ART COMM.

Report

By

Mister Jiaxian	**Chen**	**6388165**
Miss Thanaporn	**Artidtayamontol**	**6688005**
Miss Rawisara	**Chantravutikorn**	**6688015**
Miss Teerada	**Chatrattanawuth**	**6688016**
Miss Nichakorn	**Wisalkamol**	**6688146**

A Report Submitted in Partial Fulfillment of
the Requirements for

ITCS223 Introduction to Web Development

Faculty of Information and Communication Technology
Mahidol University

2024

Table of Contents

Project Overview

ART COMM. – Online Art Commission Platform

Overview:

ART COMM. is an online platform designed to connect independent artists with customers who are looking for commissioned artwork. Whether users are exploring art styles or seeking custom work, the system streamlines the process of discovery, communication, and commission placement.

The platform enables artists to showcase their works, define pricing, and describe their skills, while clients can search, filter, and browse available commissions. A dedicated admin panel allows site maintainers to manage artist data and product offerings.

Key Features:

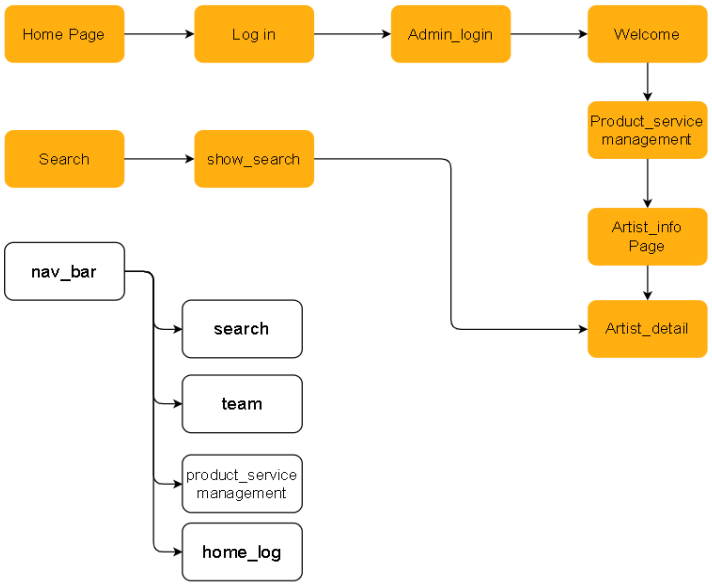
- Responsive homepage with admin-login access.
- Advanced search with category, artist name, status and base price filters.
- Artist detail page including artist's information, category, base price, and sample works.
- Admin management dashboard (CRUD) for adding/editing/deleting artists.
- Bilingual support (EN/TH) and a responsive sidebar navigation system.

System Architecture:

The web application uses a two-tier structure:

- Front-end: HTML, CSS, and JavaScript.
- Back-end: Node.js (web services), with database operations on MySQL.

RESTful API communication ensures dynamic data retrieval and update. All elements follow responsive web design principles to ensure cross-device compatibility.



The navigation diagram .

The navigation diagrams illustrate the structural flow of our web application for our administrators. Admin are required to **Log in** first in the **Admin_login page**. After successful authentication, they will access the **welcome** page, from which they can manage artists via the **Product_management** page and **Artist_detail Page**, which provides detailed management options. Admin can also access the **search page** from the nav bar. The search result will be displayed on the **search_artist page** after click search. The result of the search page is also connected with the **artist_detail page.**

The navigation bar are connected to **Home page**, **Product and Service page**, and **team page**

Database

ARTCOMM.

Table : Artist

```sql
CREATE TABLE IF NOT EXISTS Artist(
    ArtistID CHAR(13) NOT NULL,
    ArtistName VARCHAR(30) NOT NULL,
    ArtistLanguage VARCHAR(100) NOT NULL,
    ArtistCountry VARCHAR(100) NOT NULL,
    ACID CHAR(13) NOT NULL,
    BasePrice DECIMAL(10,2) ,
    Status VARCHAR(30) NOT NULL,
    AAdminID CHAR(13),
    ALogID CHAR(13) ,
    PRIMARY KEY(ArtistID),
    FOREIGN KEY(ACID) REFERENCES Category(CID),
    FOREIGN KEY(AAdminID) REFERENCES Admin(AdminID),
    FOREIGN KEY(ALogID) REFERENCES Search_log(LogID)
);
ALTER TABLE Artist
ADD COLUMN AboutMe TEXT DEFAULT NULL;
ALTER TABLE Artist ADD COLUMN PhotoPath VARCHAR(255);
```

```sql
INSERT INTO Artist(ArtistID, ArtistName, ArtistLanguage, ArtistCountry, ACID, BasePrice, Status, AAdminID, ALogID,AboutMe,PhotoPath) VALUES
('ART000000001', 'LunaDraws', 'English', 'USA', 'CAT000000001', 30.00, 'Available', 'ADM000000001', 'LOG000000001','I draw anime cute arts','https://i.pinimg.com/736x/
('ART000000002', 'PixelPete', 'Japanese', 'Japan', 'CAT000000004', 45.00, 'Unavailable', 'ADM000000004', 'LOG000000004','Food Pixel: Small object pixel !!! 💫💗','htt
('ART000000003', 'RealAmy', 'Spanish', 'Spain', 'CAT000000002', 70.00, 'Available', 'ADM000000002', 'LOG000000002','K-pop art','https://i.pinimg.com/736x/05/6b/d0/056b
('ART000000004', 'ChibiChan', 'Korean', 'South Korea', 'CAT000000005', 25.00, 'Available', 'ADM000000005', 'LOG000000005','Chibi headshot','https://i.pinimg.com/736x/6
('ART000000005', 'ToonMaster', 'English', 'UK', 'CAT000000003', 35.00, 'Available', 'ADM000000003', 'LOG000000003','Mario Peaches go brrrr','https://i.pinimg.com/736x/
('ART000000006', 'Ducartoon', 'English', 'Thailand', 'CAT000000001',99.00, 'Available', 'ADM000000001', 'LOG000000006','Spyxfamily Fanart artist','https://i.pinimg.com
('ART000000007', 'CyberInk', 'English', 'Germany', 'CAT000000002', 55.00, 'Unavailable', 'ADM000000002', 'LOG000000007','90s,80s semi-real art','https://i.pinimg.com/7
('ART000000008', 'FaceCrafter', 'French', 'France', 'CAT000000003', 40.00, 'Unavailable', 'ADM000000003', 'LOG000000008','Paper art','https://i.pinimg.com/736x/57/1a/0
('ART000000009', 'SceneQueen', 'English', 'USA', 'CAT000000004', 50.00, 'Available', 'ADM000000004', 'LOG000000009','Kurukuru','https://i.pinimg.com/736x/1d/95/1e/1d95
('ART000000010', 'Hirorosh', 'English', 'Australia', 'CAT000000005', 80.00, 'Available', 'ADM000000005', 'LOG000000010','Tsukasa For Life!!','https://i.pinimg.com/736x
```

Table : Admin

```sql
CREATE TABLE IF NOT EXISTS Admin(
    AdminID CHAR(13) NOT NULL,
    AdminGmail VARCHAR(100) NOT NULL,
    AMFirstname VARCHAR(30) NOT NULL,
    AMLastname VARCHAR(30) NOT NULL,
    PRIMARY KEY(AdminID)
);
```

```sql
INSERT INTO Admin(AdminID, AdminGmail, AMFirstname, AMLastname) VALUES
    ('ADM000000001', 'Francis.admin@gmail.com', 'Jiaxian', 'Chen'),
    ('ADM000000002', 'Thanaporn.admin@gmail.com', 'Thanaporn', 'Artidtayamontol'),
    ('ADM000000003', 'Rawisara.admin@gmail.com', 'Rawisara', 'Chantravutikorn'),
    ('ADM000000004', 'Teerada.admin@gmail.com', 'Teerada', 'Chatrattanawuth'),
    ('ADM000000005', 'Nichakorn.admin@gmail.com', 'Nichakorn', 'Wisalkamol');
```

Table : Adminlogin

```sql
CREATE TABLE IF NOT EXISTS Adminlogin(
    Username CHAR(50) NOT NULL,
    Password VARCHAR(50) NOT NULL,
    LAdminID CHAR(13),
    PRIMARY KEY (Username),
    FOREIGN KEY(LAdminID) REFERENCES Admin(AdminID)
);

INSERT INTO Adminlogin(Username, Password, LAdminID) VALUES
    ('Francis_admin', 'passAlice123', 'ADM000000001'),
    ('Thanaporn_admin', 'secureBob456', 'ADM000000002'),
    ('Rawisara_admin', 'carla789!', 'ADM000000003'),
    ('Teerada_admin', 'davidKey321', 'ADM000000004'),
    ('Nichakorn_admin', 'emmapwd456', 'ADM000000005');
```

Table : Category

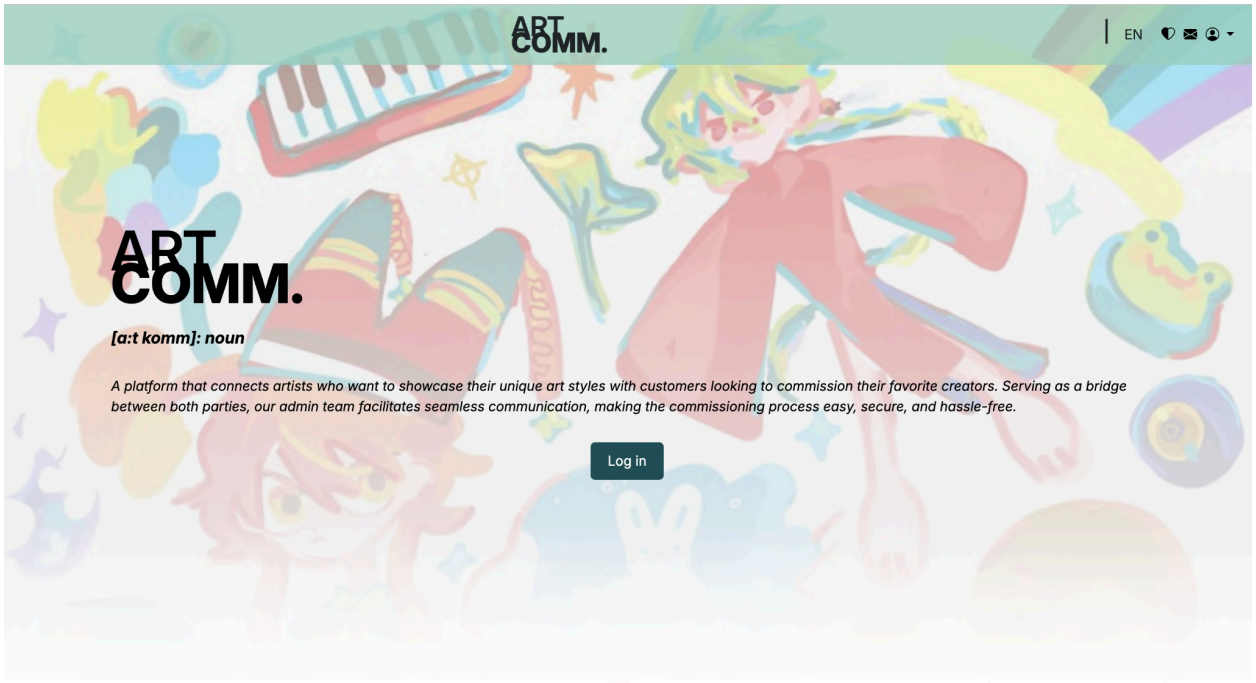
```sql
CREATE TABLE IF NOT EXISTS Category(
    CID CHAR(13) NOT NULL,
    C_Name VARCHAR(30) NOT NULL,
    PRIMARY KEY(CID)
);

INSERT INTO Category(CID, C_Name) VALUES
    ('CAT000000001', 'Anime Style'),
    ('CAT000000002', 'Realism'),
    ('CAT000000003', 'Cartoon'),
    ('CAT000000004', 'Pixel Art'),
    ('CAT000000005', 'Chibi');
```

Table : Search_log

```sql
CREATE TABLE IF NOT EXISTS Search_log(
    LogID CHAR(13) NOT NULL,
    SCID CHAR(13) NOT NULL,
    Title VARCHAR(100) NOT NULL,
    PRIMARY KEY(LogID),
    FOREIGN KEY(SCID) REFERENCES Category(CID)
    );


INSERT INTO Search_log(LogID, SCID, Title) VALUES
    ('LOG000000001', 'CAT000000001', 'Anime Art'),
    ('LOG000000002', 'CAT000000002', 'Realistic Cartoon'),
    ('LOG000000003', 'CAT000000003', 'Cartoon-style icon'),
    ('LOG000000004', 'CAT000000004', 'Pixel Game Asset'),
    ('LOG000000005', 'CAT000000005', 'Cute Chibi'),
    ('LOG000000006', 'CAT000000001', 'Anime Style Character Design'),
    ('LOG000000007', 'CAT000000002', 'Realism'),
    ('LOG000000008', 'CAT000000003', 'Cartoon'),
    ('LOG000000009', 'CAT000000004', 'Background'),
    ('LOG000000010', 'CAT000000005', 'Chibi');
```

