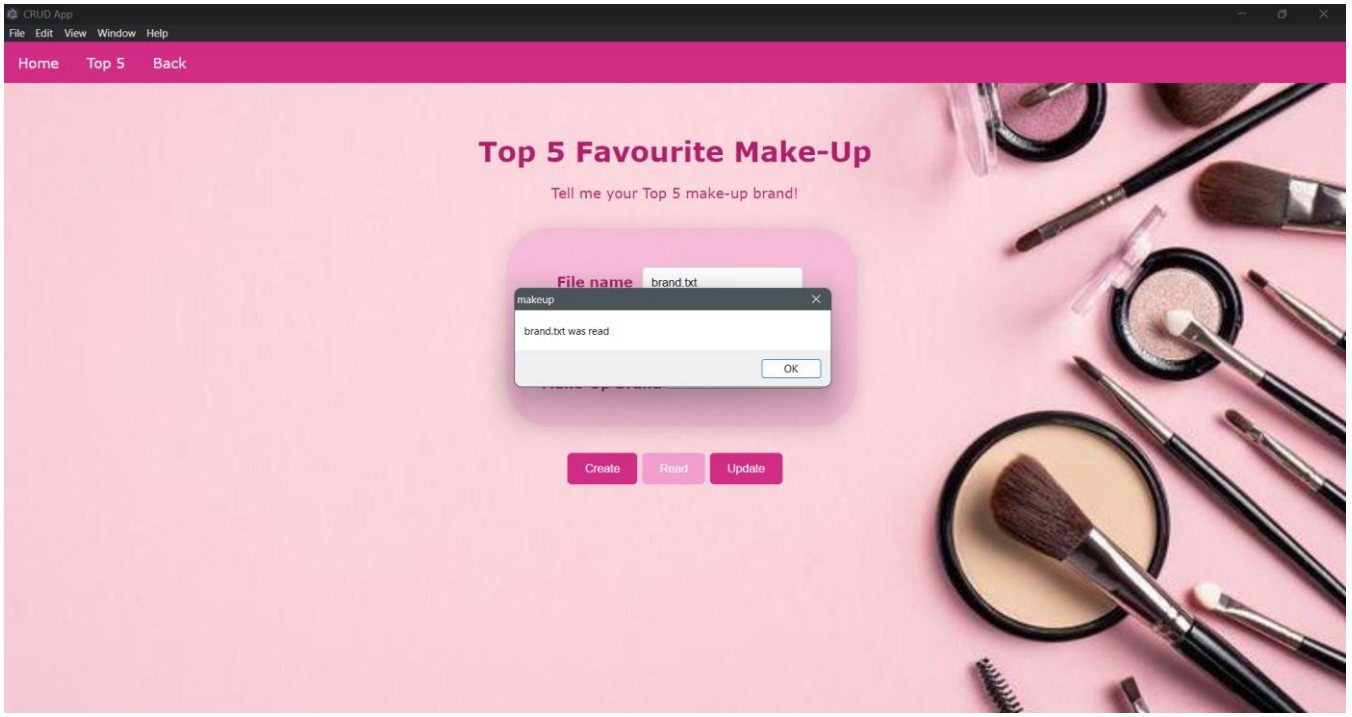
And then, when the make-up brand successfully added to brand.txt file, it will displayed in the table. If user want to add another make-up brand, user just need to repeat the same step.



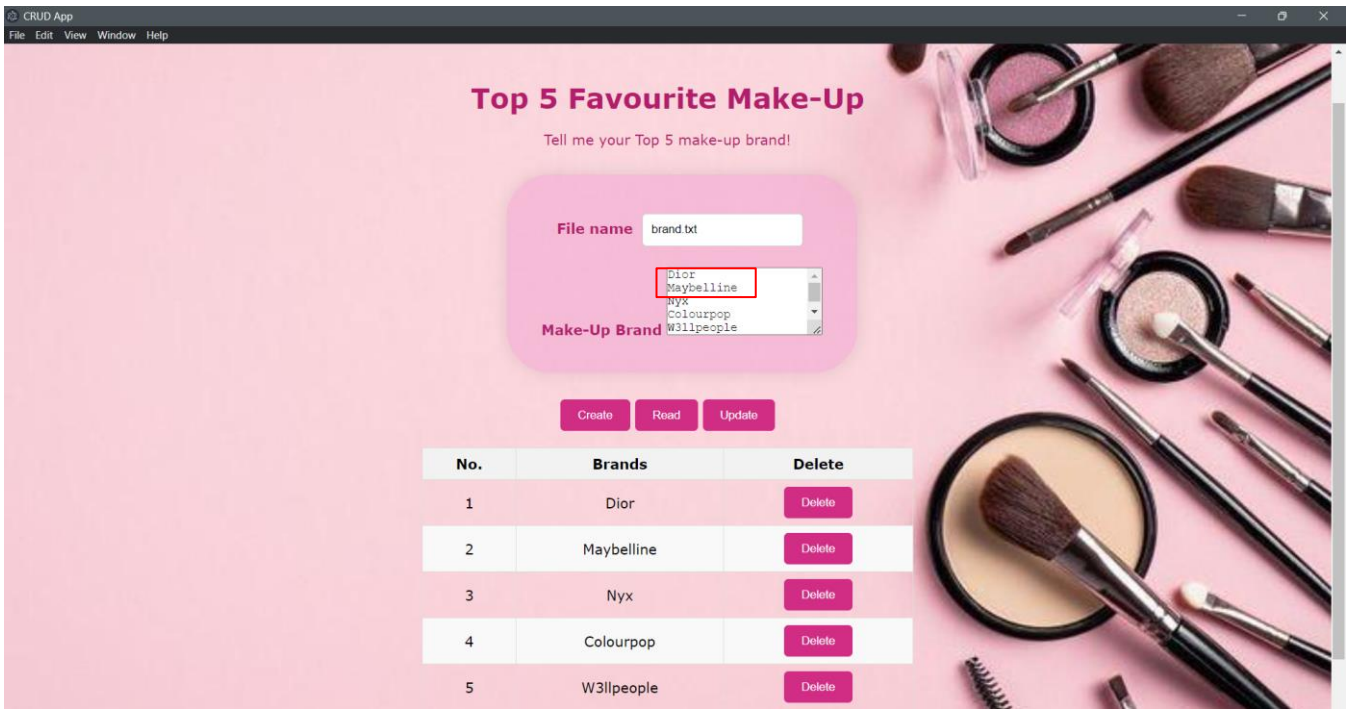
Example, user already enter their all Top 5 make-up brand.



