Documentation of API

User

- GET /users
- GET /users/:id
- POST /users

Photo

- GET /photos
- GET /photos/:id
- POST /photos

GET /Users

List all users with simple information (ID and username)

Endpoint URL

http://localhost:3000/users

Example code (fetch)

```
function load() {
  fetch("http://localhost:3000/users")
    .then((response) => {
      if (!response.ok) {
        throw new Error(`Network response was not ok :${response.status}`);
      }
      return response.json();
    })
    .then((data) => {
      const hiddenCols = document.getElementsByClassName("hiddencol");
      for (let i = 0; i < hiddenCols.length; i++) {
        hiddenCols[i].style.display = "none";
      }
      var out = "";
      for (var i = 0; i < data.length; i++) {
        out +=
          '<tr><th scope="row">' +
          data[i].id +
          "</th><td>" +
          data[i].username +
          "</td></tr>";
      }
      document.getElementById(“username_tb").innerHTML = out;
    })
    .catch((error) => {
      console.error("Fetch error:", error);
    });
}
```

Example code (server)

```
app.get("/users", (req, res) => {
  const list_user = [];
  for (let i = 0; i < data.length; i++) {
    const user = data[i];
    list_user.push({ id: user.id, username: user.username });
  }
  res.json(list_user);
});
```

Example responses

```
[
  {
    "id": 1,
    "username": "John_doe",
     },
  {
    "id": 2,
    "username": "Mary_smith",
     }
]
```

Response fields

NAME	Type	DESCRIPTION
id	string	A unique identifier for this user. This identifier is returned as a string
username	string	The username of this user.

GET /Users/:id

Enter the id of the user you want to query, and specific information about the user will be displayed (id, username, date the user was created, number of images uploaded).

Endpoint URL

http://localhost:3000/users/:id

Path parameters

NAME	Type	DESCRIPTION
id	string	The user ID you are searching for.

Example code (fetch)

```
function load_id() {
  const userid = document.getElementById("username_input").value;
  const URL = `http://localhost:3000/users/${userid}`;
  fetch(URL)
    .then((response) => {
      if (!response.ok) {
        throw new Error(`Network response was not ok :${response.status}`);
      }
      return response.json();
```

```
    })
    .then((data) => {
      const hiddenCols = document.getElementsByClassName("hiddencol");
      for (let i = 0; i < hiddenCols.length; i++) {
        hiddenCols[i].style.display = "table-cell";
      }
      var out =
        '<tr><th scope="row">' +
        data.id +
        "</th><td>" +
        data.username +
        "</td><td>" +
        data.date +
        "</td><td>" +
        data.photoCount +
        "</td></tr>";
      document.getElementById("username_tb").innerHTML = out;
    })
    .catch((error) => {
      console.error("Fetch error:", error);
    });
}
```

Example code (server)

```
app.get("/users/:id", (req, res) => {
  const ID = parseInt(req.params.id);
  let user = null;
  for (let i = 0; i < data.length; i++) {
    if (data[i].id === ID) {
      user = data[i];
      break;
    }
  }
  if (user) {
    const userInfo = {
      id: user.id,
      username: user.username,
      date: user.account_created_date,
      photoCount: user.photos.length,
    };
    res.json(userInfo);
  } else {
    res.status(400).json({ error: "User not found" });
  }
});
```

Example responses

```
{
    "id": 1,
    "username": "John_doe",
    "account_created_date": "19/07/2021",
    "number of photos”: user.photos.length
  }
```

Response fields

NAME	Type	DESCRIPTION
id	string	A unique identifier for this user. This identifier is returned as a string
username	string	The username of this user.
account_created_date	string	Date of user creation
number of photos	integer	Number of photos uploaded by this user

---

POST /users

Takes the entered user name (username) and the date of the day (account_created_date) and the automatically generated user ID (id) are added to the system as a new user.