Detail of web application and code

Home page : http://localhost:3030/

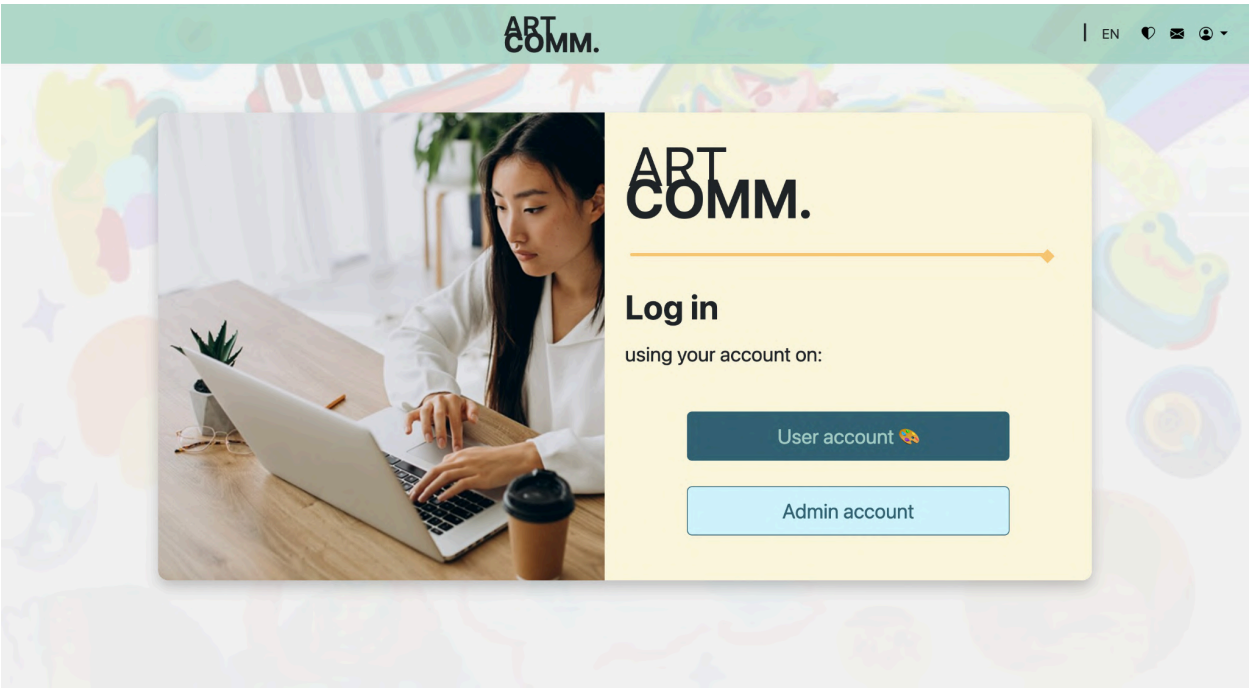


Code:

```
router.get("/", (req, res) => {
    res.sendFile(path.join(`${__dirname}/home2.html`))
    res.status(200)
    console.log(`Request at ${req.url}`);
})
```

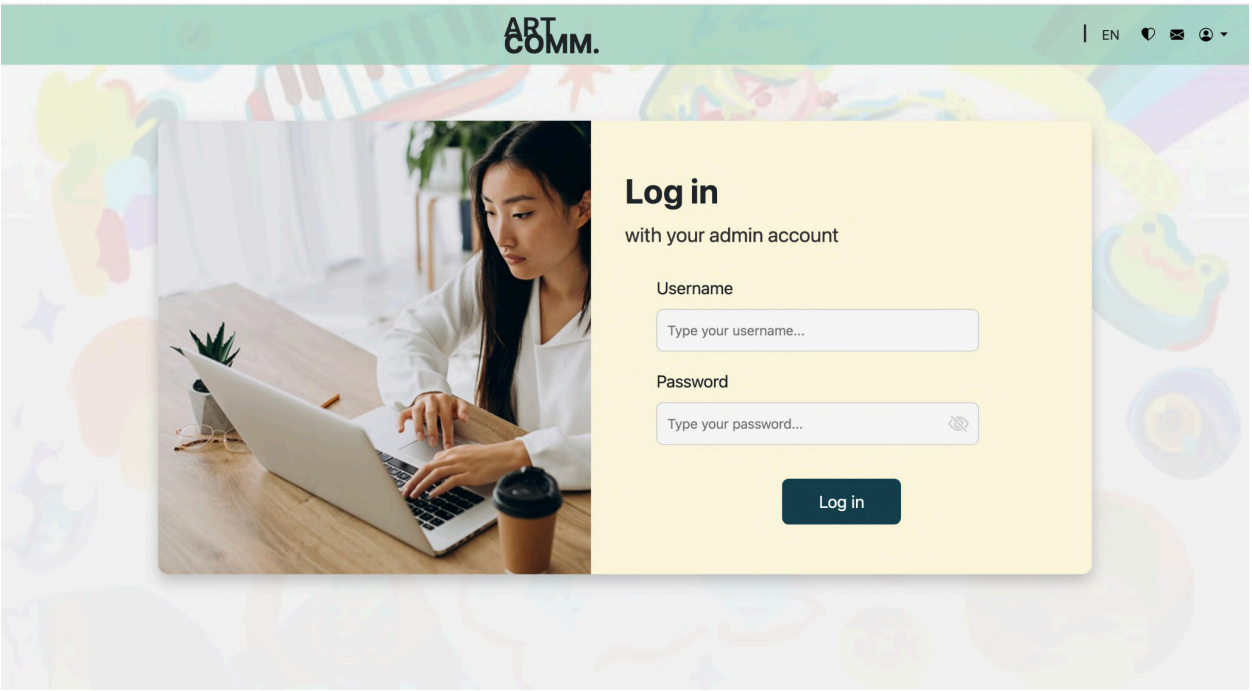
The webpage is running on port 3030. This page will show the inspiration of our website. It including login button that will led to /login page

Login page: http://localhost:3030/login



After we click the Admin account button, it will lead to /admin_login page. This page will let the admin fill the username and password.

Admin_Login page: http://localhost:3030/admin_login



Code:

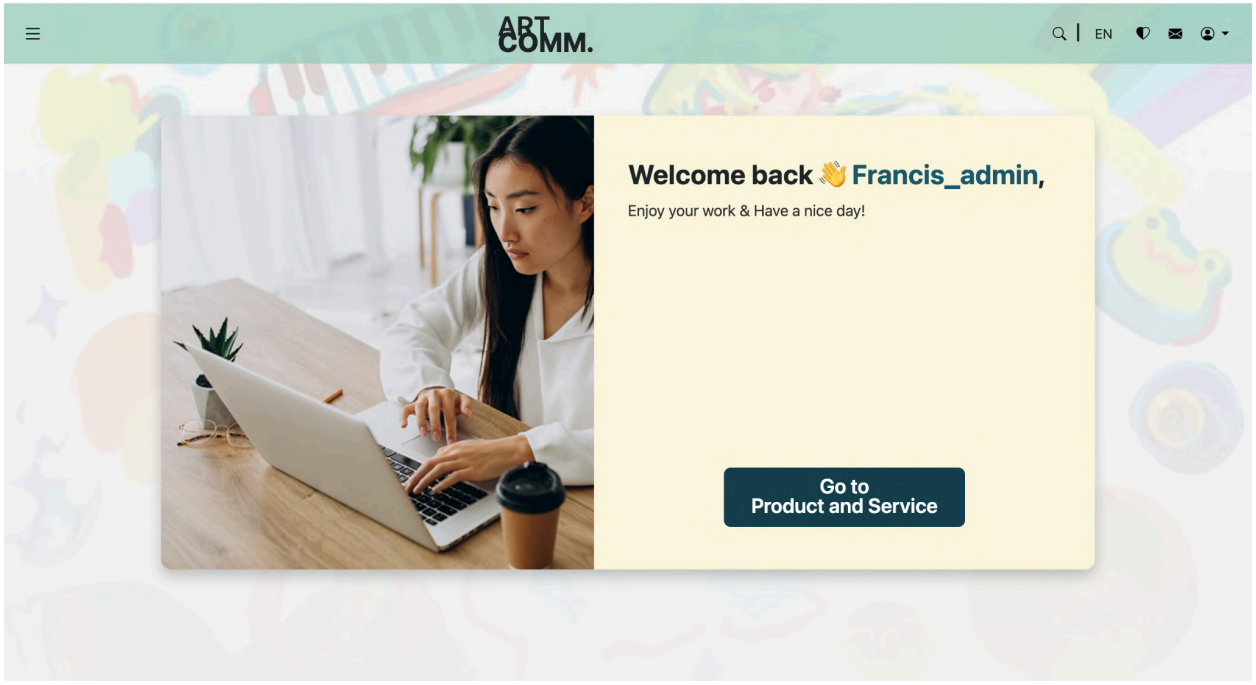
```
router.get("/admin_login", (req, res) => {
    res.status(200).sendFile(path.join(`${__dirname}/admin_login.html`));
    console.log(`Request at ${req.url}`);
})
router.post('/form-submit',function (req,res){
    console.log(req.method);
    console.log(req.body.Username)
    res.status(200);
});
```

We created a form to get the Username and Password and will be using it in the getlogin() If it is correct, it will show /welcome page.

Code Getlogin():

```
async function Getlogin() {
  const AdminUser = document.querySelector("#username").value;
  const AdminPassword = document.querySelector("#password").value;
  const data = {
    Username: AdminUser,
    Password: AdminPassword
  };
  const response = await callArtistWS(rooturl + "login", "insert", data)
  if (response.status === 200) {
    localStorage.setItem("Username", response.Username);
    console.log("Successfully Login");
    alert("Successfully Login");
    window.location.href = "/welcome";
  }
  else if (response.status === 401) {
    console.log(response.message);
    alert("Error occurs T T")
  }
  else {
    alert('Please enter Username and Password');
  }
}
```

Welcome page:

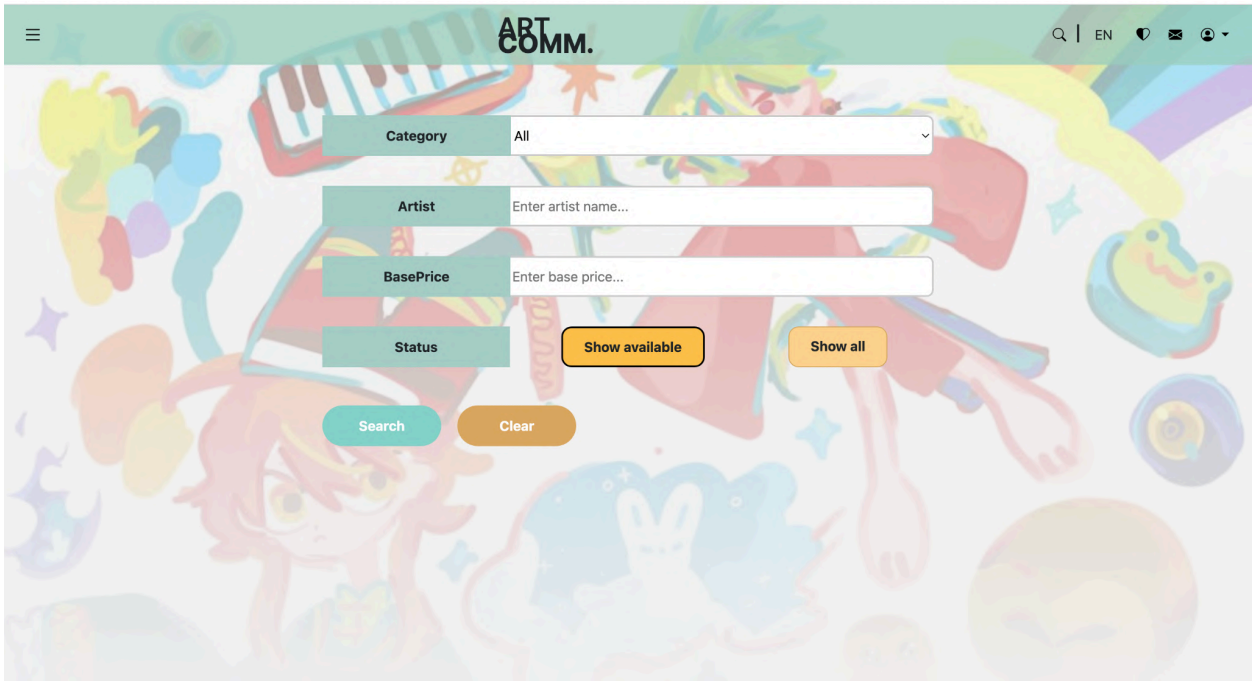


The welcome page will show the button to /product_manage page for the admin to add/edit/delete new products(artist).

The name of the admin who logged in will be displayed using the DisplayUser() function.
Code DisplayUser():

```
let DisplayUser = () => {
  const Usernamep = document.getElementById("username-placeholder");
  const username = localStorage.getItem("Username");

  if (!username) {
    console.warn("⚠️ No username in localStorage.");
    Usernamep.innerHTML = "👋 User";
    return;
  }

  fetch(`${rooturl}admin?username=${encodeURIComponent(username)}`)
    .then((res) => res.json())
    .then((data) => {
      console.log("Fetched admin data:", data); // optional debug
      if (data.data && data.data.Username) {
        Usernamep.innerHTML = `👋 ${data.data.Username}`;
      } else {
        Usernamep.innerHTML = `👋 User`;
      }
    })
    .catch((err) => {
      console.error("Error fetching user data:", err);
      Usernamep.innerHTML = "👋 User";
    });
};
if (window.location.pathname.includes("welcome")) {
  document.addEventListener('DOMContentLoaded', function () {
    DisplayUser();
  });
}
```

Search page: http://localhost:3030/search



The search page will show the criteria to search which are category, artist name, baseprice and status. When you click search it will lead to /search_artist page that will show all the artists that match the criteria we selected.The clear button can be used to clear the input that type on the search. The getArtists() function will receive the 4 criteria to fetch with all the GET methods in the back-end.

