You asked:

I have built a small project to track the scores of the Algerian national football team, but I had some problems with the database, so I decided to convert from Firebase Realtime Database to Supabase. However, I am now facing more issues, which are worse than before, so I am moving to Appwrite. Can you help me with this?

The AI replied:

Absolutely, I can help you transition your project to **Appwrite** and make sure everything works smoothly. Since you’ ve already been through Firebase and Supabase, you're likely familiar with general backend/database concepts. Appwrite is similar but has its own way of doing things.

Let’ s break this down into a clear migration and setup process.

Migration Overview (Firebase ☒ Supabase ☒ Appwrite)

1. **Understand your data model**: What collections/tables and fields did you use?

2. **Prepare Appwrite backend**:

   - Set up your Appwrite project

   - Configure authentication (if used)

   - Create databases and collections

3. **Migrate your data**

4. **Update your app code** to use Appwrite SDK

Step-by-Step: Moving to Appwrite

Step 1: Set Up Appwrite

1. **Install Appwrite (locally or cloud):**

   - Use Appwrite Cloud for simplicity: https://cloud.appwrite.io

   - Or run locally with Docker if needed.

2. **Create a Project**

- Go to the Appwrite Console

- Click **"New Project"**, give it a name (e.g., “Algeria Scores Tracker”)

3. **Add a Web/Flutter/React Native app** depending on your stack to get the Appwrite SDK credentials.

Step 2: Create Database & Collections

1. **Create a Database** under your project

- Name: `scores_tracker`

2. **Create a Collection** (like a table)

- Name: `matches`

- Fields:

  - `date` (type: date)

  - `opponent` (type: string)

  - `algeria_score` (int)

  - `opponent_score` (int)

  - `location` (string)

  - Add more if needed (e.g., tournament name)

3. **Set Permissions**

- Allow read/write access depending on your app (public, logged-in users, or roles)

Step 3: Migrate Data

If you have existing data (e.g., in JSON or CSV):

- Use the **Appwrite Functions** or directly the Appwrite SDK in a script to push the data.

- Example in Node.js:

```
const sdk = require("node-appwrite");

const client = new sdk.Client()
    .setEndpoint('https://cloud.appwrite.io/v1')
    .setProject('YOUR_PROJECT_ID')
    .setKey('YOUR_API_KEY');

const database = new sdk.Databases(client);

await database.createDocument('scores_tracker', 'matches', 'unique()', {
  date: '2025-06-20',
  opponent: 'Egypt',
  algeria_score: 2,
  opponent_score: 1,
  location: 'Algiers'
});
```

Step 4: Update Frontend Code

Use Appwrite SDK:

- Web: `npm install appwrite`

- Flutter: `flutter pub add appwrite`

Example (Web/JavaScript):

```
import { Client, Databases } from "appwrite";

const client = new Client()
    .setEndpoint("https://cloud.appwrite.io/v1")
    .setProject("YOUR_PROJECT_ID");

const databases = new Databases(client);

databases.listDocuments('scores_tracker', 'matches')
  .then(response => {
    console.log(response.documents);
});
```

Step 5: Test Everything

- Add a match
- View all matches
- Update/delete (if needed)
- Handle errors gracefully (e.g., network, permissions)

Troubleshooting

Let me know:

- Your tech stack (Flutter, React, Vue, etc.)
- Specific errors you're running into with Appwrite (e.g., permissions, data not saving, API not responding)
- If you need help writing migration scripts from Supabase/Firebase

---

Would you like me to help you write a migration script or set up a collection structure JSON for import into Appwrite?

You asked:

I haven't even provided you with the code. Also, I'm not using any frameworks or libraries; it's just plain HTML, CSS and JavaScript. I'll send you the code in my next message, be patient, OK?

The AI replied:

Got it — no worries at all! Take your time and send over your code whenever you're ready. Once I see it, I’ ll help you integrate it with **Appwrite** using plain JavaScript — no frameworks needed.

Looking forward to your next message!

You asked:

# AlgVs. ### ./admin.html

```
HTML
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Alg Vs 25</title>

  <script src="https://cdn.jsdelivr.net/npm/@tailwindcss/browser@4"></script>
  <script type="module" src="./main.js"></script>
</head>

<body>
  <header class="w-full bg-slate-200 py-4 gap-4 flex items-center justify-center">
    <a class="text-sky-500 underline font-bold" href="/">Home Page</a>
    <a class="text-sky-500 underline font-bold" href="/admin.html">Admin Page</a>
  </header>

  <fieldset class="w-96 border-2 border-black rounded-sm mx-auto mt-4">
    <legend class="ml-4 bg-black text-white rounded-sm px-2 py-0.5">New Game</legend>

    <form autocomplete="off" class="flex flex-col gap-2 p-2">
      <div class="grid">
        <label for="date">Date</label>
        <input type="date" name="date" id="date" class="border border-black rounded-sm px-2 py-0.5" />
      </div>
      <div class="grid">
        <label for="game type">Game Type</label>
        <select name="game type" id="game type" class="border border-black rounded-sm px-2 py-0.5">
          <option hidden selected>-- Game Type --</option>
          <option value="International Friendly">International Friendly</option>
          <option value="Africa Cup of Nations">Africa Cup of Nations</option>
          <option value="Africa Cup of Nations Qualifying">Africa Cup of Nations Qualifying</option>
          <option value="FIFA World Cup Qualifying - CAF">FIFA World Cup Qualifying - CAF</option>
          <option value="FIFA World Cup">FIFA World Cup</option>
        </select>
      </div>
      <div class="grid">
        <label for="coach">Coach</label>
        <input type="text" name="coach" id="coach" value="Vladimir Petkovic"
          class="border border-black rounded-sm px-2 py-0.5" />
      </div>
      <div class="grid gap-2">
        <span>Score</span>
        <div class="flex gap-2">
          <input type="text" name="Opponent_01" id="Opponent_01" value="ALGERIA"
```

```
            class="border border-black rounded-sm px-2 py-0.5 w-[70%]" />
          <input type="number" name="Score_01" id="Score_01" placeholder="Alg Score"
            class="border border-black rounded-sm px-2 py-0.5  w-[30%]" />
        </div>
        <div class="flex gap-2">
          <input type="text" name="Opponent_02" id="Opponent_02" placeholder="Opponent"
            class="border border-black rounded-sm px-2 py-0.5  w-[70%]" />
          <input type="number" name="Score_02" id="Score_02" placeholder="Op Score"
            class="border border-black rounded-sm px-2 py-0.5  w-[30%]" />
        </div>
      </div>

      <div class="grid">
        <label for="Opponent Flag">Opponent Flag</label>
        <input type="url" name="Opponent Flag" id="Opponent Flag" placeholder="Opponent Flag"
          class="border border-black rounded-sm px-2 py-0.5" />
      </div>

      <button type="submit" class="bg-emerald-300 border border-black rounded-sm">Insert</button>
    </form>
  </fieldset>
</body>

</html>
```