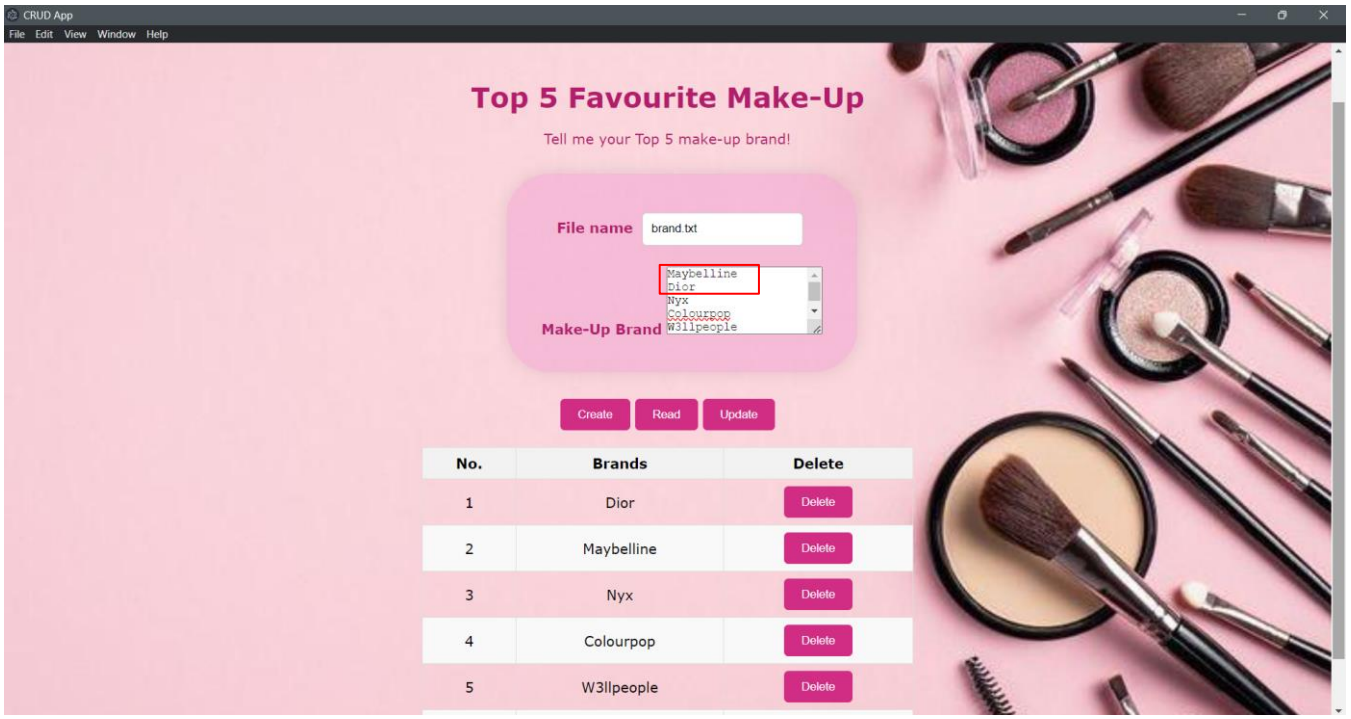
If user want to update the make-up brand that they already entered, user need to click button Read first.