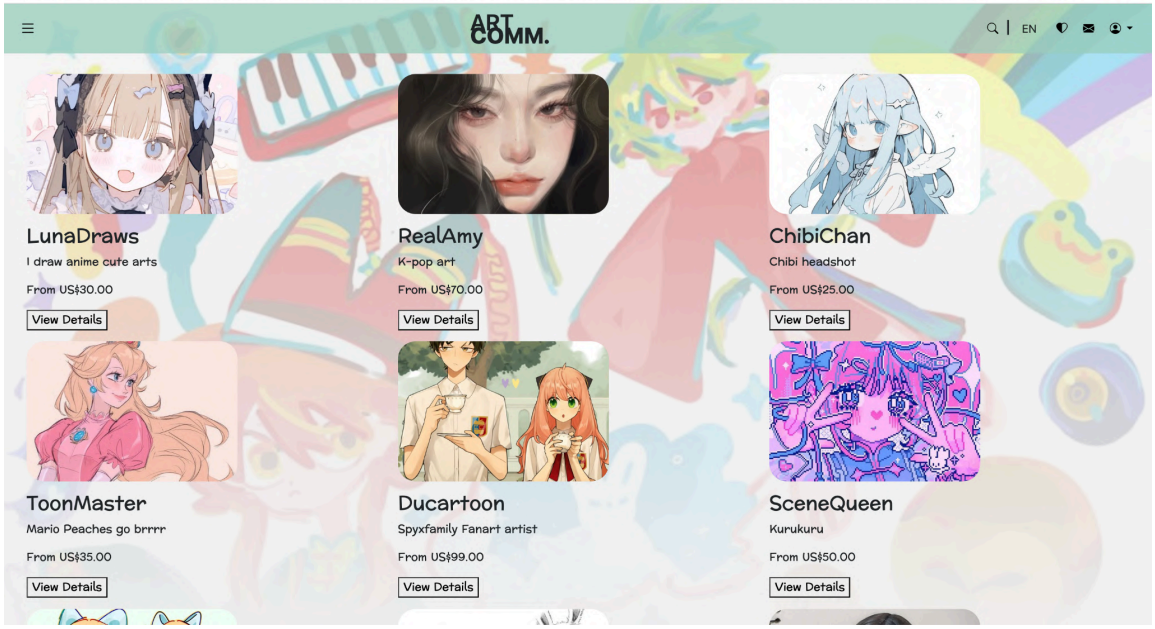
Code:

```
router.get("/search", (req, res) => {
    res.sendFile(path.join(`${__dirname}/search.html`))
    res.status(200)
    console.log(`Request at ${req.url}`);
})
```

Code getArtists():

```
let getArtists = () => {
  const category = document.getElementById("category").value;
  const name = document.getElementById("name").value.trim();
  const availableRadio = document.getElementById("available");
  const allRadio = document.getElementById("all");
  const basePriceInput = document.getElementById("baseprice");
  const basePrice = basePriceInput ? basePriceInput.value.trim() : "";
  let status = "";

  if (availableRadio && availableRadio.checked) {
    status = "Available";
  } else if (allRadio && allRadio.checked) {
    status = "all"; // "Show all" means no status filter
  }
  const rooturl = "http://localhost:3031/";
  let apiUrl = rooturl + "artist/artists"; // Default URL to get all artists

  const categoryMapReverse = {
    "Anime Style": "CAT000000001", "Realism": "CAT000000002","Cartoon": "CAT000000003","Pixel Art": "CAT000000004", "Chibi": "CAT000000005"
  };

  const categoryId = categoryMapReverse[category];
  if (status && basePrice && categoryId && name) { // Status + BasePrice + Category + Name
    apiUrl = `${rooturl}artist/statusbasecategoryname/${encodeURIComponent(status)}/${encodeURIComponent(basePrice)}/${encodeURIComponent(categor
  }
  else if (status !== "" && !categoryId && !basePrice && !name) { // Status
    apiUrl = `${rooturl}artist/status/${encodeURIComponent(status)}`;
  }
  else if (basePrice && categoryId && status !== "") { // Status + BasePrice + Category
    apiUrl = `${rooturl}artist/statusbasecategory/${encodeURIComponent(status)}/${encodeURIComponent(basePrice)}/${encodeURIComponent(categoryId)
  }
  else if (name && basePrice && status !== "") { // Status + Name + BasePrice
    apiUrl = `${rooturl}artist/statusnamebaseprice/${encodeURIComponent(status)}/${encodeURIComponent(name)}/${encodeURIComponent(basePrice)}`;
  }
  else if (name && categoryId && status !== "") { // Status + Category + Name
    apiUrl = `${rooturl}artist/statuscategoryname/${encodeURIComponent(status)}/${encodeURIComponent(categoryId)}/${encodeURIComponent(name)}`;
  }
  else if (basePrice && status) { // Status + BasePrice
    apiUrl = `${rooturl}artist/statusbaseprice/${encodeURIComponent(status)}/${encodeURIComponent(basePrice)}`;
```

The /search_artist page will use displaySearchResults() to display all the artists that match the criteria from search.

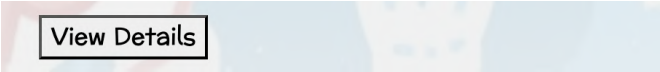


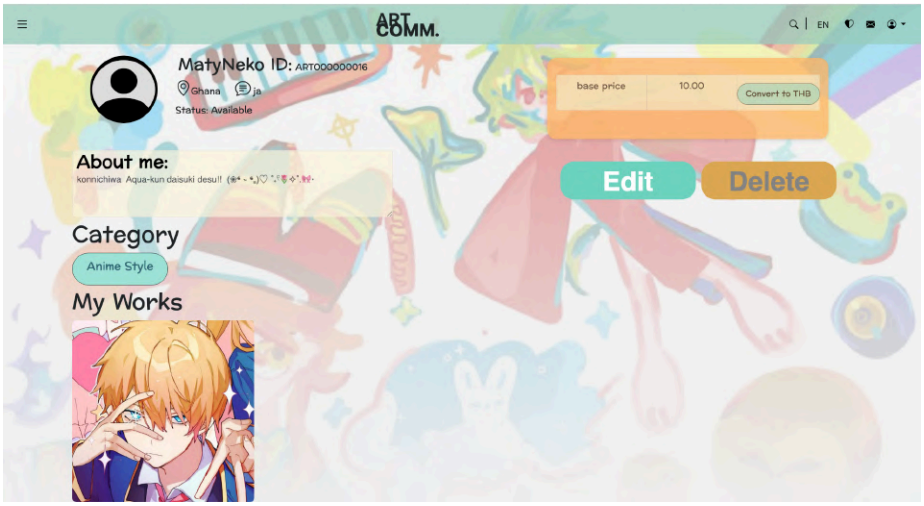
Code displaySearchResults():

```
function displaySearchResults() {
  if (!window.location.pathname.includes("search_artist")) {
    return;
  }
  const resultsContainer = document.getElementById('container');
  if (!resultsContainer) {
    console.error("Results container not found on page");
    return;
  }
  const searchResultsJSON = localStorage.getItem('searchResults');
  const searchResultsError = localStorage.getItem('searchResultsError');


  // Clear storage after reading
  localStorage.removeItem('searchResultsError');

  if (searchResultsError) {
    resultsContainer.innerHTML = `<p class="error-message">${searchResultsError}</p>`;
    return;
  }
  if (searchResultsJSON) {
    const artists = JSON.parse(searchResultsJSON);
    // Check if any artists show
    if (artists && artists.length > 0) {
      let output = '';
      // Loop through each artist
      artists.forEach(artist => {
        // Create artist card HTML
        output += `
        <div class="artist-card">
          <div class="artist-image">
            <img src="${artist.PhotoPath || '/image/Product_img/User.png'}"
                 alt="${artist.ArtistName}"
                 style="width: 300px; height: 200px; object-fit: cover; border-radius: 25px"


                 onerror="this.src='/image/Product_img/User.png'">
          </div>
          <div class="artist-info">
            <h3>${artist.ArtistName}</h3>
            <p class="artist-description">${artist.AboutMe || 'No description available.'}</p>
            <p class="artist-price">From US$${artist.BasePrice}</p>
            <button><a href="/artist_details?id=${artist.ArtistID}" class="view-button" data-en="View Details" data-th="ดูรายละเอียด">View Details</a></button>
          </div>
```

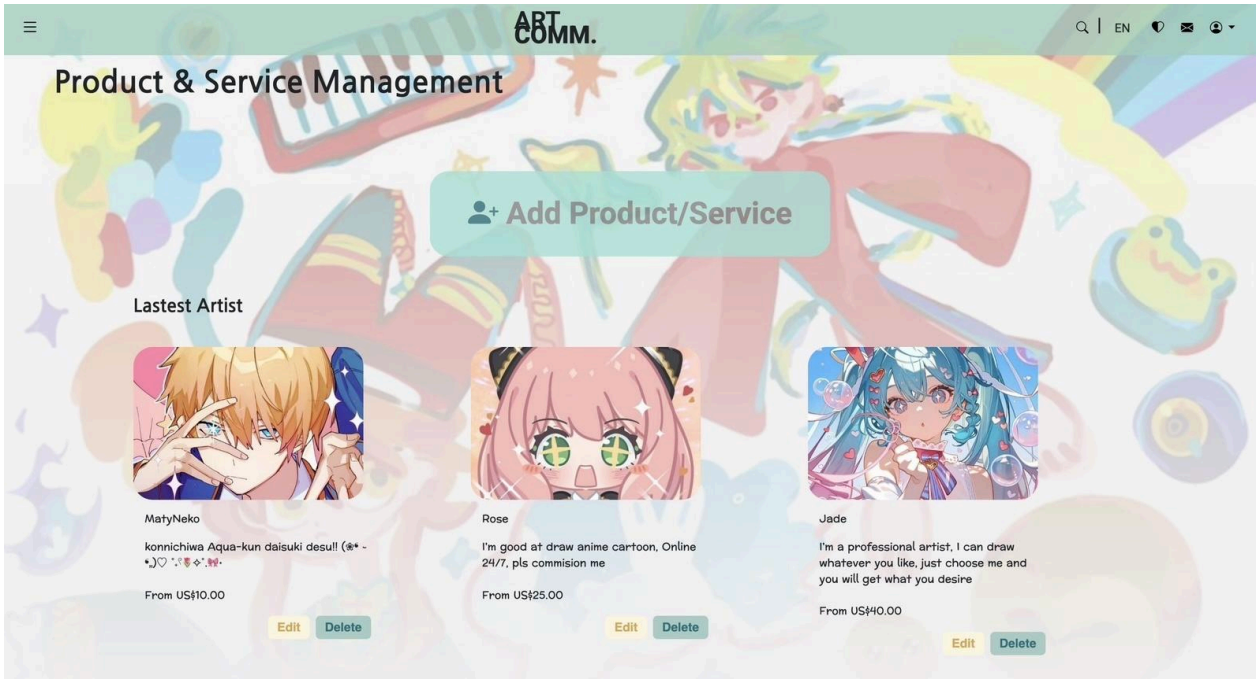
```
      </div>`;
    });
    // display outpit
    resultsContainer.innerHTML = output;
  } else {
    resultsContainer.innerHTML = '<p>No artists found matching your search criteria.</p>';
  }
 } else {
  resultsContainer.innerHTML = '<p>No search results available.</p>';
 }
}
```

Click “view details” and it will lead to /artist_details?id=… page of the artist we choose.





Product and Service page: http://localhost:3030/product



Code:

```
router.get("/product", (req, res) => {
    res.sendFile(path.join(`${__dirname}/Product.html`))
    res.status(200)
    console.log(`Request at ${req.url}`);
})
```

This page will show the 3 latest artists that have been added or edited by using the function LoadLatestArtists(). And it also has the “Add product/service” button that will lead to the page for adding artists.

Code LoadLatestArtists():

```
function loadLatestArtists() {
  if (window.location.pathname.includes("product")) {
    const artistsContainer = document.getElementById("latest-artists-container");
    if (!artistsContainer) return;

    fetch(`${rooturl}artist/latest/3`)
      .then(response => response.json())
      .then(data => {
        if (!data.error && data.data && data.data.length > 0) {
          const artists = data.data;
          let output = '';

          artists.forEach(artist => {
            output += `
            <div class="col-md-4">
              <div class="card-h-100">
                <img src="${artist.PhotoPath || '/image/Product_img/User.png'}" class="card-image-fixed"
                     onerror="this.src='/image/Product_img/User.png'">
                <div class="card-text-container">
                  <p class="preahvihear" style="margin-top: 15px;">${artist.ArtistName}</p>
                  <p class="preahvihear" style="margin-top: 15px;">${artist.AboutMe || 'No description available.'}<br><br>
                  From US$${artist.BasePrice}</p>
                  <div class="mt-auto d-flex justify-content-end gap-2">
                    <a href="/artist_details?id=${artist.ArtistID}" class="btn edit-btn fw-bold" data-en="Edit" data-th="แก้ไข">Edit</a>
                    <button class="btn delete-btn fw-bold" data-artist-id="${artist.ArtistID}" data-en="Delete" data-th="ลบ">Delete</button>
                  </div>
                </div>
              </div>
            </div>`;
          });

          artistsContainer.innerHTML = output;

          // Add event listeners to the newly created delete buttons
          document.querySelectorAll('.delete-btn').forEach(btn => {
            btn.addEventListener('click', function () {
              const artistId = this.getAttribute('data-artist-id');
              if (confirm("Are you sure you want to delete this artist?")) {
                deleteArtistById(artistId);
```

```
                }
              });
            });
        } else {
          artistsContainer.innerHTML = '<div class="col-12"><p class="text-center">No artists found.</p></div>';
        }
      })
      .catch(error => {
        console.error("Error loading latest artists:", error);
        artistsContainer.innerHTML = '<div class="col-12"><p class="text-center">Failed to load artists.</p></div>';
      });
  }
}
```