./index.html

```
HTML
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Alg Vs 25</title>

  <script src="https://cdn.jsdelivr.net/npm/@tailwindcss/browser@4"></script>
  <script type="module" src="./main.js"></script>
</head>

<body>
  <h1 class="font-bold text-3xl text-center">The matches of the Algerian national team under the leadership of coach
    <mark class="bg-sky-300 text-nowrap px-2 rounded-sm">Vladimir Petkovic</mark>
  </h1>

  <table id="StatisticsTable" class="container mx-auto my-6 text-center text-xl">
    <thead>
      <tr>
        <th class="bg-emerald-400">Win</th>
        <th class="bg-rose-400">Lose</th>
        <th class="bg-amber-400">Draw</th>
        <th class="bg-sky-400">Scored</th>
        <th class="bg-violet-400">received</th>
      </tr>
    </thead>

    <tbody></tbody>
  </table>

  <table id="DataTable" class="container mx-auto mb-4">
    <thead>
      <tr class="border-y border-black">
        <th class="">Team 1</th>
        <th class=""></th>
        <th class="">Score </th>
        <th class=""></th>
        <th class="">Opponent</th>
      </tr>
    </thead>

    <tbody></tbody>
  </table>
</body>

</html>
```

./main.js

```
JS
import { createClient } from "https://cdn.jsdelivr.net/npm/@supabase/supabase-js/+esm";

const supabaseUrl = "";
const supabaseKey = "";
const supabase = createClient(supabaseUrl, supabaseKey);

function Timestamp(timestamp) {
  const date = new Date(timestamp);
  const options = { year: "numeric", month: "short", day: "numeric" };
  return date.toLocaleDateString("en-US", options);
}

function capitalizeWords(string) {
  if (typeof string !== "string" || string.trim() === "") return "";

  return string
    .toLowerCase() // Convert entire string to lowercase first
    .replace(/\b\w/g, char => char.toUpperCase()); // Capitalize first letter of each word
};

function transformObjectKeys(obj, mode = "lower", firstLetterOnly = false) {
  if (typeof obj !== "object" || obj === null) {
    throw new Error("Input must be an object.");
  }

  if (!["lower", "upper"].includes(mode)) {
    throw new Error("Mode must be either `lower` or `upper`.");
  }

  function transformKey(key) {
    if (firstLetterOnly) {
      return mode === "lower"
        ? key.charAt(0).toLowerCase() + key.slice(1)
        : key.charAt(0).toUpperCase() + key.slice(1);
    }
    return mode === "lower" ? key.toLowerCase() : key.toUpperCase();
  }

  function deepTransform(obj) {
    if (Array.isArray(obj)) {
      return obj.map(deepTransform);
    } else if (typeof obj === "object" && obj !== null) {
      return Object.fromEntries(
        Object.entries(obj).map(([key, value]) => [transformKey(key), deepTransform(value)])
      );
    }
    return obj;
  }

  return deepTransform(obj);
}

const CasesBg = {
  "win": "emerald",
  "loss": "rose",
  "draw": "amber"
}

console.log(window.document.location.pathname);


if (
  window.document.location.pathname === "/v2/"
  // ||
  // window.document.location.pathname === "/index.hmtl" ||
  // window.document.location.pathname.includes("./admin.html")
) {
  const DataTable = document?.querySelector("#DataTable tbody");
  const StatisticsTable = document?.querySelector("#StatisticsTable tbody");
  const AlgFlag = "https://gxdpkrwfrlqnxjiqxrnh.supabase.co/storage/v1/object/sign/algvs/countries-flags/image_62.png?
token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1cmwiOiJhbGd2cy9jb3VudHJpZXMtZmxhZ3MvaW1hZ2VfNjIucG5nIiwia
L0flus";

  function TR({ opponentname, opponentflag, opponentscore, algeriascore, date, coachname, event }) {
    const Case = algeriascore === opponentscore ? "draw" : algeriascore > opponentscore ? "win" : "loss";

    return `
      <tr class="bg-${CasesBg[Case]}-400">
        <td colspan="5" class="text-center">
```

```
          <span class="underline">${Timestamp(date)}</span> | ${event}
        </td>
      </tr>
      <tr class="bg-${CasesBg[Case]}-400 py-2 border-b-2 border-black text-xl text-center text-nowrap font-bold">
        <td class="">
          Algeria
        </td>
        <td class="">
          <img src="${AlgFlag}" class="block object-cover size-8 ml-auto" />
        </td>
        <td class="">
          ${algeriascore} - ${opponentscore}
        </td>
        <td class="">
          <img src="${opponentflag}" class="block object-cover size-8 mr-auto" />
        </td>
        <td class="text-left">
          ${opponentname}
        </td>
      </tr>`;
  }

  document.addEventListener("DOMContentLoaded", async event => {
    const { data, error } = await supabase.from("algvs").select("*").eq("coachname", "Vladimir Petkovic");

