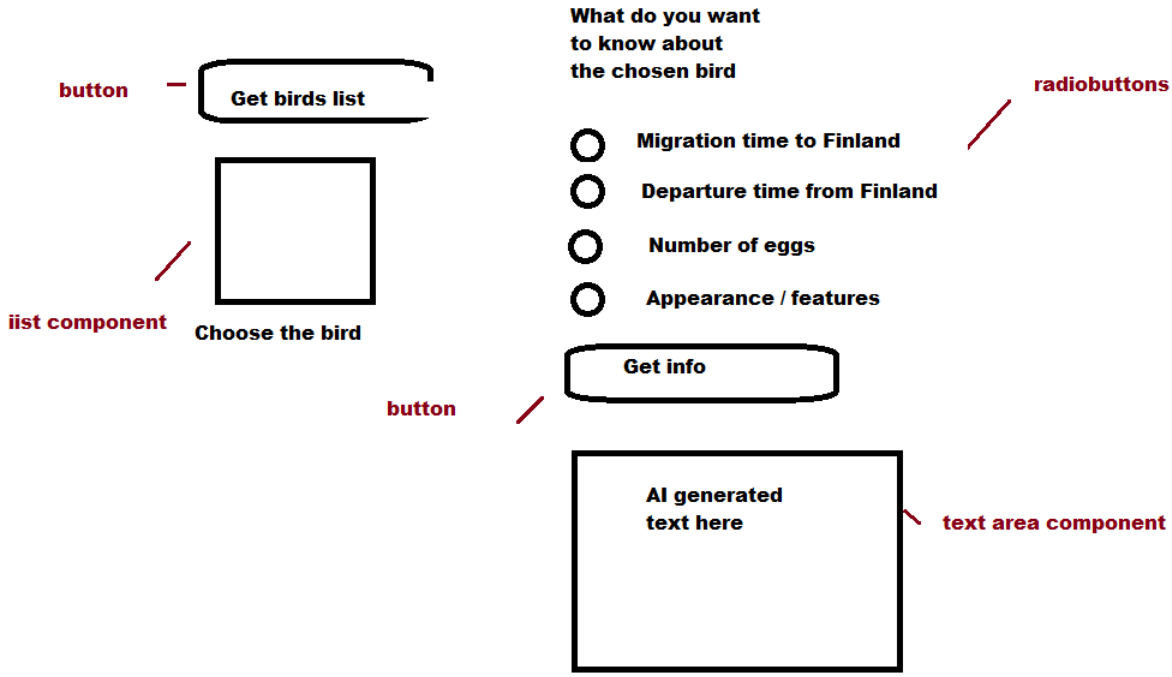
This gui proto was created first

The versatile GUI is the main idea is here: instead of textual promtp, you can use components to give information for the query. The UI is easy to use and standardized...