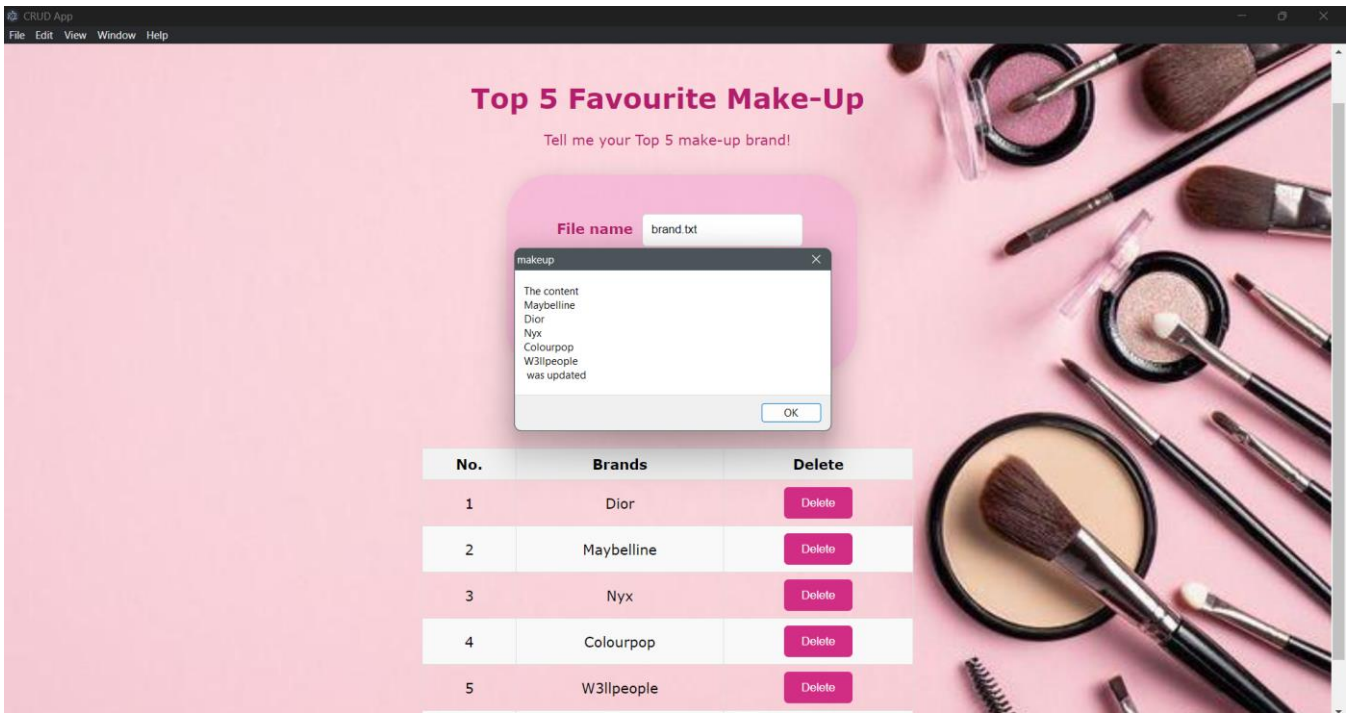
When user click read button, this pop up will automatically displayed. User just need to click OK button.



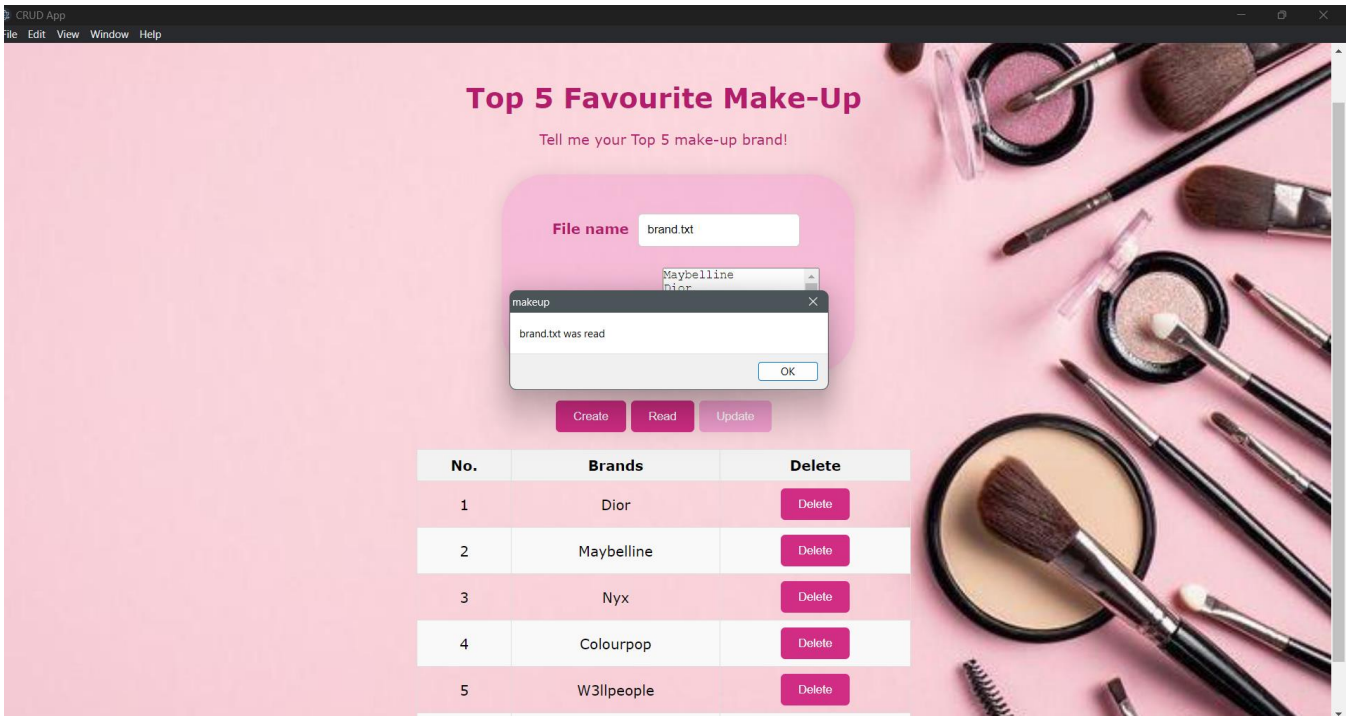
Then it will read all the brand from brand.txt file. Now user can update the brand.