Add Product(artist) page: http://localhost:3030/product_manage

Code:

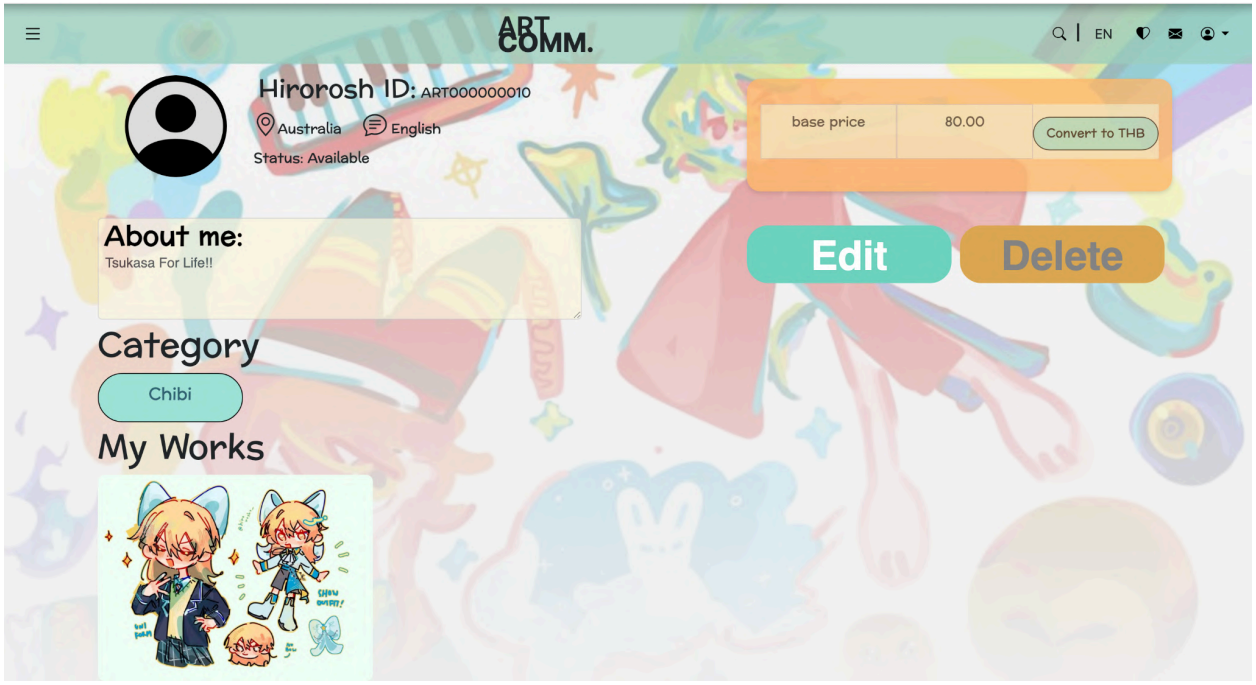
```
router.get("/product_manage", (req, res) => {
    res.sendFile(path.join(`${__dirname}/Product2.html`))
    res.status(200)
    console.log(`Request at ${req.url}`);
})
```

This page allows admin to fill the artist's details which including

- Artist's name
- Artist's language
- Artist's country
- Artist's status (Available now or Unavailable)
- About Me (Artist's bio)
- URL of Artist's work
- Artist's base price for one art

After we add all of this information and click save. The web will automatically lead to /artist_detail/id of that artist and use DisplayArtist() to show the results .

Artist's details page: http://localhost:3030/artist_details?id=ART00000…

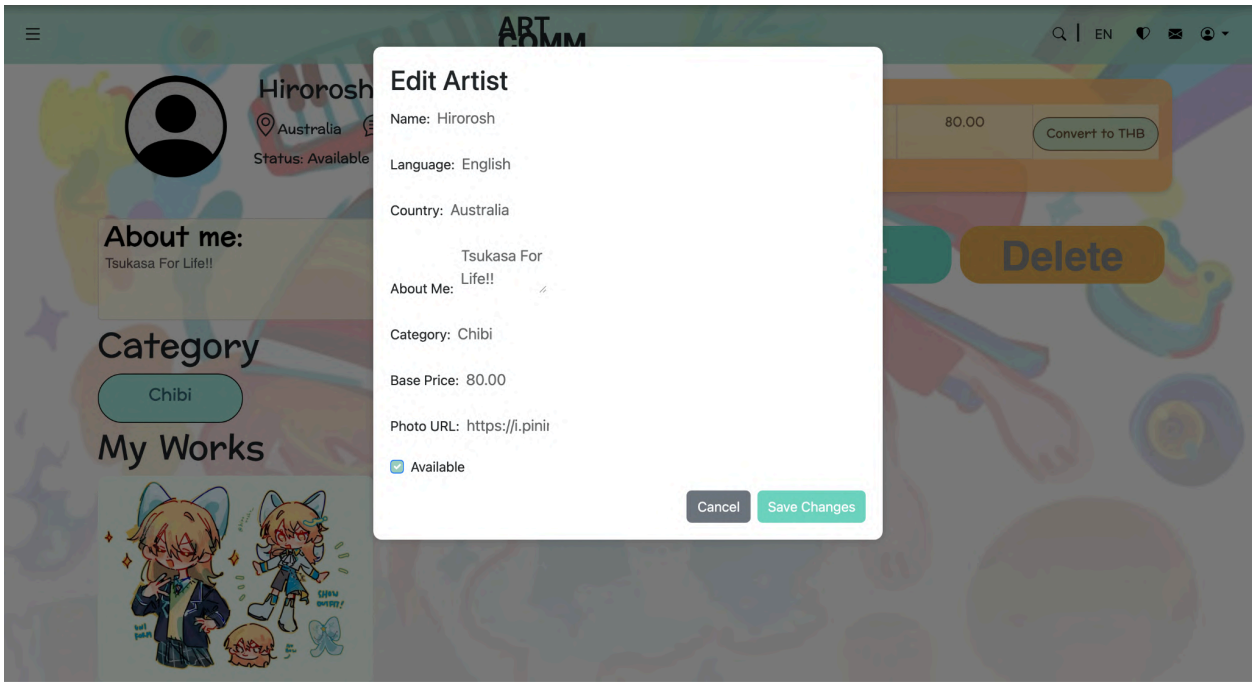


Code DisplayArtist():

```
let DisplayArtist = () => {
  const artistNameElement = document.getElementById("artistName");
  const artistAboutMeElement = document.getElementById("artistAboutMe");
  const artistLanguageElement = document.getElementById("artistLanguage");
  const artistCountryElement = document.getElementById("artistCountry");
  const artistCategoryElement = document.getElementById("artistCategory");
  const artiststatusElement = document.getElementById("artistStatus");
  const artistIDElement = document.getElementById("artistID");
  const artistBasepriceElement = document.getElementById("artistBaseprice")
  const artistPhotoElement = document.getElementById("artist-photo");
  const urlParams = new URLSearchParams(window.location.search);
  const artistId = urlParams.get('id');

  if (artistId) {
    fetch(`${rooturl}artist/artists/${artistId}`)
      .then(response => response.json())
      .then(data => {
        if (!data.error && data.data) {
          const artist = data.data;
          if (artistPhotoElement) {
            artistPhotoElement.src = artist.PhotoPath;
          }
          if (artistNameElement) {
            artistNameElement.textContent = artist.ArtistName;
          }
          if (artistAboutMeElement) {
            artistAboutMeElement.textContent = artist.AboutMe;
          }
          if (artistLanguageElement) {
            artistLanguageElement.textContent = artist.ArtistLanguage;
          }
          if (artistCountryElement) {
            artistCountryElement.textContent = artist.ArtistCountry;
          }
          if (artistCategoryElement) {
            artistCategoryElement.textContent = artist.ArtistCategoryName;
          }
          if (artiststatusElement) {
            artiststatusElement.textContent = artist.Status;
          }
```

In this page it will show all the artist's information that we filed in the previous page. And along with the new ArtistID that we generated for each artist. The page also includes the edit and delete button.

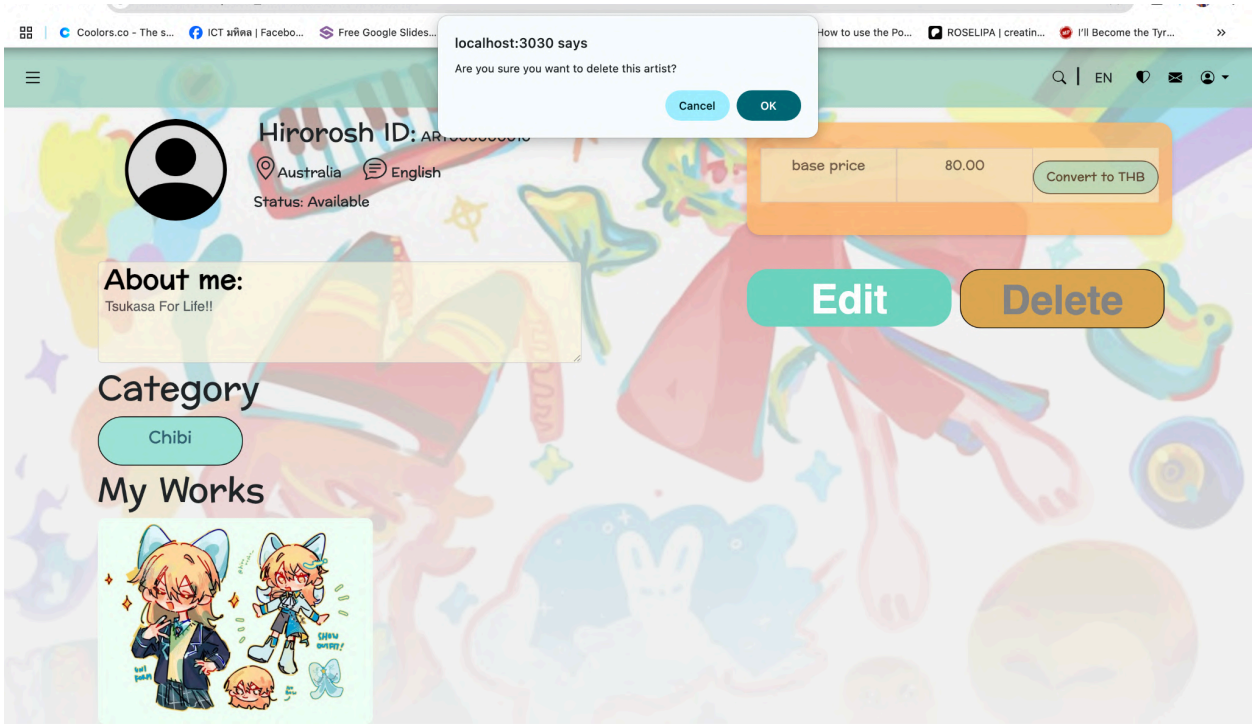
Edit button: allow admin to edit all the artist's information



Code : ShowEditForm() and updateArtists()

```
//click edit to show edit form
function showEditForm() {
  const urlParams = new URLSearchParams(window.location.search);
  const artistId = urlParams.get('id');

  if (!artistId) {
    alert("Error: No artist ID found");
    return;
  }
  // Fetch the current artist data to populate the form
  fetch(`${rooturl}artist/artists/${artistId}`)
    .then(response => response.json())
    .then(data => {
      if (!data.error && data.data) {
        const artist = data.data;
        populateEditForm(artist);
        document.getElementById('edit-form-container').style.display = 'block';
      } else {
        alert("Error: Could not retrieve artist details for editing.");
      }
    })
    .catch(error => {
      console.error("Error fetching artist details:", error);
      alert("Failed to fetch artist details for editing.");
    });
}
// after change the info click save to update information
async function updateArtist(e) {
  e.preventDefault();

  const urlParams = new URLSearchParams(window.location.search);
  const artistId = urlParams.get('id');

  if (!artistId) {
    alert("Error: No artist ID found");
    return;
  }
```

Delete button: ask admin if they really want to delete this artist before delete form database



Code: deleteArtist()

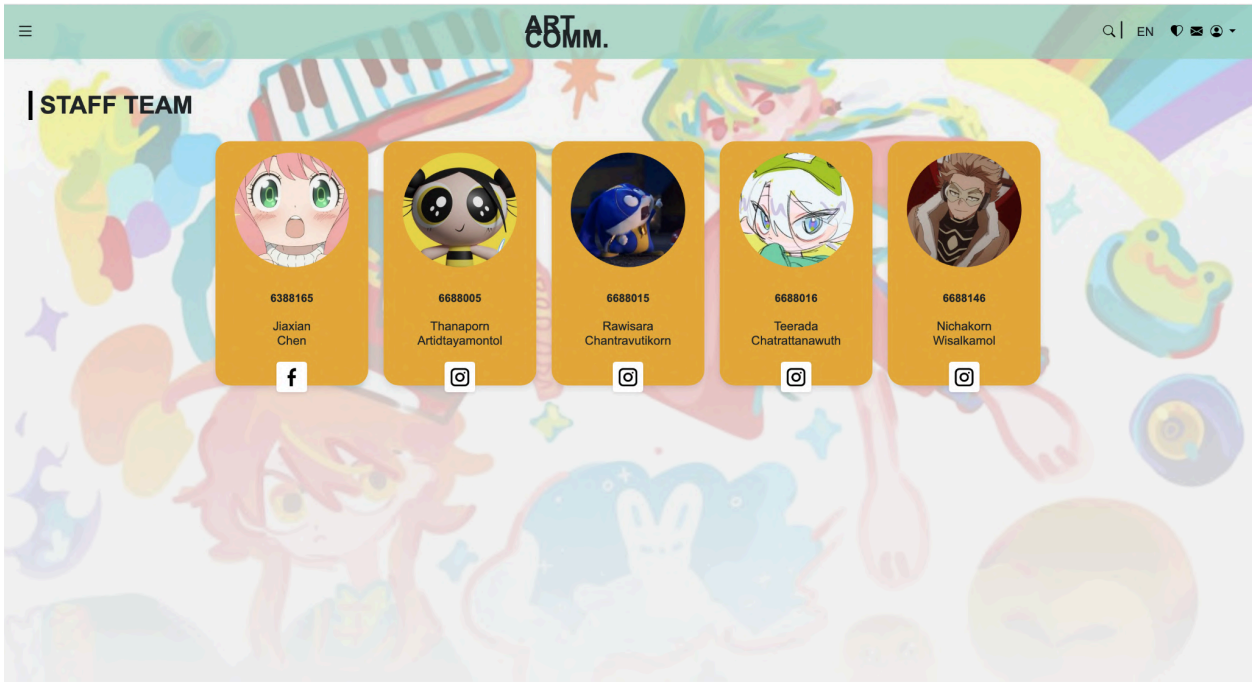
```
// Function to delete artist
async function deleteArtist() {
  const urlParams = new URLSearchParams(window.location.search);
  const artistId = urlParams.get('id');

  if (!artistId) {
    alert("Error: No artist ID found");
    return;
  }

  if (confirm("Are you sure you want to delete this artist?")) {
    try {
      const response = await fetch(`${rooturl}artist/delete`, {
        method: 'DELETE',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ ArtistID: artistId })
      });

      const result = await response.json();
      if (!result.error) {
        alert("Artist deleted successfully!");
        // Redirect to the product page
        window.location.href = "/product";
      } else {
        alert("Error: " + result.message);
      }
    } catch (error) {
      console.error("Delete request failed:", error);
      alert("Failed to delete artist. Please try again.");
    }
  }
}
```