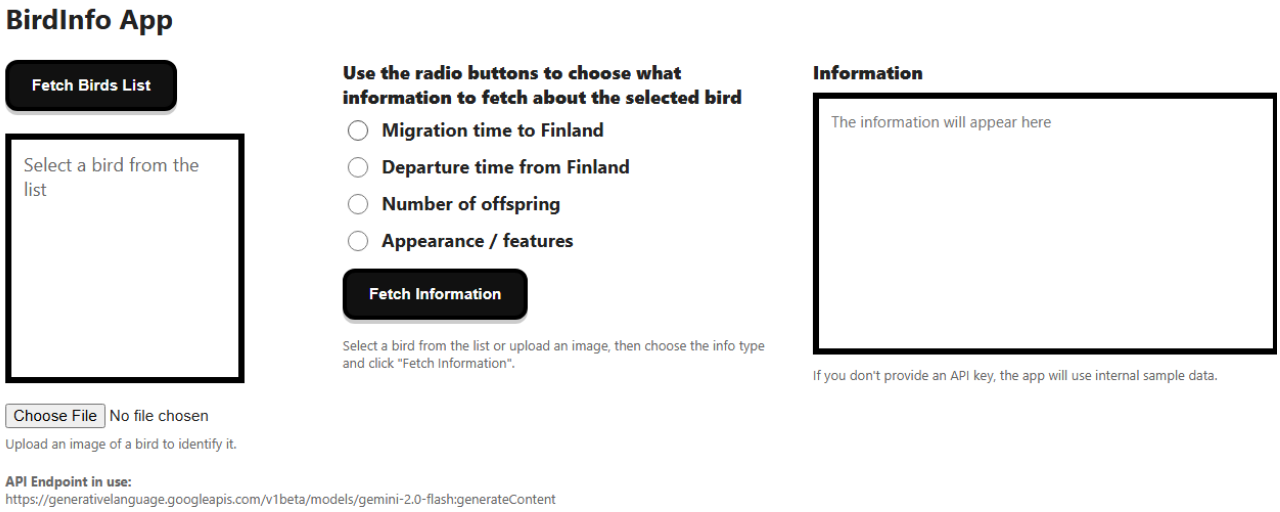


Then AI was asked to create an app that creates also the GUi...

Then AI was asked to create an app that creates also the GUi, then the app itself.

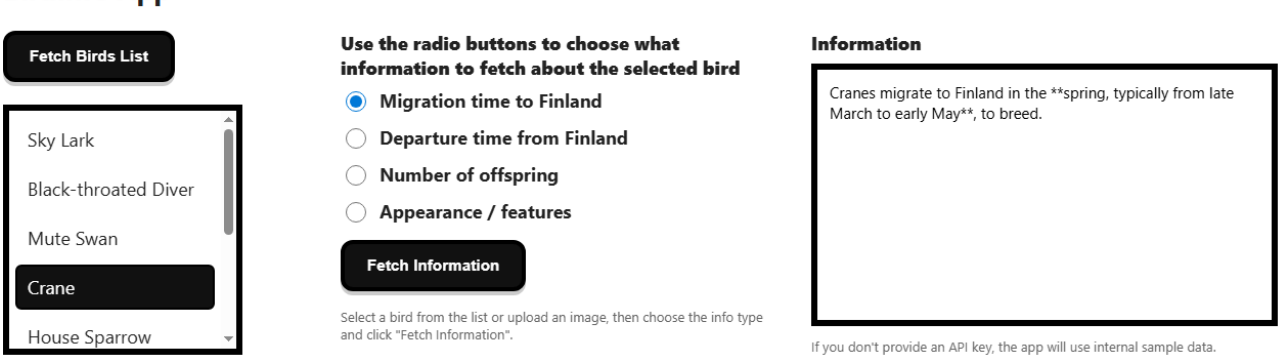
App was to be web app and it gives information about birds.

Here is the app on browser



In this version user can also upload an image.

Usage



Note: now API key is given, but in public usage you have to keep API key secret: there are some ways to do that.

Here is the code where AI contact is used:

```
# --- Configure Gemini API --- GEMINI_API_KEY = os.getenv("GEMINI_API_KEY") if not GEMINI_API_KEY: raise ValueError("GEMINI_API_KEY environment variable not set. Please create a .env file or set it.")
```

And here is the whole code (API key is not shown)

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <title>BirdInfo App</title>
  <meta name="viewport" content="width=device-width,initial-scale=1" />
```