    if (error) {
      console.log(error);
      return
    }

    data?.reverse()?.map(item => DataTable.innerHTML += TR(item));

    const win = data.filter(item => item.opponentscore < item.algeriascore).length;
    const loss = data.filter(item => item.opponentscore > item.algeriascore).length;
    const draw = data.filter(item => item.opponentscore === item.algeriascore).length;

    const scored = data.map(i => i.algeriascore).map(i => Number(i)).reduce((a, b) => a + b);
    const received = data.map(i => i.opponentscore).map(i => Number(i)).reduce((a, b) => a + b);

    StatisticsTable.innerHTML = `<tr>
    <td>${win}</td>
    <td>${loss}</td>
    <td>${draw}</td>
    <td>${scored}</td>
    <td>${received}</td>
  </tr>`;
  });

} else if (window.document.location.pathname === "/admin.html") {
  function useFormsData(selector, keyType = "name") {
    if (typeof selector !== "string" && !(selector instanceof HTMLFormElement)) {
      throw new TypeError("useFormsData(): Selector must be a string or an HTMLFormElement");
    }
  // Get the form element
  const form = typeof selector === "string" ? document.querySelector(selector) : selector;

  if (!(form instanceof HTMLFormElement)) {
    throw new Error(`useFormsData(): No form found for selector "${selector}"`);
  }

  if (typeof keyType !== "string" || keyType.trim() === "") {
    throw new TypeError("useFormsData(): keyType must be a non-empty string");
  }

  const formData = new FormData(form);
  const dataObject = {};

  // Populate the object based on the key type
  for (const [name, value] of formData.entries()) {
    const inputElement = form.querySelector(`[name="${name}"]`);

    // Skip if no input is found (shouldn"t happen, but just in case)
    if (!inputElement) continue;

    const key = keyType === "name"
      ? inputElement.name
      : keyType === "id"
        ? inputElement.id
        : inputElement.getAttribute(keyType);

    if (!key) {
      throw new Error(`useFormsData(): Key type "${keyType}" not found on element "${name}"`);
    }

    dataObject[key] = value;
  }

  return dataObject;
}

const form = document?.querySelector("form");

form.addEventListener("submit", async event => {
  event.preventDefault();
  const snap = useFormsData(form);
  const toInsert = transformObjectKeys({
    // Id: Number(),
    Date: Timestamp(snap.date),
    OpponentName: capitalizeWords(String(snap.Opponent_02).toLowerCase()),
    OpponentFlag: String(snap["Opponent Flag"]),
    OpponentScore: Number(snap.Score_02),
    AlgeriaScore: Number(snap.Score_01),
    CoachName: String(snap.coach),
    Event: String(snap["game type"])
  });
```

```
    const { data, error } = await supabase.from('algvs').insert([toInsert]).select();

    if (error) {
      console.log(error);
      return;
    }

    console.log(data);
  })
}
```

The AI replied:

Thanks for sharing your full codebase — it’s well-structured and surprisingly clean for a plain JavaScript project! Let's now adapt it from **Supabase** to **Appwrite**.

Goal

Convert your current setup to use **Appwrite** instead of Supabase, without introducing frameworks or external libraries — just HTML, CSS, and JavaScript.

You asked:

Here is the code. Please analyse it and make the necessary changes and improvements, then send me the whole code. Don't just send pieces and tell me to complete them.

The AI replied:

Perfect. Based on your current setup, I’ve fully analyzed your code and converted it from using **Supabase** to using **Appwrite**, while also preserving your original architecture (plain HTML, CSS, and JavaScript). I've also improved a few things for readability and robustness without changing your project's character.

Migration Summary

Element	Supabase	Appwrite
`supabase-js` import	`@supabase/supabase-js`	Appwrite CDN SDK
Data storage	`supabase.from(...).select`	`databases.listDocuments()`
Insert logic	`supabase.from(...).insert`	`databases.createDocument()`

Database model	Table: `algvs`	Database + Collection: `algvs`

☒ Required Setup in Appwrite

Before using the new code, ensure your Appwrite backend has:

1. **Project created**

2. **Database named `algvs_db`**

3. **Collection named `algvs`**

4. The following fields in the `algvs` collection:

   - `date` (type: string / ISO date)
   - `opponentname` (string)
   - `opponentflag` (string - URL)
   - `opponentscore` (integer)
   - `algeriascore` (integer)
   - `coachname` (string)
   - `event` (string)

Make sure the **API Key or Session** has permission to read/write documents in this collection.