In the Artist_details page, the page also has a “convert to THB” button that links to the public API name “https://freecurrencyapi.com”.
The API will convert the base price to THB (the limit is 5000 times)

Code convertToTHB():

```
async function convertToTHB() {
  const priceElement = document.getElementById("artistBaseprice");
  const thbPriceElement = document.getElementById("thb-price");

  if (!priceElement || !priceElement.textContent) {
    alert("Price not available");
    return;
  }
  // Get the base price
  const basePrice = parseFloat(priceElement.textContent.replace(/[^\d.]/g, ''));
  if (isNaN(basePrice)) {
    alert("Invalid price format");
    return;
  }
  // Show loading state
  thbPriceElement.textContent = "Converting...";
  try {
    const apiKey = "fca_live_cFTahnh8IMlUAfTUAEDqqk2JdkayhxWSy1FcPPN8";
    const response = await fetch(`https://api.freecurrencyapi.com/v1/latest?apikey=fca_live_cFTahnh8IMlUAfTUAEDqqk2JdkayhxWSy1FcPPN8&currencie

    if (!response.ok) {
      throw new Error(`HTTP error! status: ${response.status}`);
    }
    const data = await response.json();
    if (data && data.data && data.data.THB) {
      const exchangeRate = data.data.THB;
      const thbPrice = basePrice * exchangeRate;
      thbPriceElement.textContent = `฿${thbPrice.toFixed(2)} THB`;
    } else {
      throw new Error("Could not get THB exchange rate");
    }
  } catch (error) {
    console.error("Currency conversion failed:", error);
    thbPriceElement.textContent = "Conversion failed";
  }
}
```

Team page: http://localhost:3030/team

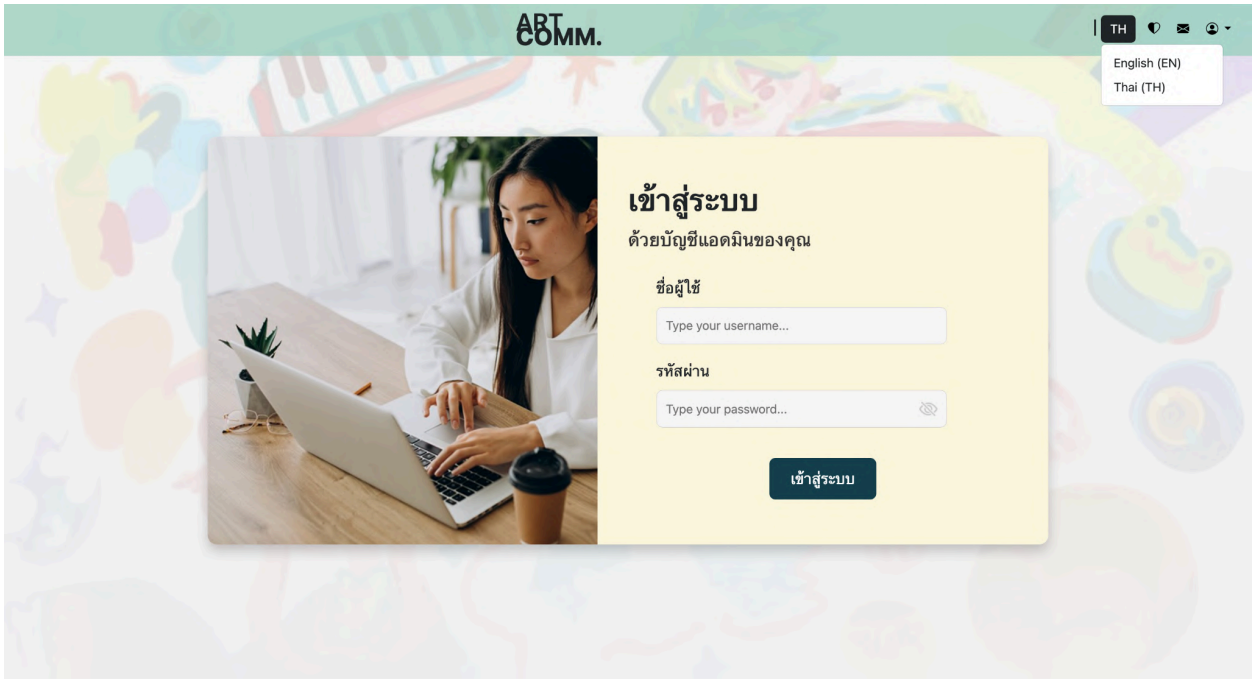


Code:

```
router.get("/team", (req, res) => {
    res.sendFile(path.join(`${__dirname}/team.html`))
    res.status(200)
    console.log(`Request at ${req.url}`);
})
```

This page will provide the admin's information and the link to their social media.

Our website can also change to Thai language by selecting [TH] at the navigation bar



Code:

```
document.querySelectorAll('.lang-option').forEach(option => {
  option.addEventListener('click', function (e) {
    e.preventDefault();
    const selectedLang = this.getAttribute('data-lang');
    const langBtn = document.getElementById('langDropdown');
    langBtn.textContent = selectedLang.toUpperCase();

    document.querySelectorAll('[data-en]').forEach(el => {
      const en = el.getAttribute('data-en');
      const th = el.getAttribute('data-th');
      el.textContent = (selectedLang === 'en') ? en : th;
    });
    // Handle elements with line1/line2 attributes
    document.querySelectorAll(`[data-${selectedLang}-line1]`).forEach(el => {
      const line1 = el.getAttribute(`data-${selectedLang}-line1`);
      const line2 = el.getAttribute(`data-${selectedLang}-line2`);
      el.innerHTML = line1 + '<br>' + line2;
    });
  });
});
```

Details of web services and code

The backend server include app.js and artist.js (module import from app.js) which locate in port 3031

- **Login authentication** - For the Login page we use POST for the HTTP method. Require admin's Username and Password and check if it matches with the data(password) in the database.

```
async function Getlogin() {
  const AdminUser = document.querySelector("#username").value;
  const AdminPassword = document.querySelector("#password").value;
  const data = {
    Username: AdminUser,
    Password: AdminPassword
  };
  const response = await callArtistWS(rooturl + "login", "insert", data)
  if (response.status === 200) {
    localStorage.setItem("Username", response.Username);
    console.log("Successfully Login");
    alert("Successfully Login");
    window.location.href = "/welcome";
  }
  else if (response.status === 401) {
    console.log(response.message);
    alert("Error occurs T T")
  }
  else {
    alert('Please enter Username and Password');
  }
}
```

- If the login is successful, it will show the pop up alert message "Successfully Login", and send to welcome page

```
router.get('/admin', (req, res) => {
    const username = req.query.username;
    connection.query("SELECT * FROM Adminlogin WHERE Username = ?", [username], (error, results) => {
        if (error) throw error;
        if (results.length > 0) {
            res.send({ error: false, data: results[0], message: 'Admin username' });
        } else {
            res.send({ error: false, data: {}, message: 'No admin found' });
        }
    });
});
router.post('/login', (req, res) => {
    let admin_username = req.body.Username;
    let admin_password = req.body.Password;
    console.log('admin', admin_username, 'log in', `at ${Date()}`);
    if (!admin_username || !admin_password) {
        return res.status(400).send({
            error: true,
            message: 'Please enter username and password'
        });
    }
    connection.query("SELECT * FROM Adminlogin WHERE Username = ? AND Password = ? ",
        [admin_username, admin_password],
        function (error, results) {
        if (error) throw error;
        if (results.length == 0) {
            return res.status(401).send({
                error: true,
                message: 'Invalid Login',
                status: 401
            });
        }
        else {
            return res.status(200).send({
                error: false,
                data: results[0],
                Username: results[0].Username,
                message: 'Successfully log in',
                status: 200
            });
```

- **Add artist -** POST Method

```
// Add artist
router.post("/add", function (req, res) {
  let Artist = req.body;
  if (!Artist) {
    return res.status(400).send({ error: true, message: 'Please provide Artist Information' });
  }
  //  provided URL
  if (Artist.photoURL) {
    Artist.PhotoPath = Artist.photoURL;
    delete Artist.photoURL;
  }
  connection.query("SELECT ArtistID FROM Artist ORDER BY ArtistID DESC LIMIT 1", function (err, result) {
    if (err) throw err;

    let nextNumericPart = 1;
    if (result.length > 0) {
      // Extract the numeric part
      const currentId = result[0].ArtistID;
      const numericPart = parseInt(currentId.replace('ART', ''));
      nextNumericPart = numericPart + 1;
    }

    // Format artist id with zeros to be ART000000000...
    const paddedNumber = String(nextNumericPart).padStart(9, '0');
    Artist.ArtistID = 'ART' + paddedNumber;

    connection.query("INSERT INTO Artist SET ?", [Artist], function (error, results) {
      if (error) throw error;
      const newArtistId = Artist.ArtistID;
      res.send({ error: false, data: { ArtistID: newArtistId }, message: 'New Artist has been created!' });
    });
  });
});
```

- The method shows the process of adding new artists from the /product_manage by fetch with the POST /add. The method creates a new ArtistID by counting the last digit of the latest ArtistID and plus by 1. Create a new artistID in the form of ART000000000XX and then add the new ID and information including Photo URL to the database.

- **Update artist -** PUT Method

```
// Update artist
router.put("/update", function (req, res) {
  let ArtistID = req.body.ArtistID;
  let Artist = req.body;
  if (!Artist || !ArtistID) {
    return res.status(400).send({ error: true, message: 'Please provide Artist Information' });
  }
  connection.query("UPDATE Artist SET ? WHERE ArtistID = ?", [Artist, ArtistID], function (error, results) {
    if (error) throw error;
    res.send({ error: false, data: results.affectedRows, message: 'Artist has been updated!' });
  });
});
```

- The update(put) method will be used with the edit button in /artist_details page to allow admin to edit artist's information by receiving artistID (It will be automatically retrieved from the database by the function, so the admin does not have to enter the ID each time.)

- **Delete artist -** DELETE Method

```
// Delete artist
router.delete("/delete", function (req, res) {
  let ArtistID = req.body.ArtistID;
  if (!ArtistID) {
    return res.status(400).send({ error: true, message: 'Please provide Artist ID' });
  }
  connection.query("DELETE FROM Artist WHERE ArtistID = ?", [ArtistID], function (error, results) {
    if (error) throw error;
    res.send({ error: false, data: results.affectedRows, message: 'Artist has been deleted!' });
  });
});
```

  - The delete method will delete all the artist information from the database and send back the alert that the Artist has been deleted!

- **Search artist -** The search bar will have 4 criteria, but the "status" will be a checkbox where either "Show Available" or "Show All" must always be selected.