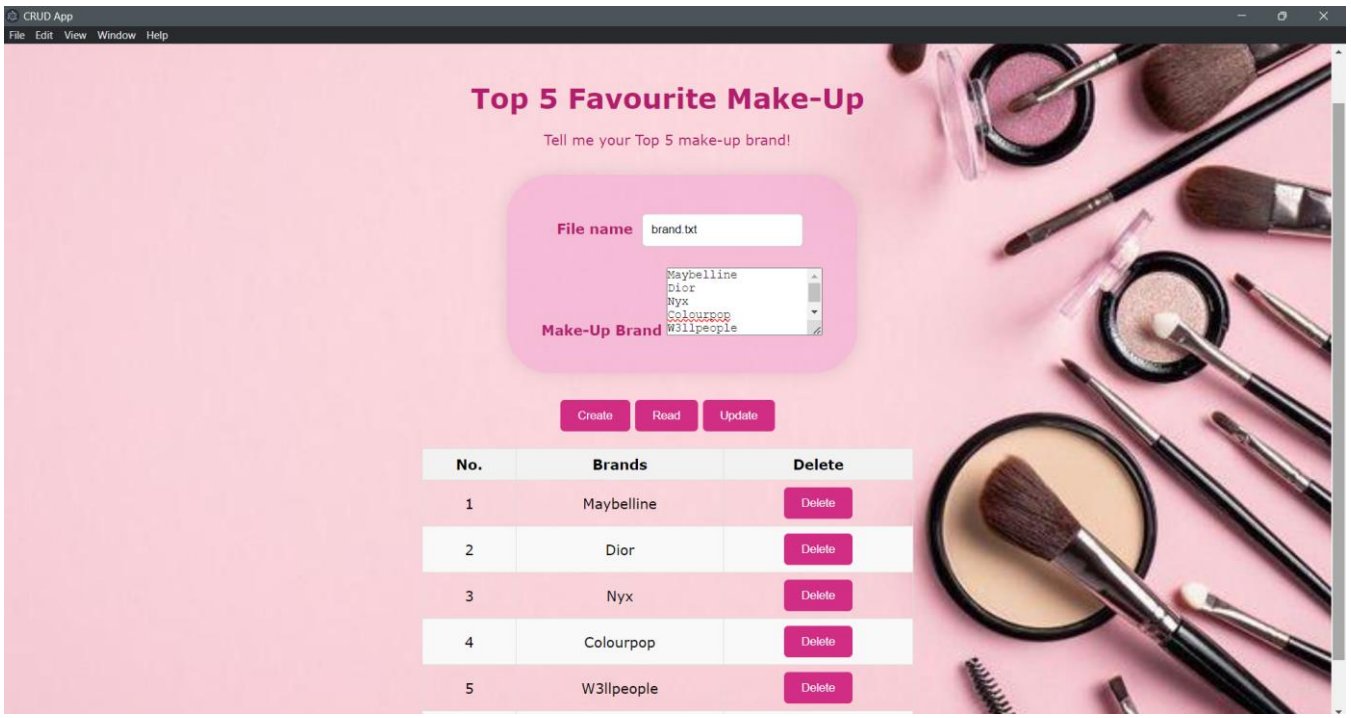
Example, user's has change their mind, they think they like Maybelline more than they like Dior. Then, user need to click update button to update the content.