Final Project Structure

```
.
☒☒☒ index.html
☒☒☒ admin.html
☒☒☒ main.js
```

`index.html`

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
  <title>Alg Vs 25</title>
  <script src="https://cdn.jsdelivr.net/npm/@tailwindcss/browser@4"></script>
  <script type="module" src="./main.js"></script>
</head>
<body>
  <h1 class="font-bold text-3xl text-center">
    The matches of the Algerian national team under the leadership of coach
    <mark class="bg-sky-300 text-nowrap px-2 rounded-sm">Vladimir Petkovic</mark>
  </h1>

  <table id="StatisticsTable" class="container mx-auto my-6 text-center text-xl">
    <thead>
      <tr>
        <th class="bg-emerald-400">Win</th>
        <th class="bg-rose-400">Lose</th>
        <th class="bg-amber-400">Draw</th>
        <th class="bg-sky-400">Scored</th>
        <th class="bg-violet-400">Received</th>
      </tr>
    </thead>
    <tbody></tbody>
  </table>

  <table id="DataTable" class="container mx-auto mb-4">
    <thead>
      <tr class="border-y border-black">
        <th>Team 1</th>
        <th></th>
        <th>Score</th>
        <th></th>
        <th>Opponent</th>
      </tr>
    </thead>
    <tbody></tbody>
  </table>
</body>
</html>
```

---

admin.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8"/>
  <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
```

```html
  <title>Alg Vs 25</title>
  <script src="https://cdn.jsdelivr.net/npm/@tailwindcss/browser@4"></script>
  <script type="module" src="./main.js"></script>
</head>
<body>
  <header class="w-full bg-slate-200 py-4 gap-4 flex items-center justify-center">
    <a class="text-sky-500 underline font-bold" href="/">Home Page</a>
    <a class="text-sky-500 underline font-bold" href="/admin.html">Admin Page</a>
  </header>

  <fieldset class="w-96 border-2 border-black rounded-sm mx-auto mt-4">
    <legend class="ml-4 bg-black text-white rounded-sm px-2 py-0.5">New Game</legend>

    <form autocomplete="off" class="flex flex-col gap-2 p-2">
      <div class="grid">
        <label for="date">Date</label>
        <input type="date" name="date" id="date" class="border border-black rounded-sm px-2 py-0.5"/>
      </div>

      <div class="grid">
        <label for="game type">Game Type</label>
        <select name="game type" id="game type" class="border border-black rounded-sm px-2 py-0.5">
          <option hidden selected>-- Game Type --</option>
          <option value="International Friendly">International Friendly</option>
          <option value="Africa Cup of Nations">Africa Cup of Nations</option>
          <option value="Africa Cup of Nations Qualifying">Africa Cup of Nations Qualifying</option>
          <option value="FIFA World Cup Qualifying - CAF">FIFA World Cup Qualifying - CAF</option>
          <option value="FIFA World Cup">FIFA World Cup</option>
        </select>
      </div>

      <div class="grid">
        <label for="coach">Coach</label>
        <input type="text" name="coach" id="coach" value="Vladimir Petkovic" class="border border-black rounded-sm px-2 py-0.5"/>
      </div>

      <div class="grid gap-2">
        <span>Score</span>
        <div class="flex gap-2">
          <input type="text" name="Opponent_01" id="Opponent_01" value="ALGERIA" class="border border-black rounded-sm px-2 py-0.5 w-[70%]"/>
          <input type="number" name="Score_01" id="Score_01" placeholder="Alg Score" class="border border-black rounded-sm px-2 py-0.5  w-[30%]"/>
        </div>
        <div class="flex gap-2">
          <input type="text" name="Opponent_02" id="Opponent_02" placeholder="Opponent" class="border border-black rounded-sm px-2 py-0.5  w-[70%]"/>
          <input type="number" name="Score_02" id="Score_02" placeholder="Op Score" class="border border-black rounded-sm px-2 py-0.5  w-[30%]"/>
        </div>
      </div>
      <div class="grid">
        <label for="Opponent Flag">Opponent Flag</label>
        <input type="url" name="Opponent Flag" id="Opponent Flag" placeholder="Opponent Flag" class="border border-black rounded-sm px-2 py-0.5"/>
      </div>

      <button type="submit" class="bg-emerald-300 border border-black rounded-sm">Insert</button>
    </form>
  </fieldset>
</body>
</html>
```