METHOD: GET

search by Status

```
// Search by status
router.get('/status/:status', (req, res) => {
  const artist_status = req.params.status;

  if (!artist_status) {
    return res.status(400).send({ error: true, message: 'Please provide status' });
  }

  let query = `
  SELECT
    a.*,
    c.C_Name AS ArtistCategory
  FROM
    Artist a
  JOIN
    Category c ON a.ACID = c.CID
`;

  let params = [];

  if (artist_status.toLowerCase() !== "all") {
    query += " WHERE a.Status = ?";
    params.push(artist_status);
  }

  query += " ORDER BY ArtistID ASC";

  connection.query(query, params, function (error, results) {
    if (error) throw error;
    res.send({ error: false, data: results, message: `Artists with status "${artist_status}" found!` });
  });
});
```

Method: GET
URL:http://localhost:3031/artist/status/

Receive a parameter which is **artist_status** and check if it matches the given status or not, when the status parameter equals "all", it returns all artists regardless of status, The order of results will be ascending by ArtistID.

search by Name + Status

```
// Search by name + status
router.get('/namestatus/:name/:status', (req, res) => {
  let artist_name = req.params.name;
  let artist_status = req.params.status;
  if (!artist_name) {
    return res.status(400).send({ error: true, message: 'Please provide name' });
  }
  let query = `
  SELECT
    a.*,
    c.C_Name AS ArtistCategory
  FROM
    Artist a
  JOIN
    Category c ON a.ACID = c.CID
  WHERE
    a.ArtistName LIKE ?
`;
  let params = [`%${artist_name}%`];
  if (artist_status.toLowerCase() !== "all") {
    query += " AND a.Status = ?";
    params.push(artist_status);
  }
  // if status is all  don't filter by status (show all statuses)
  connection.query(query, params, function (error, results) {
    if (error) throw error;
    res.send({ error: false, data: results, message: `Artists found with name "${artist_name}"` });
  });
});
```

Method: GET
URL:http://localhost:3031/artist/namestatus/

Receive two parameters which are **artist_status** , **artist_name** and check if it matches the given status and name or not.

search by Base price + Status

```
// Search by status + baseprice
router.get('/statusbaseprice/:status/:baseprice', (req, res) => {
  let artist_status = req.params.status;
  let artist_baseprice = parseFloat(req.params.baseprice);
  if (isNaN(artist_baseprice)) {
    return res.status(400).send({ error: true, message: 'Please provide provide a valid base price' });
  }
  let query = `
SELECT
    a.*,
    c.C_Name AS ArtistCategory
  FROM
    Artist a
  JOIN
    Category c ON a.ACID = c.CID
  WHERE
    a.BasePrice >= ?`;
  let params = [artist_baseprice];

  if (artist_status.toLowerCase() !== "all") {
    query += " AND a.Status = ?";
    params.push(artist_status);
  }

  query += " ORDER BY a.BasePrice ASC";
  connection.query(query, params, function (error, results) {
    if (error) {
      return res.status(500).send({ error: true, message: 'Database error' });
    }
    res.send({
      error: false,
      data: results,
      message: results.length > 0 ? 'Artists found!' : 'No artists found matching your criteria.'
    });
  });
});
```

Method: GET
URL:http://localhost:3031/artist/statusbaseprice/

Receive two parameters which are **artist_status** , **artist_baseprice** and check if it matches the given status and baseprice or not.

search by Category + Status

```
// Search by status + category
router.get('/statuscategory/:status/:category', (req, res) => {
  let artist_status = req.params.status;
  let artist_category = req.params.category;

  if (!artist_category) {
    return res.status(400).send({ error: true, message: 'Please provide both status and category' });
  }
  let query = `
    SELECT
      a.*,
      c.C_Name AS ArtistCategory
    FROM
      Artist a
    JOIN
      Category c ON a.ACID = c.CID
    WHERE
      a.ACID = ?
  `;
  let params = [artist_category];
  if (artist_status.toLowerCase() !== "all") {
    query += " AND a.Status = ?";
    params.push(artist_status);
  }
  query += " ORDER BY a.ArtistName ASC";
  connection.query(query, params, function (error, results) {
    if (error) {
      return res.status(500).send({ error: true, message: 'Database error' });
    }
    res.send({
      error: false,
      data: results,
      message: results.length > 0 ? 'Artists found!' : 'No artists found matching your criteria.'
    });
  });
});
```

Method: GET
URL:http://localhost:3031/artist/statuscategory/

Receive two parameters which are **artist_status** , **artist_category** and check if it matches the given status and category or not.

search by Status + Category + Name

```
// Search by status + category + name
router.get('/statuscategoryname/:status/:category/:name', (req, res) => {
  let artist_status = req.params.status;
  let artist_category = req.params.category;
  let artist_name = req.params.name;

  if (!artist_category || !artist_name) {
    return res.status(400).send({ error: true, message: 'Please provide status, category, and name' });
  }
  let query = `
    SELECT
      a.*,
      c.C_Name AS ArtistCategory
    FROM
      Artist a
    JOIN
      Category c ON a.ACID = c.CID
    WHERE
      a.ACID = ?
      AND a.ArtistName LIKE ?
  `;
  let params = [artist_category, `%${artist_name}%`];
  if (artist_status.toLowerCase() !== "all") {
    query += " AND a.Status = ?";
    params.push(artist_status);
  }
  query += " ORDER BY a.ArtistName ASC";
  connection.query(query, params, function (error, results) {
    if (error) {
      return res.status(500).send({ error: true, message: 'Database error' });
    }
    res.send({
      error: false,
      data: results,
      message: results.length > 0 ? 'Artists found!' : 'No artists found matching your criteria.'
    });
```

Method: GET
URL:http://localhost:3031/artist/statuscategoryname/

Receive three parameters which are **artist_status** , **artist_category, artist_name** and check if it matches the given status, category and name or not.

search by Status + Name + Base price

```
// Search by status + name + baseprice
router.get('/statusnamebaseprice/:status/:name/:baseprice', (req, res) => {
  let artist_status = req.params.status;
  let artist_name = req.params.name;
  let artist_baseprice = parseFloat(req.params.baseprice);
  if (!artist_name || isNaN(artist_baseprice)) {
    return res.status(400).send({ error: true, message: 'Please provide name and a valid base price' });
  }
  let query = `
    SELECT
      a.*,
      c.C_Name AS ArtistCategory
    FROM
      Artist a
    JOIN
      Category c ON a.ACID = c.CID
    WHERE
      a.ArtistName LIKE ?
      AND a.BasePrice >= ?
  `;
  let params = [`%${artist_name}%`, artist_baseprice];
  if (artist_status.toLowerCase() !== "all") {
    query += " AND a.Status = ?";
    params.push(artist_status);}
  query += " ORDER BY a.BasePrice ASC";
  connection.query(query, params, function (error, results) {
    if (error) {
      return res.status(500).send({ error: true, message: 'Database error' });
    }
    res.send({
      error: false,
      data: results,
      message: results.length > 0 ? 'Artists found!' : 'No artists found matching your criteria.'
    });
  });
});
```

Method: GET
URL:http://localhost:3031/artist/statusnamebaseprice/

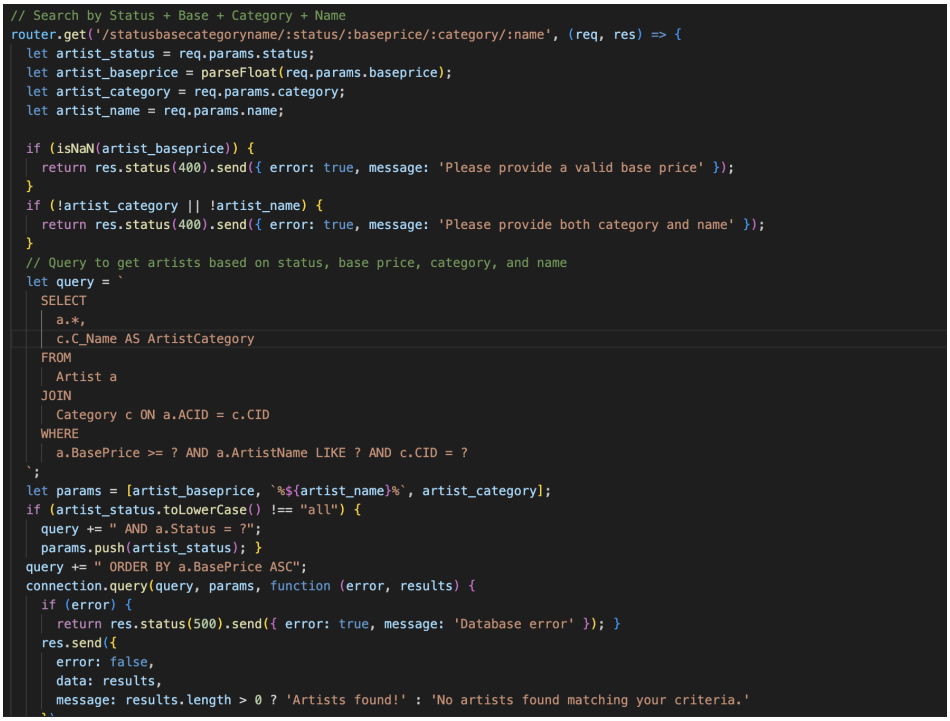
Receive three parameters which are **artist_status** , **artist_baseprice,** **artist_name** and check if it matches the given status, baseprice and name or not.

search by Status + Base price + Category

```
// Search by status + baseprice + category
router.get('/statusbasecategory/:status/:baseprice/:category', (req, res) => {
  let artist_status = req.params.status;
  let artist_baseprice = parseFloat(req.params.baseprice);
  let artist_category = req.params.category;
  if (!artist_category || isNaN(artist_baseprice)) {
    return res.status(400).send({ error: true, message: 'Please provide a valid base price and category' });
  }
  let query = `
    SELECT
      a.*,
      c.C_Name AS ArtistCategory
    FROM
      Artist a
    JOIN
      Category c ON a.ACID = c.CID
    WHERE
      a.BasePrice >= ?
      AND c.C_Name = ?
  `;
  let params = [artist_baseprice, artist_category];
  if (artist_status.toLowerCase() !== "all") {
    query += " AND a.Status = ?";
    params.push(artist_status); }
  query += " ORDER BY a.BasePrice ASC";
  connection.query(query, params, function (error, results) {
    if (error) {
      return res.status(500).send({ error: true, message: 'Database error' });
    }
    res.send({
      error: false,
      data: results,
      message: results.length > 0 ? 'Artists found!' : 'No artists found matching your criteria.'
    });
  });
});
```

Method: GET
URL:http://localhost:3031/artist/statusbasecategory/

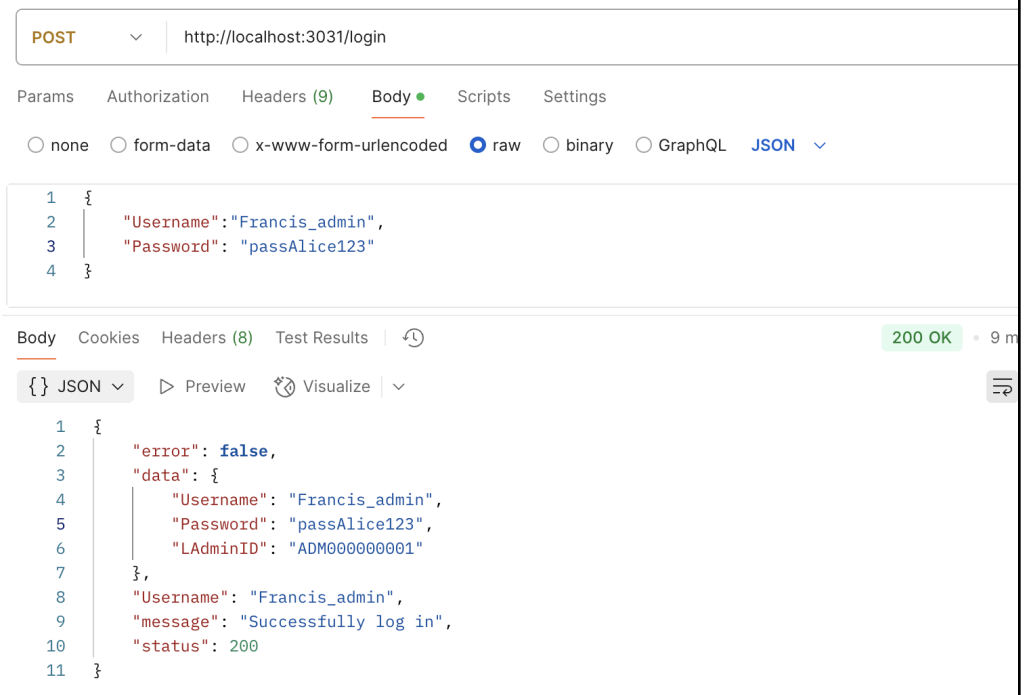
Receive three parameters which are **artist_status** , **artist_baseprice,** **artist_category** and check if it matches the given status, baseprice and category or not.

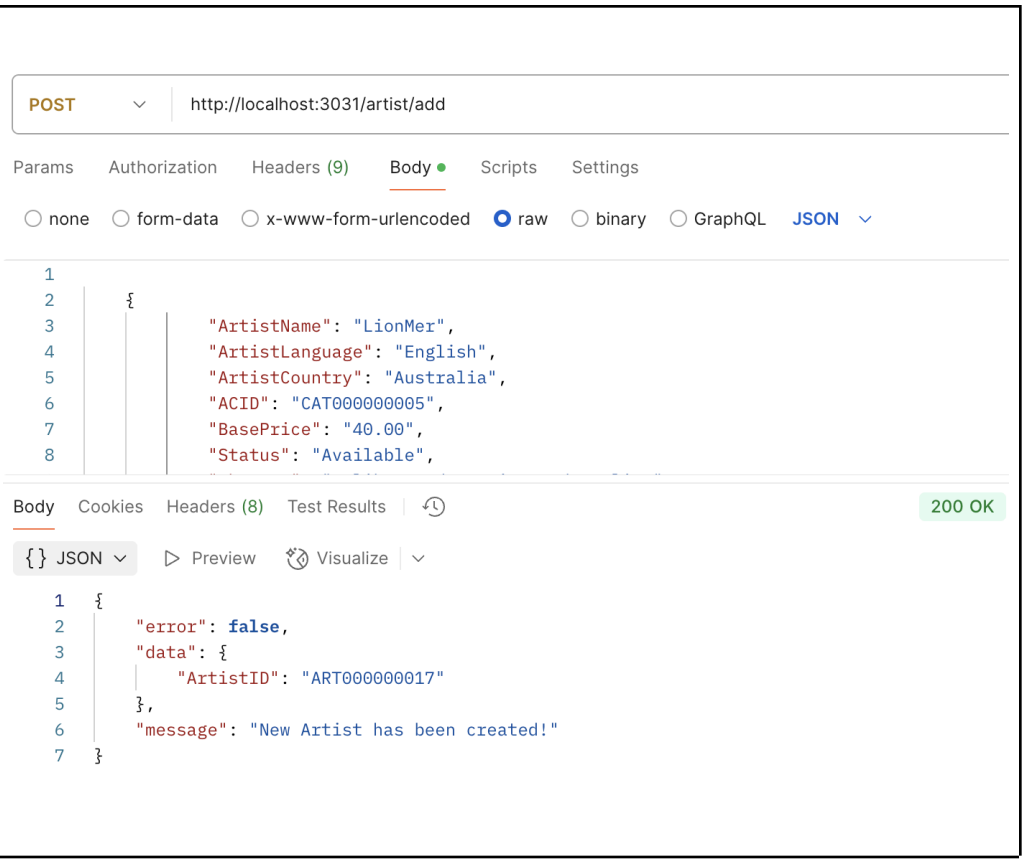
search by Status + Category + Name + Base price	
	Method: GET URL:http://localhost:3031/artist/statusbasecategoryname/ Receive four parameters which are **artist_status** , **artist_baseprice, artist_category, artist_name** and check if it matches the given status, baseprice, name and category or not.