Endpoint URL

http://localhost:3000/users

Example code (fetch)

```
function getCurrentDate() {
  const today = new Date();
  const yyyy = today.getFullYear();
  let mm = today.getMonth() + 1;
  let dd = today.getDate();
  return dd + "/" + mm + "/" + yyyy;
}

function add_user() {
  const username = document.getElementById("add").value;
  const accountCreatedDate = getCurrentDate();
  const userData = {
    username: username,
    account_created_date: accountCreatedDate,
  };
  fetch("http://localhost:3000/users", {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
    },
    body: JSON.stringify(userData),
  })
    .then((response) => {
      if (!response.ok) {
        throw new Error(`Network response was not ok: ${response.status}`);
      }
      return response.json();
    })
    .then((data) => {
      console.log("New user added:", data);
      const table = document.getElementById("username_tb");
      const newUserRow = table.insertRow();
      newUserRow.innerHTML =
        "<tr><td>" + data.id + "</td><td>" + data.username + "</td></tr>";
```

```
    })
    .catch((error) => {
      console.error("Add user error:", error);
    });
}
```

Example code (server)

```
app.use(express.json());
app.post("/users", (req, res) => {
 const ID = data.length + 1;
  const newUser = req.body;
  newUser.id = ID;
   const newuser = {
    id: newUser.id,
    username: newUser.username,
    account_created_date: newUser.account_created_date,
    photos: [],
  };
  data.push(newuser);
  fs.writeFile("./data.json", JSON.stringify(data, null, 2), (err) => {
    if (err) {
      console.error("Error writing JSON file:", err);
      res.status(500).json({ error: "Internal server error" });
    } else {
      console.log("JSON file updated successfully");
      res.status(200).json(newUser);
    }
  });
});
```

Example responses

```
{
    "id": 4,
    "username": "gqibdi",
    "account_created_date": “21/04/2024“,
    "photos”: [ ],
  }
```

Response fields

NAME	Type	DESCRIPTION
id	string	Auto-generated user id. Equal to max user ID+1
username	string	The username that was entered
account_created_date	date	The date on which the account was created
photos	array	Empty array where the added details of images will appear.

GET /photos

List all photos with simple information (ID, topic of photo and username that created this photo)

Endpoint URL

http://localhost:3000/photos

Example code (fetch)

```
function load_photos() {
  fetch("http://localhost:3000/photos")
    .then((response) => {
      if (!response.ok) {
        throw new Error(`Network response was not ok :${response.status}`);
      }
      return response.json();
    })
    .then((data) => {
      document.getElementById("photo_list").style.display = "table";
      document.getElementById("details").style.display = "none";
      var out = "";
      for (var i = 0; i < data.length; i++) {
        out +=
          '<tr onclick="load_photos_id(' +
          data[i].id +
          ')"><th scope="row">' +
          data[i].id +
          "</th><td>" +
          data[i].username +
          "</td><td>" +
          data[i].topic +
          "</td></tr>";
      }
      document.getElementById("photo_tb").innerHTML = out;
    })
    .catch((error) => {
      console.error("Fetch error:", error);
    });
}
```

Example code (server)

```
app.get("/photos", (req, res) => {
  const list_photos = [];
  for (let i = 0; i < data.length; i++) {
    const user = data[i];
    for (let j = 0; j < user.photos.length; j++) {
      const photo = user.photos[j];
      list_photos.push({
        id: photo.id,
        topic: photo.topic,
        username: user.username,
      });
    }
  }
  res.json(list_photos);
});
```

Example responses

```
{
   "id": 1,
   "username": "John_doe",
   "topic": "City A",
 }
```

Response fields

NAME	Type	DESCRIPTION
id	string	A unique identifier for this photo. This identifier is returned as a string
username	string	Username of the user who added this image
Topic	String	Topic of the image

GET /Photos/:id

Enter the id of the photo you want to query, and specific information about the photo will be displayed (id, username, image, date, topic and comments).

Endpoint URL

http://localhost:3000/users/:id

Path parameters

NAME	Type	DESCRIPTION
id	string	The photo ID you are searching for.

Example code (fetch)

```
function load_photos_id(id) {
  const URL = `http://localhost:3000/photos/${id}`;
  fetch(URL)
    .then((response) => {
      if (!response.ok) {
        if (response.status === 404) {
          alert("The user did not upload any photo");
        } else {
          throw new Error(`Network response was not ok :${response.status}`);
        }
      }
      return response.json();
    })
    .then((data) => {
      document.getElementById("photo_list").style.display = "none";
      document.getElementById("details").style.display = "inline";
      var out =
        '<div class="col"><img src="' +
```

```
        data.url +
        '" alt="' +
        data.topic +
        '"></div><div class="col"><p class="fs-4">' +
        data.topic +
        '</p><p class="fs-5">' +
        data.username +
        '</p><p class="fs-5">' +
        data.date +
        '</p><p class="fs-6">' +
        data.comments +
        "</p></div>";
      document.getElementById("photo_row").innerHTML = out;
    })
    .catch((error) => {
      console.error("Fetch error:", error);
    });
}
```