---

main.js

```
import { Client, Databases, ID } from "https://cdn.jsdelivr.net/npm/appwrite@13.0.0/+esm";

// Appwrite setup
const client = new Client()
  .setEndpoint("https://cloud.appwrite.io/v1") // or your Appwrite endpoint
  .setProject("YOUR_PROJECT_ID");

const databases = new Databases(client);
const DATABASE_ID = "algvs_db";
const COLLECTION_ID = "algvs";

// Utilities
function Timestamp(dateString) {
  const date = new Date(dateString);
  return date.toLocaleDateString("en-US", { year: "numeric", month: "short", day: "numeric" });
}

function capitalizeWords(str) {
  return str?.toLowerCase()?.replace(/\b\w/g, l => l.toUpperCase()) || "";
}

function transformObjectKeys(obj) {
  const newObj = {};
  for (const [key, val] of Object.entries(obj)) {
    newObj[key.toLowerCase()] = val;
  }
  return newObj;
}

const CasesBg = { win: "emerald", loss: "rose", draw: "amber" };
const AlgFlag = "https://gxdpkrwfrlqnxjiqxrnh.supabase.co/storage/v1/object/sign/algvs/countries-flags/image_62.png";

if (location.pathname === "/" || location.pathname.includes("index.html")) {
  document.addEventListener("DOMContentLoaded", async () => {
    try {
      const DataTable = document.querySelector("#DataTable tbody");
      const StatisticsTable = document.querySelector("#StatisticsTable tbody");

      const res = await databases.listDocuments(DATABASE_ID, COLLECTION_ID);
      const data = res.documents.filter(d => d.coachname === "Vladimir Petkovic").reverse();

      const TR = ({ opponentname, opponentflag, opponentscore, algeriascore, date, event }) => {
        const Case = algeriascore === opponentscore ? "draw" : algeriascore > opponentscore ? "win" : "loss";
        return `
          <tr class="bg-${CasesBg[Case]}-400">
            <td colspan="5" class="text-center"><span class="underline">${Timestamp(date)}</span> | ${event}</td>
          </tr>
          <tr class="bg-${CasesBg[Case]}-400 py-2 border-b-2 border-black text-xl text-center font-bold">
            <td>Algeria</td>
            <td><img src="${AlgFlag}" class="block object-cover size-8 ml-auto"/></td>
            <td>${algeriascore} - ${opponentscore}</td>
            <td><img src="${opponentflag}" class="block object-cover size-8 mr-auto"/></td>
            <td class="text-left">${opponentname}</td>
          </tr>`;
      };

      data.forEach(item => {
        DataTable.innerHTML += TR(item);
      });

      const win = data.filter(i => i.algeriascore > i.opponentscore).length;
      const loss = data.filter(i => i.algeriascore < i.opponentscore).length;
      const draw = data.filter(i => i.algeriascore === i.opponentscore).length;
      const scored = data.reduce((acc, cur) => acc + cur.algeriascore, 0);
      const received = data.reduce((acc, cur) => acc + cur.opponentscore, 0);

      StatisticsTable.innerHTML = `<tr>
        <td>${win}</td><td>${loss}</td><td>${draw}</td>
        <td>${scored}</td><td>${received}</td>
      </tr>`;
    } catch (err) {
      console.error("Appwrite fetch error:", err);
    }
  });

} else if (location.pathname.includes("admin.html")) {
  document.addEventListener("DOMContentLoaded", () => {
    const form = document.querySelector("form");

    form.addEventListener("submit", async (e) => {
      e.preventDefault();

      const data = Object.fromEntries(new FormData(form));
      const payload = transformObjectKeys({
        date: data.date,
        opponentname: capitalizeWords(data.Opponent_02),
        opponentflag: data["Opponent Flag"],
        opponentscore: Number(data.Score_02),
        algeriascore: Number(data.Score_01),
        coachname: data.coach,
        event: data["game type"]
      });

      try {
        const res = await databases.createDocument(DATABASE_ID, COLLECTION_ID, ID.unique(), payload);
        alert("Match added successfully!");
        form.reset();
      } catch (err) {
        console.error("Appwrite insert error:", err);
        alert("Error saving match.");
      }
    });
  });
}
```

Final Steps

- Replace `YOUR_PROJECT_ID` with your actual Appwrite Project ID.

- If you're self-hosting Appwrite, update the endpoint accordingly.

- Make sure your Appwrite project allows web access (set up the platform in the dashboard).

- Optional: Secure it with proper Appwrite Auth instead of public API Key (not shown here for simplicity).

Let me know if you’d like help generating initial test data, securing with Appwrite Auth, or deploying this to a static host.

You asked:

I'm facing these errors when starting.

```
Access to XMLHttpRequest at 'https://cloud.appwrite.io/v1/databases/algvs_db/collections/algvs/documents' from origin 'http://127.0.0.1:5500' has been blocked by CORS policy: The 'Access-Control-Allow-Origin' header has a value 'http://localhost:5500' that is not equal to the supplied origin.

Appwrite fetch error: AppwriteException: Network request failed
    at s.<anonymous> (https://cdn.jsdelivr.net/npm/appwrite@13.0.0/+esm:7:6885)
    at Generator.throw (<anonymous>)
    at c (https://cdn.jsdelivr.net/npm/appwrite@13.0.0/+esm:7:233)
```

Also, please change the UI; I don't like the current one.

The AI replied:

You're running into **CORS (Cross-Origin Resource Sharing)** issues, which are common when using APIs like Appwrite in frontend-only environments.

Problem 1: CORS error

Cause

The error:

```
Access to XMLHttpRequest at 'https://cloud.appwrite.io/...' from origin 'http://127.0.0.1:5500' has been blocked by CORS...
```

Means that Appwrite has only allowed `http://localhost:5500`, but you're accessing it via `http://127.0.0.1:5500`.

Fix

Go to your **Appwrite Console** ☒ **Project** ☒ **Platforms** and do this:

1. **Add both URLs**:

   - `http://localhost:5500`

   - `http://127.0.0.1:5500`

2. Make sure they're allowed in **CORS origin settings** for your platform.

3. Save and deploy changes.

Tip: Always match the exact protocol + domain + port (no trailing slashes).

Problem 2: `AppwriteException: Network request failed`

Cause

This usually happens if:

- You're using a Project ID that doesn't exist.

- CORS prevented the request from even being made.

- You’ re not connected to the internet.

- Your collection or document permissions are misconfigured.

Fix Checklist

- Double-check your `Project ID`.

- Make sure the collection `algvs` is inside the database `algvs_db`.

- Confirm your collection’ s **Read and Write Permissions** allow access (either anonymously or via API Key/ session).

- Wait 1–2 minutes after changing CORS settings in Appwrite Cloud—it can take time to apply.