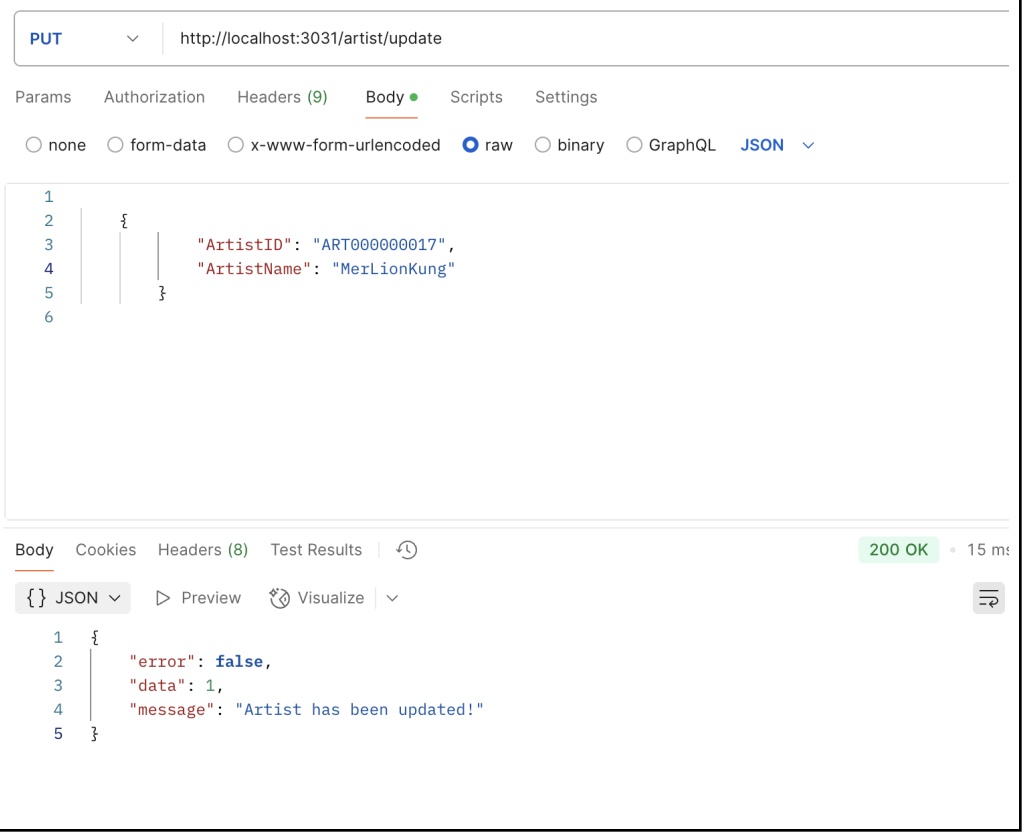
```
// Search by Status + Base + Category + Name
router.get('/statusbasecategoryname/:status/:baseprice/:category/:name', (req, res) => {
  let artist_status = req.params.status;
  let artist_baseprice = parseFloat(req.params.baseprice);
  let artist_category = req.params.category;
  let artist_name = req.params.name;

  if (isNaN(artist_baseprice)) {
    return res.status(400).send({ error: true, message: 'Please provide a valid base price' });
  }
  if (!artist_category || !artist_name) {
    return res.status(400).send({ error: true, message: 'Please provide both category and name' });
  }
  // Query to get artists based on status, base price, category, and name
  let query = `
    SELECT
      a.*,
      c.C_Name AS ArtistCategory
    FROM
      Artist a
    JOIN
      Category c ON a.ACID = c.CID
    WHERE
      a.BasePrice >= ? AND a.ArtistName LIKE ? AND c.CID = ?
  `;
  let params = [artist_baseprice, `%${artist_name}%`, artist_category];
  if (artist_status.toLowerCase() !== "all") {
    query += " AND a.Status = ?";
    params.push(artist_status); }
  query += " ORDER BY a.BasePrice ASC";
  connection.query(query, params, function (error, results) {
    if (error) {
      return res.status(500).send({ error: true, message: 'Database error' }); }
    res.send({
      error: false,
      data: results,
      message: results.length > 0 ? 'Artists found!' : 'No artists found matching your criteria.'
```

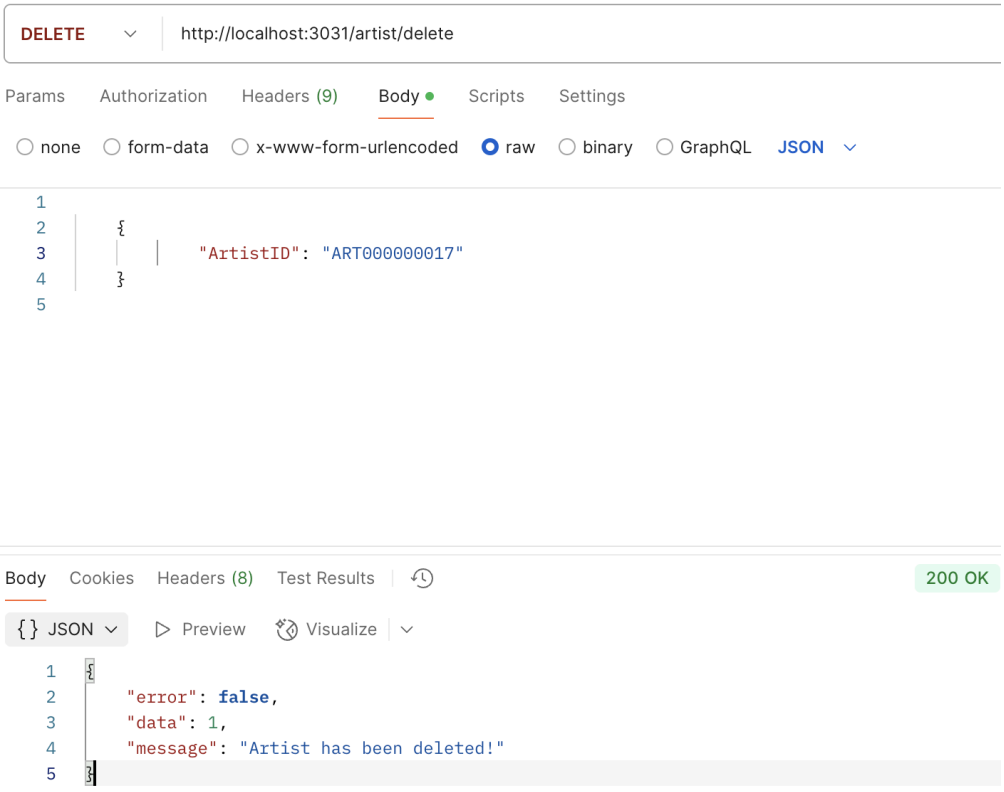
Testing results of web services

Test on **POSTMAN**

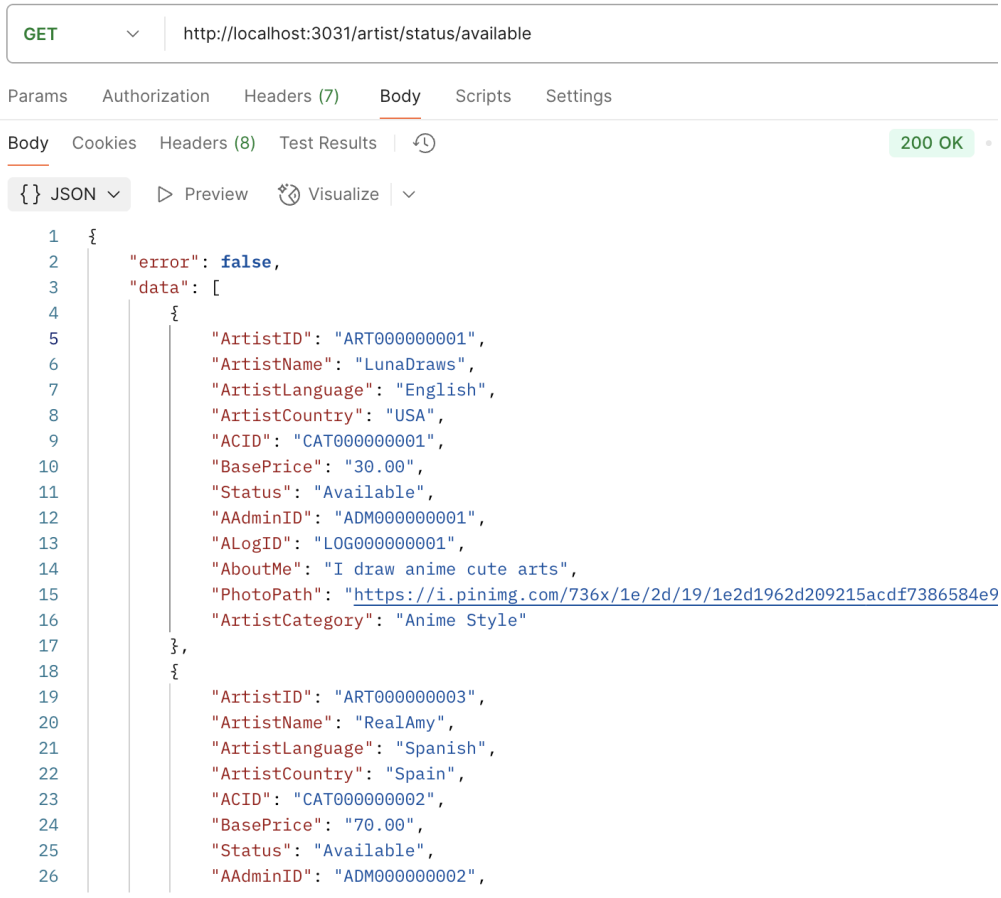
log in [POST]	

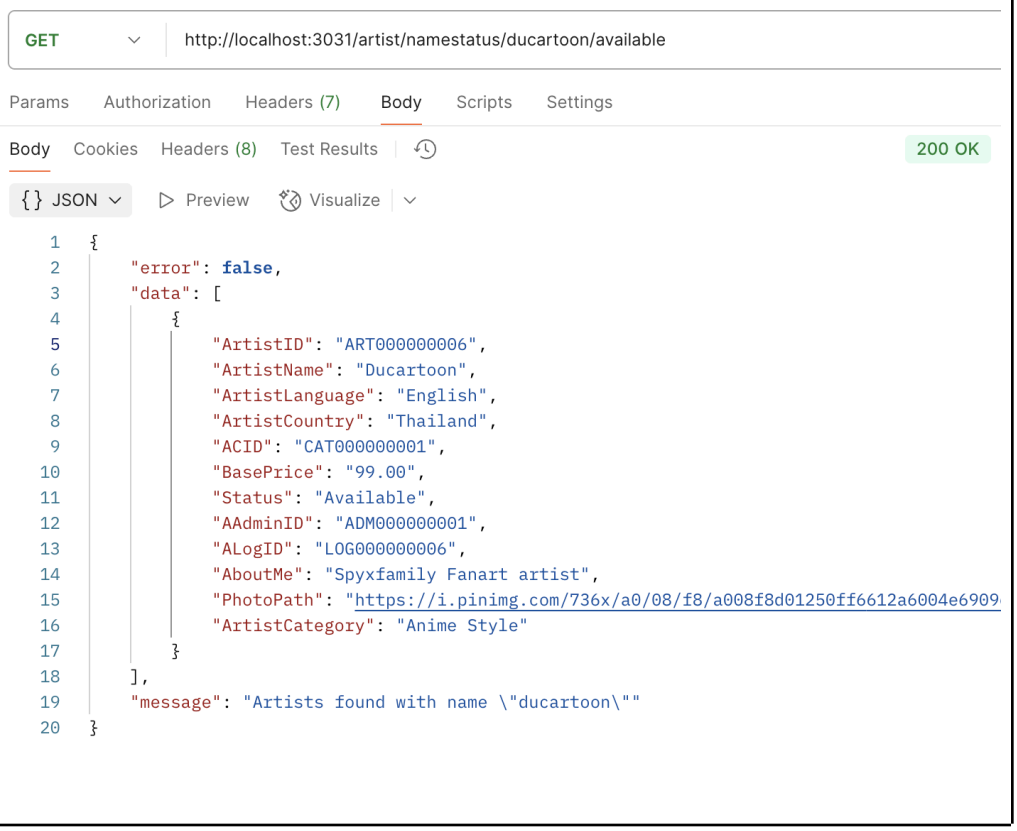
Add artist [POST]	

Update artist [PUT]	