```
  <style>
    :root{
      --accent:#222;
      --muted:#666;
      --btn-bg:#111;
      --btn-text:#fff;
      font-family: "Segoe UI", Arial, sans-serif;
    }
    body{ margin:20px; color:var(--accent); }
    .container{ display:flex; gap:40px; align-items:flex-start; flex-wrap:wrap; }
    .left, .right { min-width:260px; }
    .btn {
      display:inline-block; padding:12px 22px; background:var(--btn-bg); color:var(--btn-text);
      border-radius:12px; border:4px solid #000; cursor:pointer; font-weight:700;
      box-shadow: 0 4px 0 rgba(0,0,0,0.2); margin-bottom:10px;
    }
    .list-box{ width:210px; height:220px; border:6px solid #000; display:flex; flex-direction:column;
      overflow:auto; padding:6px; gap:4px; background: #fff; }
    .list-item{ padding:8px 10px; border-radius:6px; border:2px solid transparent; cursor:pointer;
user-select:none; }
    .list-item.selected{ background:#111; color:#fff; border-color:#000; }
    .center{ max-width:420px; flex:1; }
    .radios{ display:flex; flex-direction:column; gap:12px; margin:8px 0 16px 0; }
    .radios label { display:flex; align-items:center; gap:10px; font-weight:700; }
    .radios input[type="radio"]{ width:20px; height:20px; }
    .textarea-box{ width:420px; height:220px; border:6px solid #000; padding:12px; resize:none;
font-size:15px; font-family:inherit; }
    .hint { color:var(--muted); font-size:13px; margin-top:6px; }
    footer { margin-top:20px; color:var(--muted); font-size:13px; }
    .image-preview{ max-width:260px; max-height:200px; border:4px solid #000; margin-
bottom:10px; }
  </style>
</head>
<body>
  <h2>BirdInfo App</h2>
  <div class="container">

    <!-- LEFT PANEL -->
    <div class="left">
      <button id="fetchListBtn" class="btn">Fetch Birds List</button>
      <div style="height:12px"></div>

      <div class="list-box" id="birdList" aria-label="Select a bird from the list">
        <div style="color:#666; padding:6px;">Select a bird from the list</div>
      </div>

      <div style="height:18px"></div>

      <div>
        <input type="file" id="birdImageUpload" accept="image/*">
        <div class="hint">Upload an image of a bird to identify it.</div>
        <img id="imagePreview" class="image-preview" src="" alt="" style="display:none;">
```

```
    </div>
   </div>

   <!-- CENTER PANEL -->
   <div class="center">
    <div style="font-weight:800; margin-bottom:8px">
     Use the radio buttons to choose what information to fetch about the selected bird
    </div>

    <div class="radios" id="options">
     <label><input type="radio" name="infoType" value="migration_to_finland"> Migration time to Finland</label>
     <label><input type="radio" name="infoType" value="departure_from_finland"> Departure time from Finland</label>
     <label><input type="radio" name="infoType" value="offspring_count"> Number of offspring</label>
     <label><input type="radio" name="infoType" value="appearance"> Appearance / features</label>
    </div>

    <button id="getInfoBtn" class="btn">Fetch Information</button>
    <div class="hint">
     Select a bird from the list or upload an image, then choose the info type and click "Fetch Information".
    </div>
   </div>

   <!-- RIGHT PANEL -->
   <div class="right">
    <div style="font-weight:800; margin-bottom:8px">Information</div>
    <textarea id="result" class="textarea-box" placeholder="The information will appear here" readonly></textarea>
    <div class="hint">
     If you don't provide an API key, the app will use internal sample data.
    </div>
   </div>
  </div>

  <footer>
   <strong>API Endpoint in use:</strong><br>
   https://generativelanguage.googleapis.com/v1beta/models/gemini-2.0-flash:generateContent
  </footer>

<script>
/* ---------------- SETTINGS ---------------- */
const API_KEY = "xxxxxxxxxxxxxxxxxxxx";

/* ---------------- DATA ---------------- */
const finnishMigratoryBirds = [
  "Sky Lark", "Black-throated Diver", "Mute Swan", "Crane", "House Sparrow",
  "Wood Warbler", "Northern Lapwing", "Barn Swallow", "Reed Warbler"
];
```

```
const fallbackInfo = {
  'migration_to_finland': (bird)=> `${bird}: Typically arrives between April and May.`,
  'departure_from_finland': (bird)=> `${bird}: Typically departs between August and September.`,
  'offspring_count': (bird)=> `${bird}: Usually 2–5 offspring per nesting.`,
  'appearance': (bird)=> `${bird}: Typical features and coloring vary by species.`
};