UI Overhaul (Tailwind + Modern Layout)

I’ ll now create a **completely refreshed modern UI**, still keeping it functional with the Appwrite backend.

Features of the new design:

- Centered layout with cards

- Modern typography & Tailwind design

- Separate admin panel with a sidebar

- Responsive

- No clutter

Let’ s proceed...

Would you prefer a:

1. **Clean dashboard** style (sidebar + content panel)?

2. **Centered app-style** with a table or list in the middle of the page?

Please choose **1** or **2**, or describe what you're visualizing (like "dark mode", "scoreboard style", "flag icons prominent", etc.). I'll then redesign everything accordingly — with the **updated HTML/CSS/JS code in one go**, as requested.

You asked:

This is a sample of the previous database that I want to upload to Appwrite. I exported it from Supabase as a CSV file. How can I upload it to the new database?

```
CSV
id,opponentname,opponentflag,opponentscore,algeriascore,date,coachname,event

1,Tanzania,https://gxdpkrwfrlqnxjiqxrnh.supabase.co/storage/v1/object/public/algvs/countries-flags/image_135.png,1,1,2011-09-03 00:00:00,Vahid Halilhodzic,Africa Cup of Nations Qualifying

2,Central African Republic,https://gxdpkrwfrlqnxjiqxrnh.supabase.co/storage/v1/object/public/algvs/countries-flags/image_174.png,0,2,2011-10-09 00:00:00,Vahid Halilhodzic,Africa Cup of Nations Qualifying

3,Tunisia,https://gxdpkrwfrlqnxjiqxrnh.supabase.co/storage/v1/object/public/algvs/countries-flags/image_103.png,0,1,2011-11-12 00:00:00,Vahid Halilhodzic,International Friendly

4,Gambia,https://gxdpkrwfrlqnxjiqxrnh.supabase.co/storage/v1/object/public/algvs/countries-flags/image_198.png,1,2,2012-02-29 00:00:00,Vahid Halilhodzic,Africa Cup of Nations Qualifying

5,Niger,https://gxdpkrwfrlqnxjiqxrnh.supabase.co/storage/v1/object/public/algvs/countries-flags/image_141.png,0,3,2012-05-26 00:00:00,Vahid Halilhodzic,International Friendly

6,Rwanda,https://gxdpkrwfrlqnxjiqxrnh.supabase.co/storage/v1/object/public/algvs/countries-flags/image_138.png,0,4,2012-06-02 00:00:00,Vahid Halilhodzic,FIFA World Cup Qualifying - CAF

7,Mali,https://gxdpkrwfrlqnxjiqxrnh.supabase.co/storage/v1/object/public/algvs/countries-flags/image_159.png,2,1,2012-06-06 00:00:00,Vahid Halilhodzic,FIFA World Cup Qualifying - CAF

8,Gambia,https://gxdpkrwfrlqnxjiqxrnh.supabase.co/storage/v1/object/public/algvs/countries-flags/image_198.png,1,4,2012-06-15 00:00:00,Vahid Halilhodzic,Africa Cup of Nations Qualifying
```

The AI replied:

Great — thanks for sharing the CSV sample!

You want to:

Upload a CSV file (exported from Supabase) to your new **Appwrite database**.

I'll walk you through a clean and correct way to do this. Appwrite **doesn't support CSV import out of the box**, so you'll use a **custom script** (using JavaScript + Appwrite SDK) to import your CSV data programmatically.

Overview of the Process

1. **Convert the CSV to JSON** in your JavaScript code.

2. **Use the Appwrite SDK** to insert each row as a new document in your collection.

Step-by-Step Setup

1. Prepare your Appwrite details

You'll need:

- **Project ID**

- **Database ID** (e.g., `algvs_db`)

- **Collection ID** (e.g., `algvs`)

- **API Key** with permission to create documents

Make sure the API Key has `database.write` and `documents.write` permission.

---

2. Create the script

Here’s a complete script you can run in your browser (or Node.js with slight changes):

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <title>CSV to Appwrite Importer</title>
</head>
<body>
  <input type="file" id="csvFile" accept=".csv" />
  <button id="uploadBtn">Upload to Appwrite</button>

  <script type="module">
    import { Client, Databases } from "https://cdn.jsdelivr.net/npm/appwrite@13.0.0/+esm";

    const client = new Client()
      .setEndpoint("https://cloud.appwrite.io/v1")
      .setProject("YOUR_PROJECT_ID") // Replace
      .setKey("YOUR_API_KEY");      // Replace

    const databases = new Databases(client);
    const databaseId = "YOUR_DATABASE_ID";     // Replace
    const collectionId = "YOUR_COLLECTION_ID"; // Replace

    // Utility to parse CSV string to JSON
    function parseCSV(csvText) {
      const lines = csvText.trim().split("\n");
      const headers = lines[0].split(",").map(h => h.trim().toLowerCase());

      return lines.slice(1).map(line => {
        const values = line.split(",").map(v => v.trim());
        const obj = {};
        headers.forEach((h, i) => {
          obj[h] = values[i];
        });

        return {
          opponentname: obj.opponentname,
          opponentflag: obj.opponentflag,
          opponentscore: Number(obj.opponentscore),
          algeriascore: Number(obj.algeriascore),
          date: new Date(obj.date).toISOString(), // Appwrite expects ISO 8601
          coachname: obj.coachname,
          event: obj.event
        };
      });
    }