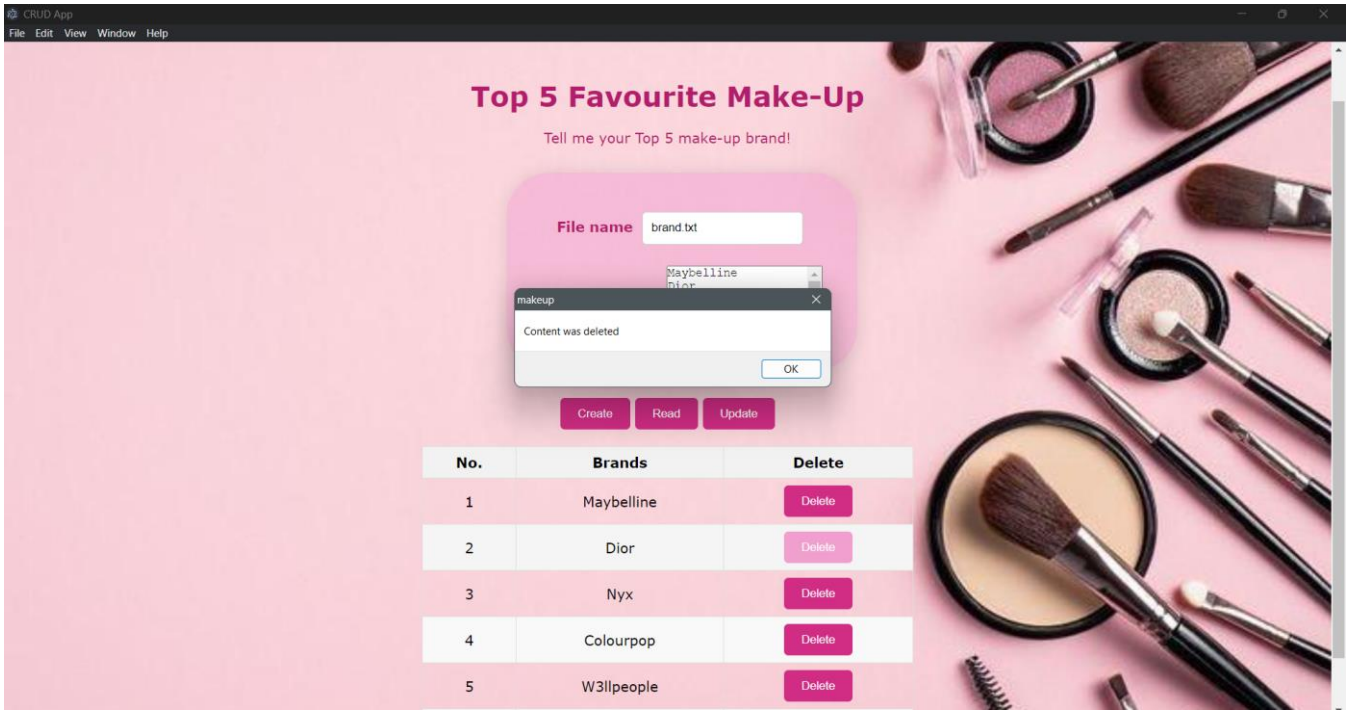
Then, this pop up will be displayed. User just need to click OK.



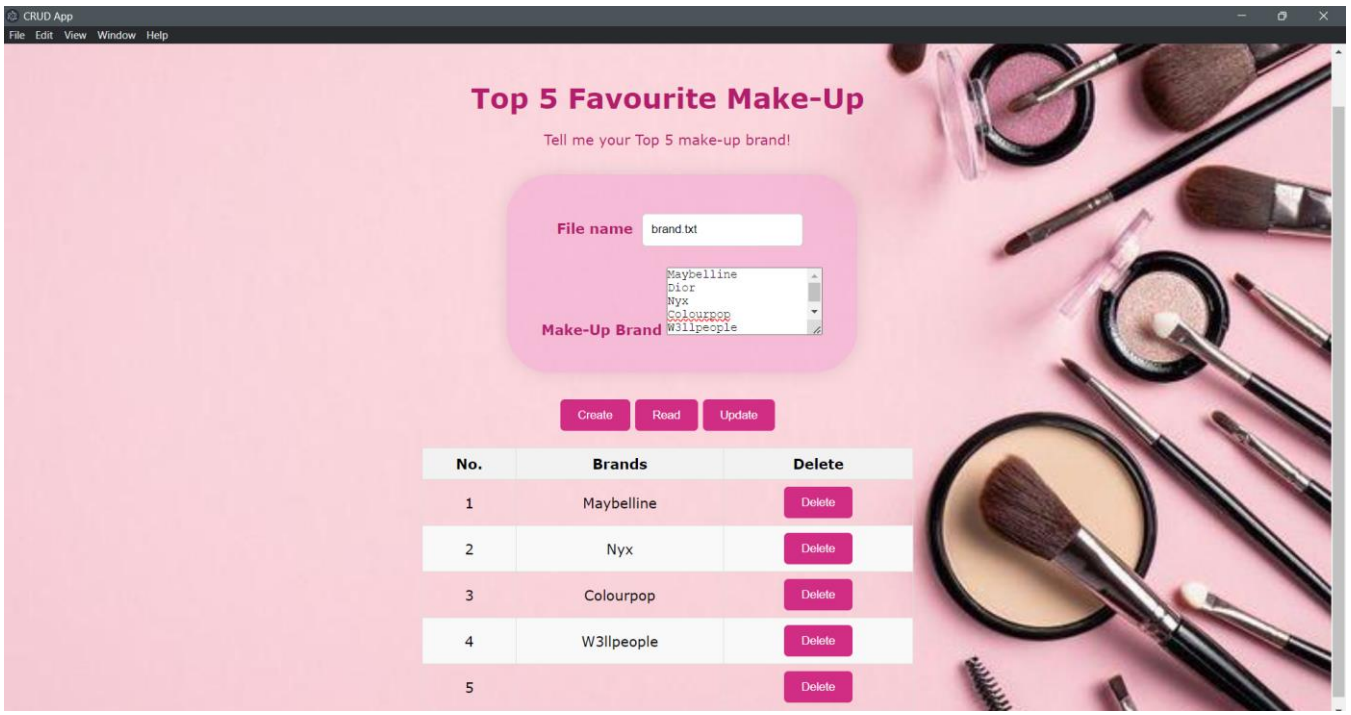
This pop up will be displayed after user click Ok button. User just need to click Ok button again.