/* ---------------- DOM ---------------- */
const birdListEl = document.getElementById("birdList");
const fetchListBtn = document.getElementById("fetchListBtn");
const getInfoBtn = document.getElementById("getInfoBtn");
const resultEl = document.getElementById("result");
const birdImageUpload = document.getElementById("birdImageUpload");
const imagePreview = document.getElementById("imagePreview");

let selectedBird = null;

/* ---------------- FUNCTIONS ---------------- */
function renderBirdList(list){
  birdListEl.innerHTML = "";
  list.forEach((name) => {
    const div = document.createElement("div");
    div.className = "list-item";
    div.textContent = name;
    div.addEventListener("click", () => {
      const prev = birdListEl.querySelector(".selected");
      if(prev) prev.classList.remove("selected");
      div.classList.add("selected");
      selectedBird = name;
      imagePreview.style.display = "none";
    });
    birdListEl.appendChild(div);
  });
}

function getSelectedInfoType(){
  const radios = document.getElementsByName("infoType");
  for(const r of radios) if(r.checked) return r.value;
  return null;
}

function buildPrompt(bird, infoType){
  const infoText = {
    'migration_to_finland': 'Migration time to Finland',
    'departure_from_finland': 'Departure time from Finland',
    'offspring_count': 'Number of offspring per nest',
    'appearance': 'Appearance and features'
  }[infoType];
  return `Write a concise English description of the bird "${bird}" about: ${infoText}.`;
}
```

```
async function callGemini(prompt){
  const url =
    `https://generativelanguage.googleapis.com/v1beta/models/gemini-2.0-flash:generateContent?
key=${API_KEY}`;
  const body = { contents:[{ parts:[{ text: prompt }] }] };
  const resp = await fetch(url, { method:"POST", headers:{ "Content-Type":"application/json" },
body: JSON.stringify(body) });
  if(!resp.ok){ const text = await resp.text(); throw new Error("API error: "+resp.status+" — "+text);
}
  const json = await resp.json();
  try { return json.candidates[0].content.parts[0].text; }
  catch(e){ return JSON.stringify(json,null,2); }
}

/* ---------------- IMAGE UPLOAD ---------------- */
birdImageUpload.addEventListener("change", (e)=>{
  const file = e.target.files[0];
  if(!file) return;
  const reader = new FileReader();
  reader.onload = () => {
    imagePreview.src = reader.result;
    imagePreview.style.display = "block";
    // Placeholder bird recognition
    selectedBird = "Unknown Bird";
    resultEl.value = "Bird image uploaded. Information will be fetched once you select info type and
click 'Fetch Information'.";
  };
  reader.readAsDataURL(file);
});

/* ---------------- BUTTON HANDLERS ---------------- */
fetchListBtn.addEventListener("click", () => {
  renderBirdList(finnishMigratoryBirds);
  resultEl.value = "Birds fetched.";
});

getInfoBtn.addEventListener("click", async () => {
  const infoType = getSelectedInfoType();
  if(!selectedBird){
    alert("Please select a bird from the list or upload an image first!");
    return;
  }
  if(!infoType){
    alert("Please select an information type (radio).");
    return;
  }

  resultEl.value = "Fetching information...";

  if(!API_KEY || API_KEY === "YOUR_API_KEY_HERE"){
    setTimeout(() => {
      resultEl.value = fallbackInfo[infoType](selectedBird) +
```

```
        "\n\n(Note: Using local sample data — add your API_KEY for live info.)";
    }, 400);
    return;
  }

  try{
    const prompt = buildPrompt(selectedBird, infoType);
    const answer = await callGemini(prompt);
    resultEl.value = answer;
  } catch(err){
    console.error(err);
    resultEl.value = "Error fetching information:\n" + err.message +
      "\n\nFallback:\n" + fallbackInfo[infoType](selectedBird);
  }
});
</script>
</body>
</html>
```

is also an image is uploaded and used, AI has to allow multimodal mode.