    document.getElementById("uploadBtn").addEventListener("click", async () => {
      const fileInput = document.getElementById("csvFile");
      if (!fileInput.files.length) return alert("Please select a CSV file.");

      const file = fileInput.files[0];
      const text = await file.text();
      const records = parseCSV(text);

      for (const record of records) {
        try {
          await databases.createDocument(databaseId, collectionId, 'unique()', record);
          console.log(`Uploaded: ${record.opponentname} (${record.date})`);
        } catch (err) {
          console.error("Failed to insert document:", err);
        }
      }

      alert("Upload complete!");
    });
  </script>
</body>
</html>
```

Notes:

- If you want to run this **locally**, just save it as `upload.html` and open it in your browser.
- Make sure your Appwrite database allows **document creation via API key**.
- You can remove the `id` field from the CSV — Appwrite will auto-generate unique IDs unless you want to explicitly include them.

What’s next?

Let me know **how you want the new UI** to look (from earlier), and I’ll:

- Finalize a fully working `index.html`, `admin.html`, and `main.js`
- Include the improved layout with modern UX
- Keep it Appwrite-ready (i.e., no Supabase references)

Would you like:

- A **dark scoreboard** UI?
- A **clean admin dashboard** with inputs on a sidebar?
- Something **mobile-first**?

Let me know and I’ll build it fully.

You asked:

```
Uncaught TypeError: (intermediate value).setEndpoint(...).setProject(...).setKey is not a function
    at upload.html:19:8
```

The AI replied:

Thanks — that error tells us you're trying to use `.setKey()` on the Appwrite `Client` object in the browser, but **`.setKey()` is only available in server-side (Node.js) environments**, not in frontend JavaScript running in the browser.

Why this happens:

Appwrite restricts API Keys from being used in the browser **for security reasons**. Exposing an API key in client-side code would allow **anyone** to use it and manipulate your database.

What to do instead:

You have **2 secure options**:

Option 1: Use Appwrite's session-based authentication

You create a **dummy login page for admin** and log in with an email/password account. Then you can make secure requests without needing `.setKey()`.

Steps:

1. In Appwrite console, go to **Users** ☒ **Invite yourself** or **create a user**.

2. Allow email/password auth in **Auth settings**.

3. In your admin dashboard script, use:

```
import { Account } from 'https://cdn.jsdelivr.net/npm/appwrite@13.0.0/+esm';
```

```
const account = new Account(client); await account.createEmailSession("youremail@example.com", "yourpassword");
```

Then you can use the `Databases` client normally — no need for `.setKey()`.

Option 2: Run the CSV upload script server-side (Node.js)

If you prefer not to handle authentication in the browser, here’s a **Node.js script** that uses `.setKey()` securely.

Node.js script

Save this as `upload.js`:

```
import fs from 'fs';
import csv from 'csv-parser';
import { Client, Databases } from 'appwrite';

const client = new Client()
  .setEndpoint('https://cloud.appwrite.io/v1')
  .setProject('YOUR_PROJECT_ID')
  .setKey('YOUR_API_KEY'); //  Safe to use here (not in browser)

const databases = new Databases(client);

const databaseId = 'YOUR_DATABASE_ID';
const collectionId = 'YOUR_COLLECTION_ID';

const results = [];

fs.createReadStream('your-csv-file.csv')
  .pipe(csv())
  .on('data', (data) => results.push(data))
  .on('end', async () => {
    for (const record of results) {
      try {
        const payload = {
          opponentname: record.opponentname,
          opponentflag: record.opponentflag,
          opponentscore: Number(record.opponentscore),
          algeriascore: Number(record.algeriascore),
          date: new Date(record.date).toISOString(),
          coachname: record.coachname,
          event: record.event
        };

        await databases.createDocument(databaseId, collectionId, 'unique()', payload);
        console.log(`Uploaded: ${payload.opponentname} (${payload.date})`);
      } catch (err) {
        console.error("Failed to insert:", err.message);
      }
    }
  });
```

How to run:

1. Save CSV as `your-csv-file.csv`.

2. Run:

```
npm install appwrite csv-parser
node upload.js
```

Recommendation

Use **Node.js for data imports** (more secure, easier to control). Use **frontend code with session-based auth** for admin UI.

Let me know which path you prefer and I’ll finalize the exact script or UI accordingly.

You asked:

---

will it work like this?

```
HTML
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8" />
  <title>CSV to Appwrite Importer</title>
</head>

<body>
  <input type="file" id="csvFile" accept=".csv" />
  <button id="uploadBtn">Upload to Appwrite</button>

  <script type="module">
    import { Client, Databases } from "https://cdn.jsdelivr.net/npm/appwrite@18.1.1/+esm";

    const client = new Client()
      .setEndpoint("https://cloud.appwrite.io/v1")
      .setProject("YOUR_PROJECT_ID") // Replace
      // .setKey("");      // Replace

    const databases = new Databases(client);

    // const DATABASE_ID = "algvs_db";
    // const COLLECTION_ID = "algvs";

    const databaseId = "algvs_db";     // Replace
    const collectionId = "algvs"; // Replace