Then the make-up brand will be updated. If user want to delete the brand, user can click to the delete button.



User want to delete Dior brand. User click to the delete button then this pop up will be displayed. User just need to click OK button.



The content Dior was deleted.

c. Program codes of your system

crud.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8" />
  <title>CRUD App</title>
  <link rel="stylesheet" href="index.css" />
  <link rel="stylesheet" href="photon.min.css" />
</head>
<body>
  <div class="navbar">
    <a href="index.html">Home</a> <!--navigation bar menu-->
    <a href="crud.html">Top 5</a>
    <a href="javascript:history.go(-1)">Back</a>
  </div>
<div class="mainWrapper">
    <br><br>
    <h1>Top 5 Favourite Make-Up</h1>
    <br>
    <center>
    <div class="desc">Tell me your Top 5 make-up brand!</div>
```

```
      <form>
  <div class="containerCRUD">
        <div class="form-group"> <!--form for file name and make-up brand-->
          <b><label>File name</label></b>
          <input id="fileName" type="text" class="form-control" readonly
     value="brand.txt">
        </div>
        <br>
        <div class="form-group">
          <b><label>Make-Up Brand</label></b>
          <textarea id="fileContents" class="form-control" rows="5"></textarea>
        </div>
      </form>
  </div>
      <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
      <!--button for create update read-->
      <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>
      <br><br>
      <table id="fileTable" class="table" style="display: none;">
        <thead>
          <!--table to display top 5 brand-->
          <tr>
            <th>No.</th>
            <th>Contents</th>
            <th>Action</th>
          </tr>
        </thead>
        <tbody id="fileTableBody">
          <!-- Table rows will be added dynamically here -->
        </tbody>
      </table>
    </center>
</div>
  <script src="crud.js"></script>
</body>
     </html>
```

crud.js

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

/*untuk get element*/
var fileNameInput = document.getElementById('fileName');
var btnCreate = document.getElementById('btnCreate');
var fileContents = document.getElementById('fileContents');
var fileTable = document.getElementById('fileTable');
var fileTableBody = document.getElementById('fileTableBody');

let pathName = path.join(__dirname, 'Files');
let currentFileNumber = 1; // Initialize the file number counter

function updateTable() {
  const rows = fileTableBody.getElementsByTagName('tr');
  currentFileNumber = 1; // Reset the file number counter
  for (let i = 0; i < rows.length; i++) {
    rows[i].getElementsByTagName('td')[0].textContent = currentFileNumber++;
  }
}

/* untuk add row kt table yg display brand*/
function addRowToTable(fileName, fileContent) {
  var contentLines = fileContent.split('\n');
  var startNumber = currentFileNumber; // Store the initial file number

  contentLines.forEach(content => {
    var newRow = document.createElement('tr');
    var fileNumberCell = document.createElement('td');
    fileNumberCell.textContent = currentFileNumber++; // Set the file number and
     increment

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

    var actionCell = document.createElement('td');

    /*untuk delete content*/
    var deleteButton = document.createElement('button');
```

```
    deleteButton.textContent = 'Delete';
    deleteButton.addEventListener('click', function() {
      var contentToDelete = contentCell.textContent;
      var updatedContent = fileContent.replace(`${contentToDelete}\n`, '');
      fs.writeFile(path.join(pathName, fileName), updatedContent, function(err) {
        if (err) {
          return console.log(err);
        }
        newRow.remove();
        alert('Content was deleted'); /*pop up untuk kalau dia dh n=berjaya delete
     content*/
        console.log('The content was deleted!');
        updateTable(); // Update the table after deleting content
      });
    });

    actionCell.appendChild(deleteButton);

    newRow.appendChild(fileNumberCell);
    newRow.appendChild(contentCell);
    newRow.appendChild(actionCell);

    fileTableBody.appendChild(newRow);
  });

  currentFileNumber = startNumber + 1; // Adjust the file number after adding
     content
  updateTable(); // Update the table after adding content
}

/*untuk clear table*/
function clearTable() {
  fileTableBody.innerHTML = '';
}

/* untuk display table yg ada brand*/
function showTable() {
  fileTable.style.display = 'table';
}
```

```
/*untuk display file*/
function displayFileContents() {
  let fileName = fileNameInput.value.trim();
  let file = path.join(pathName, fileName);

  fs.readFile(file, 'utf8', function(err, data) {
    if (err) {
      return console.log(err);
    }
    fileContents.value = data;
    alert(`${fileName} was read`);
    console.log('The file was read!');
    clearTable();
    addRowToTable(fileName, data);
    showTable(); // Show the table after reading content
  });
}

/*untuk create content masuk dalam brand.txt file*/
btnCreate.addEventListener('click', function() {
  let fileName = fileNameInput.value.trim();
  let file = path.join(pathName, fileName);
  let contents = fileContents.value;

  fs.appendFile(file, contents + '\n', function(err) {
    if (err) {
      return console.log(err);
    }

    alert(`New Top-5 brand was added to ${fileName}`);
    console.log('New content was added');
    fileContents.value = '';

    displayFileContents();
    showTable(); // Show the table after creating content
  });
});

/* untuk button read*/
btnRead.addEventListener('click', function() {
```

```
  displayFileContents();
});

/*untuk button update*/
btnUpdate.addEventListener('click', function() {
  let fileName = fileNameInput.value.trim();
  let file = path.join(pathName, fileName);
  let contents = fileContents.value;

  fs.writeFile(file, contents, function(err) {
    if (err) {
      return console.log(err);
    }
    alert(`The content \n${contents} was updated`);
    console.log('The file was updated');

    displayFileContents();
  });
     });
```