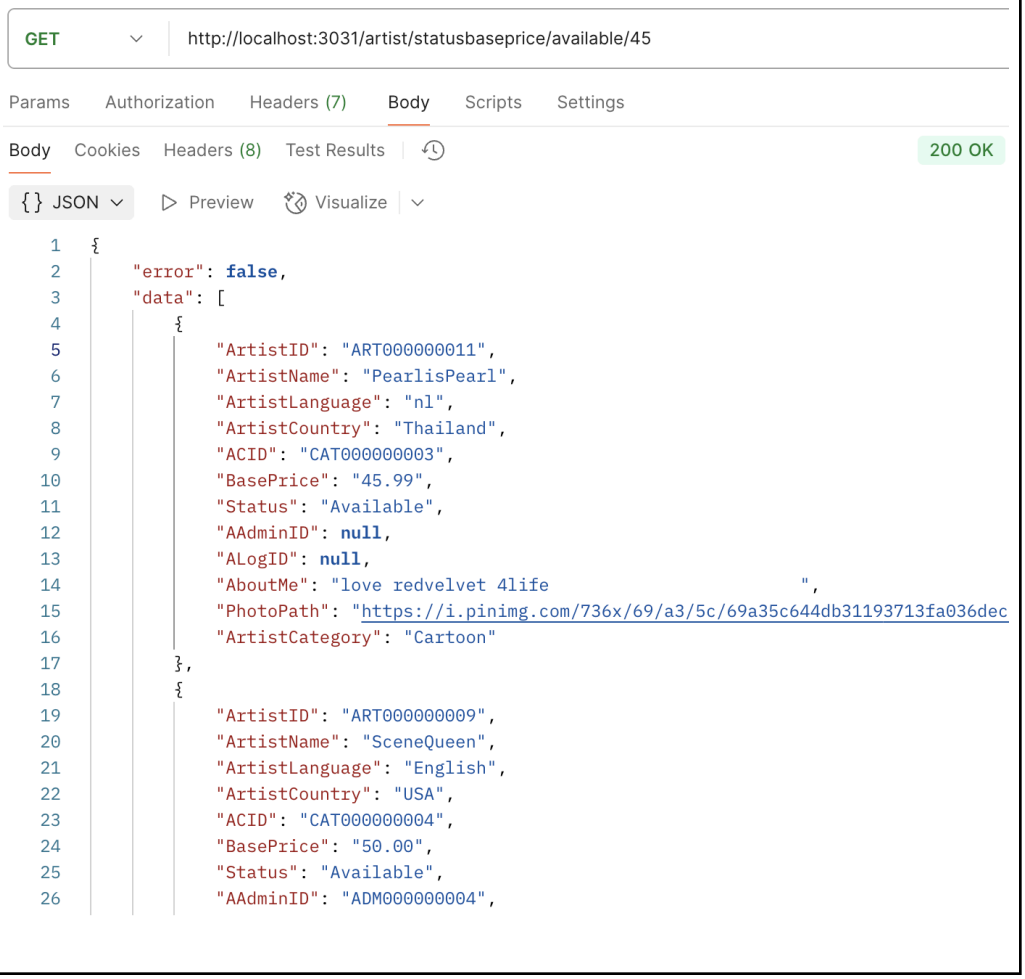
Delete artist [DELETE]	

search by Status

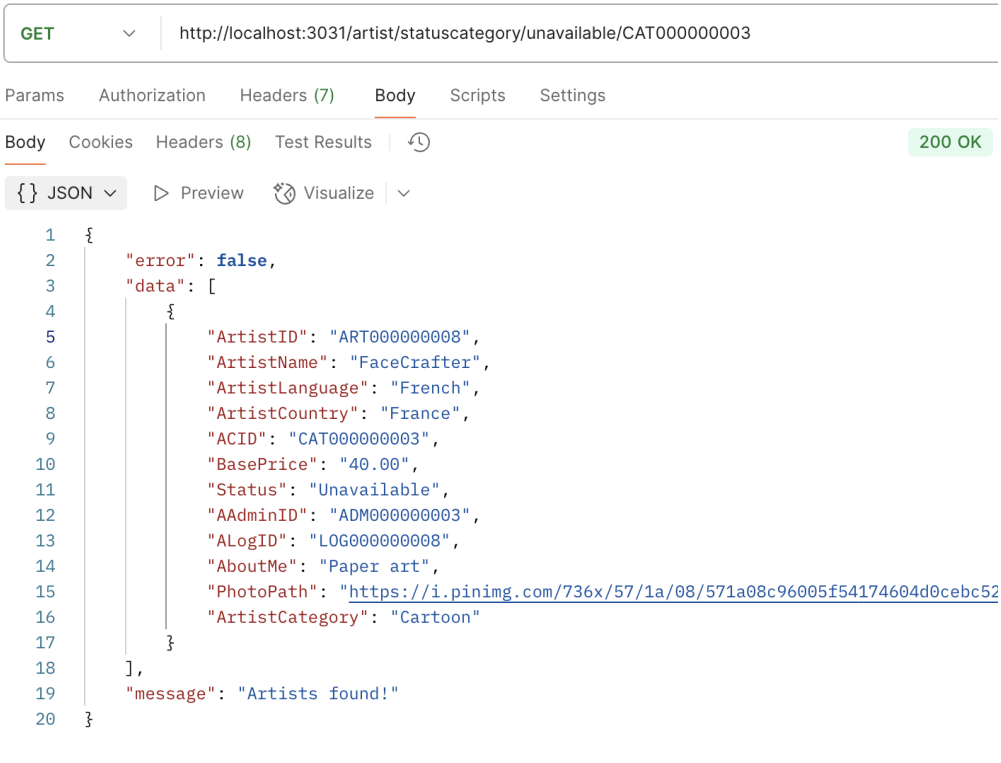


search by Name + Status	

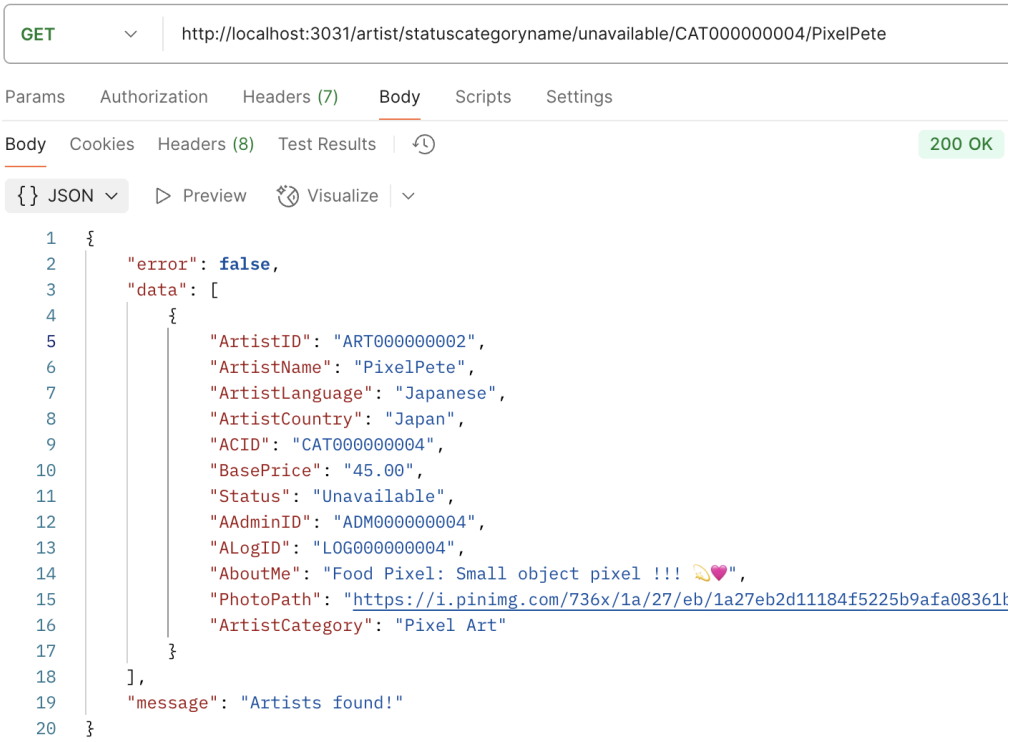
search by Base price + Status

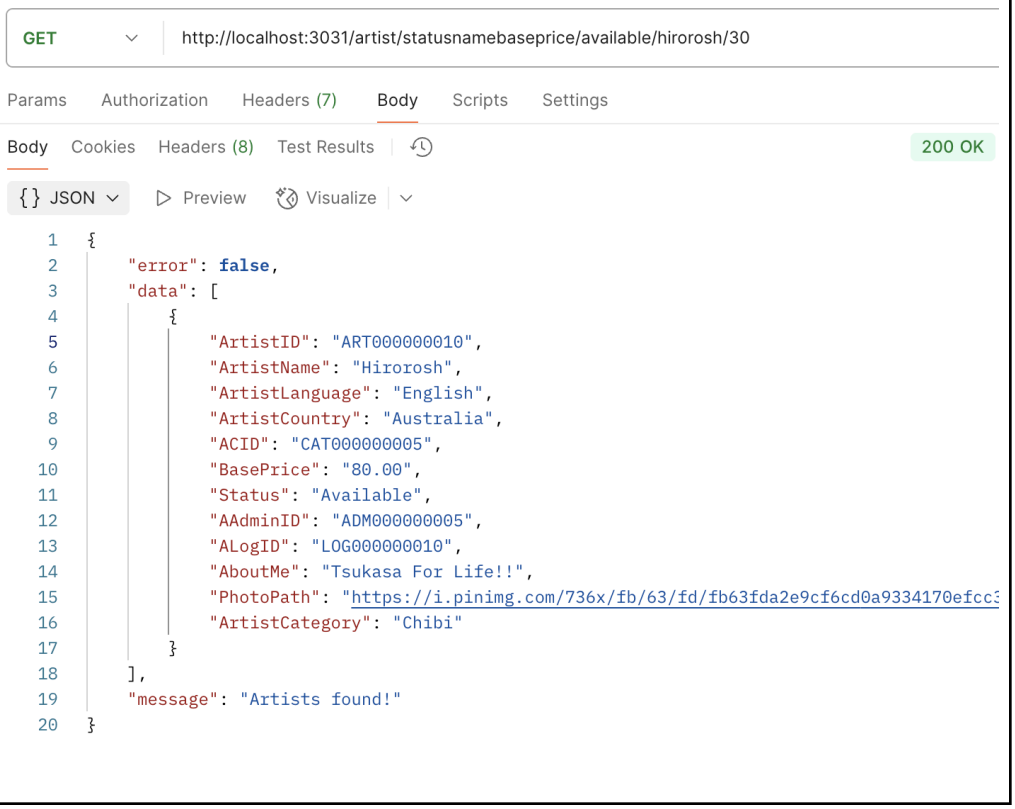


search by Category + Status

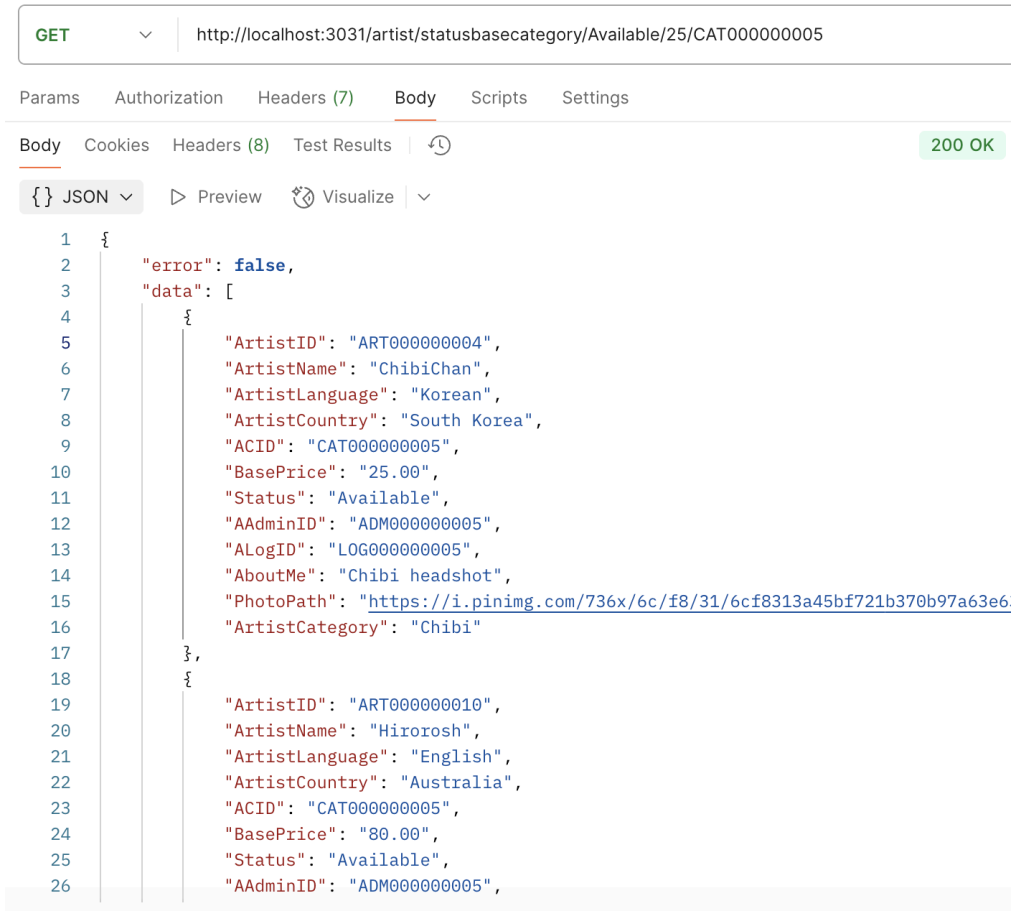


search by Status + Category + Name



search by Status + Name + Base price	

search by Status + Category + Base price



search by Status + Category + Name + Base price