Example code (server)

```
app.get("/photos/:id", (req, res) => {
  const ID = parseInt(req.params.id);
  let photo = null;
  for (let i = 0; i < data.length; i++) {
    const user = data[i];
    for (let j = 0; j < user.photos.length; j++) {
      if (String(user.photos[j].id) === String(ID)) {
        photo = user.photos[j];
        username = user.username;
        break;
      }
    }
  }
  if (photo) {
    const photoInfo = {
      id: photo.id,
      url: photo.url,
      username: username,
      date: photo.date,
      topic: photo.topic,
      comments: photo.comments,
    };
    res.json(photoInfo);
  } else {
    res.status(400).json({ error: "Photo not found" });
  }
});
```

Example responses

```
 {
    "id": 2,
    "url": "photo/2.jpeg",
    "username": “Mary_smith",
    "date": "07/07/2023",
    "topic": "City B",
    "comments": "Another comment here."
     }
```

Response fields

NAME	Type	DESCRIPTION
id	string	A unique identifier for this photo. This identifier is returned as a string
Url	Url	The url of the image, which will appear as an image when displaying specific image information
Date	string	The date when the photo was taken, this is the information you need to provide when adding a photo, not the automatic date when adding a photo
username	string	Username of the user who added the photo
topic	string	Topic of the image
comments	String	comments of the image

POST /photos

Add the entered URL, photo subject, comment, date, the user who chose to upload the image, and the auto-generated photo ID (id) to the photo array in the user's data as a new photo.

Endpoint URL

http://localhost:3000/photos

Example code (fetch)

```
function add_photos() {
  const username = document.getElementById("add_username").value;
  const topic = document.getElementById("add_topic").value;
  const date = document.getElementById("add_date").value;
  const url = document.getElementById("add_url").value;
  const comments = document.getElementById("add_comments").value;
  console.log(username)
  if (username === "dropdown_value") {
    alert("Please choose a username.");
    return;
  }
  const photoInfo = {
    username: username,
    url: url,
    date: date,
    topic: topic,
    comments: comments,
  };
  console.log(photoInfo)
```

```
  fetch("http://localhost:3000/photos", {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
    },
    body: JSON.stringify(photoInfo),
  })
    .then((response) => {
      if (!response.ok) {
        throw new Error(`Network response was not ok: ${response.status}`);
      }
      return response.json();
    })
    .then((data) => {
      console.log("New photo added");
      alert("New photo added");
    })
    .catch((error) => {
      console.error("Add photo error:", error);
    });
}
```

Example code (server)

```
app.use(express.json());
app.post("/photos", (req, res) => {
  let maxID = 0;
  data.forEach((user) => {
    user.photos.forEach((photo) => {
      if (photo.id > maxID) {
        maxID = photo.id;
      }
    });
  });
  const ID = maxID + 1;
  const newPhoto = req.body;
  const username = newPhoto.username;
  newPhoto.id = ID;
  const newphoto = {
    id: newPhoto.id,
    url: newPhoto.url,
    date: newPhoto.date,
    topic: newPhoto.topic,
    comments: newPhoto.comments,
  };
  data.forEach((user) => {
    if (user.username === username) {
      console.log(newphoto)
      user.photos.push(newphoto);
    }
  });
  fs.writeFile("./data.json", JSON.stringify(data, null, 2), (err) => {
    if (err) {
      console.error("Error writing JSON file:", err);
      res.status(500).json({ error: "Internal server error" });
    } else {
      console.log("JSON file updated successfully");
      res.status(201).json(newPhoto);
    }
  });
```

});

Example responses

```
{
  id: 5,
  url: 'https://cdn.pixabay.com/photo/2016/07/07/16/46/dice-1502706_640.jpg',
  date: '11/12/2000',
  topic: 'dice',
  comments: 'co'
}
```

Response fields

NAME	Type	DESCRIPTION
id	string	Auto-generated photo id. Equal to max photo ID+1
url	url	The url that was entered
date	string	The date that was entered
topic	String	The topic that was entered
comments	String	The comments that was entered