fetchapi.js

```
// Fetch the brands and populate the dropdown
function fetchBrands() {
  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');

      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(`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(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 productDetailsDiv = document.createElement('div');

    const imageDiv = document.createElement('div');
    if (product.image_link) {
      const img = document.createElement('img');
      img.src = product.image_link;
      img.dataset.description = product.description || '';
      img.dataset.productLink = product.product_link || '';
      img.dataset.productType = product.product_type || 'Product type not
     available';
      img.style.maxWidth = '200px';
      img.classList.add('productImage');
      imageDiv.appendChild(img);
    }
```

```
    productDetailsDiv.appendChild(imageDiv);

    const productNameDiv = document.createElement('div');
    productNameDiv.innerHTML = `<p><b>${product.name}</b></p> <br>`;
    productDetailsDiv.appendChild(productNameDiv);

    const priceDiv = document.createElement('div');
    if (product.price) {
      priceDiv.innerHTML = `<b>Price:</b> $${product.price}`;
    } else {
      priceDiv.innerHTML = `<b>Price:</b> Price not available`;
    }
    productDetailsDiv.appendChild(priceDiv);

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

// Open modal to display description, product link, and product type
document.addEventListener('click', function (event) {
  if (event.target.classList.contains('productImage')) {
    const description = event.target.dataset.description || 'Description not
     available';
    const productLink = event.target.dataset.productLink || 'Product link not
     available';
    const productType = event.target.dataset.productType || 'Product type not
     available';

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

// Function to display the modal with description, product link, and product type
function displayModal(description, productLink, productType) {
  const modal = document.getElementById('makeupModal');
  const descriptionSpan = document.getElementById('descriptionSpan');
  const productLinkSpan = document.getElementById('productLinkSpan');
  const productTypeSpan = document.getElementById('productTypeSpan');
```

```
        descriptionSpan.textContent = description;
        productLinkSpan.innerHTML = productLink !== 'Product link not available' ?
          `<a href="${productLink}" target="_blank">Buy Now!</a>` : 'Product link not
           available';
        productTypeSpan.textContent = productType || 'Product type not available';

        modal.style.display = 'block';
      }

      // Function to close the modal
      function closeModal() {
        document.getElementById('makeupModal').style.display = 'none';
      }

      // Fetch brands when the page loads
           fetchBrands();
```

index.css

```
/* Reset default margin and padding */
* {
  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-image: url('makeup.jpeg');
  background-size: cover;
  background-position: center;
  background-attachment: fixed;
}

/* Style for the select dropdowns */
select {
```

```
  position: relative;
  appearance: none;
  -webkit-appearance: none;
  -moz-appearance: none;
  padding: 10px;
  border: 1px solid #ccc;
  border-radius: 5px;
  margin: 5px;
  width: 200px;
  cursor: pointer;
  background: #fff;
  color: #000000;
}

/* Creating the arrow on the right */
select::after {
  content: '▼';
  position: absolute;
  top: 50%;
  right: 10px;
  transform: translateY(-50%);
  pointer-events: none;
}

/* Hover effect for the dropdown */
select:hover {
  border-color: #f19fcf;
}

/* Focus effect for the dropdown */
select:focus {
  outline: none;
  box-shadow: 0 0 5px rgba(210, 45, 133, 0.6);
}

/* Style the options within the dropdown */
select option {
  padding: 10px;
  color: #333;
  background: #fff;
```

```
}

/* New hover state for the options */
select option:hover {
  background-color: #f19fcf;
  color: #fff;
}

/* Style for the navigation bar */
.navbar {
  overflow: hidden;
  background-color: #d22d85;
  margin-bottom: 20px;
}

.navbar a {
  float: left;
  display: block;
  color: #f2f2f2;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
}

/* untuk tukar warna button tu bila cursor kena kt situ*/
.navbar a:hover {
  background-color: #f19fcf;
}

.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;
}

/* Style for the product list */
ul#productList {
  list-style: none;
  padding: 0;
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(250px, 1fr)); /* Dynamic grid layout */
  gap: 20px; /* Increased gap between grid items */
  margin-left: 20px;
  margin-right: 20px;
}

/* Update the product container styles */
ul#productList li {
  background-color: rgba(236, 136, 189, 0.8);
  border-radius: 5px;
  box-shadow: 0 0 5px rgba(0, 0, 0, 0.2);
  display: flex;
  align-items: center;
  flex-direction: column;
  transition: transform 0.3s ease; /* Smooth transition on hover */
  margin-bottom: 20px; /* Space between each product container */
  color: #ffffff;
  padding-left: 30px;
  padding-right: 30px;
  padding-bottom: 20px;
  padding-top: 20px;
}

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

/* Style for the "Search" button */
```

```
button {
  background-color: #d22d85; /* Dark pink button */
  color: #fff;
  padding: 10px 20px;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  transition: background-color 0.3s ease; /* Smooth color transition on hover */
}

button:hover {
  background-color: #f19fcf;
}

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

/* Style for the "Hello" heading */
h1 {
  font-family: Verdana, Geneva, Tahoma, sans-serif; /* Use the imported font */
  color: #b41d6e;
  text-align: center;
}

/* Style for the product description text */
h2 {
  color: #ffffff;
  font-weight: bold;
  text-align: center;
}

h4 {
  color: #c6468a;
}
```

```
p {
  color: #ffffff;
  font-weight: bold;
  font-size: 17px;
}

/* Style for the image */
img {
  max-width: 100%;
  border-radius: 5px;
  display: block; /* Ensures image is block-level */
  margin: auto; /* Center horizontally */
  padding-bottom: 20px;
}

/* Style for the product container on hover */
.product-container:hover {
  transform: scale(1.05); /* Enlarge on hover for a zoom effect */
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.3); /* Highlight shadow on hover */
}

/* Style for product container box */
.product-container {
  padding-left: 15px;
  padding-right: 15px;
  cursor: pointer; /* Add pointer cursor for interaction */
}

/* Smooth transition on image scaling */
.product-container img {
  transition: transform 0.3s ease;
}

/* Enlarge image on hover */
.product-container img:hover {
  transform: scale(1.1);
}

/* Style for product links */
```