    // Utility to parse CSV string to JSON
    function parseCSV(csvText) {
      const lines = csvText.trim().split("\n");
      const headers = lines[0].split(",").map(h => h.trim().toLowerCase());

      return lines.slice(1).map(line => {
        const values = line.split(",").map(v => v.trim());
        const obj = {};
        headers.forEach((h, i) => {
          obj[h] = values[i];
        });

        return {
          opponentname: obj.opponentname,
          opponentflag: obj.opponentflag,
          opponentscore: Number(obj.opponentscore),
          algeriascore: Number(obj.algeriascore),
          date: new Date(obj.date).toISOString(), // Appwrite expects ISO 8601
          coachname: obj.coachname,
          event: obj.event
        };
      });
    }
    document.getElementById("uploadBtn").addEventListener("click", async () => {
      const fileInput = document.getElementById("csvFile");
      if (!fileInput.files.length) return alert("Please select a CSV file.");

      const file = fileInput.files[0];
      const text = await file.text();
      const records = parseCSV(text);

      for (const record of records) {
        try {
          await databases.createDocument(databaseId, collectionId, 'unique()', record);
          console.log(`Uploaded: ${record.opponentname} (${record.date})`);
        } catch (err) {
          console.error("Failed to insert document:", err);
        }
      }

      alert("Upload complete!");
    });
  </script>
</body>

</html>
```

The AI replied:

You're very close! But unfortunately, **this exact code will not work in the browser** because you're trying to use `Appwrite`'s Admin privileges (with `createDocument()` and no authentication) **without authentication or an API key**.

Here's why it won't work:

- The browser **cannot use `.setKey()`** (for security).
- You're trying to call `createDocument()` without being **authenticated as a user**, which Appwrite will reject.
- Appwrite’s **security rules** block unauthenticated frontend users from writing to the database.

You Have Two Working Options:

Option 1: Use Appwrite with Email/Password Auth in the Browser

This approach allows the script to run in the browser, securely, using **Appwrite's user-based authentication**.

Steps:

1. **Enable Email/Password auth** in Appwrite:

   - Go to your Appwrite console ⊠ `Auth` ⊠ enable **Email/Password**.

2. **Create a user** (admin account for this importer):

   - Console ⊠ `Users` ⊠ create new user (e.g. `admin@algvs.com` + password).

3. Update your script like this:

```
import { Client, Databases, Account } from "https://cdn.jsdelivr.net/npm/appwrite@18.1.1/+esm";

const client = new Client() .setEndpoint("https://cloud.appwrite.io/v1") .setProject("YOUR_PROJECT_ID");

const databases = new Databases(client); const account = new Account(client);

// Log in as a user await account.createEmailSession("admin@algvs.com", "your_password");
```

Then your `createDocument()` call will work, because you're authenticated as a user.

Note: You must set the correct **database rules** to allow authenticated users to create documents.

Option 2: Move This Script to Node.js (Recommended for Uploading CSV)

If you're uploading large CSV data and don’t need this in the frontend, just copy your logic into a Node.js script where you can safely use `.setKey()`.

This is **secure**, **faster**, and doesn't expose any credentials in the browser.

Example CLI script (Node.js):

```
npm install appwrite csv-parser


// upload.js (Node)
import fs from 'fs';
import csv from 'csv-parser';
import { Client, Databases } from 'appwrite';

const client = new Client()
  .setEndpoint('https://cloud.appwrite.io/v1')
  .setProject('YOUR_PROJECT_ID')
  .setKey('YOUR_API_KEY'); //  Allowed here

const databases = new Databases(client);
const databaseId = 'algvs_db';
const collectionId = 'algvs';

const results = [];

fs.createReadStream('algvs.csv')
  .pipe(csv())
  .on('data', (data) => results.push(data))
  .on('end', async () => {
    for (const record of results) {
      try {
        await databases.createDocument(databaseId, collectionId, 'unique()', {
          opponentname: record.opponentname,
          opponentflag: record.opponentflag,
          opponentscore: Number(record.opponentscore),
          algeriascore: Number(record.algeriascore),
          date: new Date(record.date).toISOString(),
          coachname: record.coachname,
          event: record.event
        });
        console.log(` Uploaded: ${record.opponentname}`);
      } catch (err) {
        console.error(` Failed: ${err.message}`);
      }
    }
  });
```

Then run with:

```
node upload.js
```

Summary

Option	Frontend (HTML/JS)	Backend (Node.js)	Uses API Key	Secure
1. Auth via Email	☒ Yes	☒ No	☒ No	☒
2. Node script with API key	☒ No	☒ Yes	☒ Yes	☒☒☒

Let me know which way you prefer (frontend with login or backend upload), and I’ ll provide a clean version of the full code for it.

You asked:

.setKey('YOUR_API_KEY'); // Allowed here

Where can I get this?

The AI replied:

To get your **Appwrite API Key** (`setKey(...)`), follow these steps:

How to Get Your Appwrite API Key

1. **Log into your Appwrite Console**:
   Go to https://cloud.appwrite.io/console

2. **Select your project** (e.g., the one with `algvs_db`)

3. In the left-hand menu, go to:
   API Keys ☒ click **Create API Key**

4. **Fill in the details:**

   - **Name**: e.g. `CSV Import Script`

   - **Scopes**: Make sure to include at least:

     - `databases.read`

     - `databases.write`

     - `documents.read`

     - `documents.write`

   - **Expiration**: (optional) Set a time if you want it to auto-expire.

5. Click **Create** – Appwrite will generate a key **once only**. Copy and save it securely.

How to Use It in Your Script

In your **Node.js script**, you'll do:

```
const client = new Client()
  .setEndpoint('https://cloud.appwrite.io/v1')
  .setProject('YOUR_PROJECT_ID')
  .setKey('YOUR_GENERATED_API_KEY'); // Paste it here
```

Important: Never expose this key in frontend code (HTML/JS in the browser) — it has write privileges.

Would you like me to give you the full working Node.js CSV importer script next?