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

/* Change link color on hover */
.product-container a:hover {
  color: #f19fcf;
}

/* CSS to control the modal display */
.modal {
  display: none; /* Initially hide the modal */
  position: fixed;
  z-index: 1;
  left: 0;
  top: 0;
  width: 100%;
  height: 100%;
  overflow: auto;
  background-color: rgba(0, 0, 0, 0.4); /* Black background with opacity */
}

#descriptionContainer {
  color: black;
}

.modal-content {
  background-color: #fefefe;
  position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  padding: 20px;
  border: 1px solid #888;
  width: 30%; /* Adjust width to your preference */
  color: grey; /* Change the text color to grey */
}
```

```
.close {
  color: #aaaaaa;
  float: right;
  font-size: 28px;
  font-weight: bold;
}

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

/* setting table bila dia nak display balik */
table {
  width: 37%;
  border-collapse: collapse;
  margin-bottom: 20px;
}

th,
td {
  border: 1px solid #ddd;
  padding: 8px;
  text-align: center;
}

th {
  background-color: #f2f2f2;
}

tr:nth-child(even) {
  background-color: #f9f9f9;
}

tr:hover {
  background-color: #f5f5f5;
}
```

```
/*container untuk kt crud.html*/
.containerCRUD {
  padding: 40px;
  border-radius: 50px;
  box-shadow: 0 0 20px rgba(0, 0, 0, 0.1);
  background-color: rgba(241, 159, 207, 0.5); /* Pink with 50% opacity */
  margin-top: 30px;
  margin-left: 575px;
  max-width: 600px;
  position: absolute;
}

/*setting untuk tulisan dalam container kt crud.html*/
.form-group {
  color: #b41d6e;
}

.desc {
  color: #b41d6e;
  font-size: 16px;
}
```

Index.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Makeup Box</title>
    <link rel="stylesheet" href="index.css" /> <!-- Link to the CSS file -->
    <script src="fetchapi.js"></script> <!-- Fetch API Script -->
    <link href="https://fonts.googleapis.com/css2?family=Brush+Script"
rel="stylesheet"> <!-- Font import -->
  </head>
  <body>
    <div class="navbar">
      <a href="index.html">Home</a> <!-- Navigation link to Home -->
      <a href="crud.html">Top 5</a> <!-- Navigation link to Top 5 -->
    </div>

    <center>
```

```
      <br><br><br>
      <h1>The Beauty Palette</h1> <!-- Main header -->
      <br>
      <div class="desc">Pick your favourite make-up now!</div> <!-- Description -->
      <br><br>
      <select id="brandDropdown" onchange="getProductsByBrand()">
        <option disabled selected>Brand</option> <!-- Default placeholder for brand
dropdown -->
      </select>
      <select id="productTypeDropdown" onchange="getProductsByType()">
        <option disabled selected>Product Type</option> <!-- Default placeholder for
product type dropdown -->
      </select>
    </center>

    <!-- Modal HTML structure -->
    <div id="makeupModal" class="modal">
      <div class="modal-content">
        <span class="close" onclick="closeModal()">&times;</span> <!-- Modal close
button -->
        <div id="descriptionContainer">
          <b>Product Type:</b> <span id="productTypeSpan"></span>
          <br><br>
          <b>Description:</b> <span id="descriptionSpan"></span>
          <br><br>
          <b>Product Link:</b> <span id="productLinkSpan"></span>
        </div>
      </div>
    </div>

    <br>
    <ul id="productList"></ul> <!-- Product list container -->
  </body>
</html>
```

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')) {
  // eslint-disable-line global-require
  app.quit();
}

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

  });

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

6. Submit files in GitHub.
   https://github.com/hadirahaffendy/makeup.git