LAMPIRAN-LAMPIRAN

Lampiran 1. Lampiran isi kodingan file Index.js

```
// YouTube Spam JUDOL Remover
require("dotenv").config();
const fs = require("fs");
const path = require("path");
const express = require("express");
const bodyParser = require("body-parser");
const { google } = require("googleapis");
const ExcelJS = require("exceljs");
const natural = require("natural");
const unidecode = require("unidecode");

const app = express();
const PORT = process.env.PORT || 3000;
const exportDir = path.join(__dirname, "exported");
if (!fs.existsSync(exportDir)) fs.mkdirSync(exportDir);

let lastScanComments = [];
let spamClassifier = null;

// 🔽 Load model Naive Bayes

function loadSpamClassifier() {
  return new Promise((resolve, reject) => {
    if (fs.existsSync("naive_model.json")) {
      natural.BayesClassifier.load("naive_model.json", null, (err, classifier)
=> {
        if (err) return reject(err);
        spamClassifier = classifier;
        console.log("✅ Model Naive Bayes berhasil dimuat");
        resolve();
      });
    } else {
      reject(new Error("❌ Model Naive Bayes belum tersedia. Jalankan
naivebayes.js dulu."));
    }
  });
}

// 🔽 Konfigurasi Express

app.set("view engine", "ejs");
app.set("views", path.join(__dirname, "views"));
app.use(express.static("public"));
app.use(express.urlencoded({ extended: true }));
app.use(express.json());

// 🔽 Autentikasi YouTube API

async function authorize() {
  const credentials = JSON.parse(fs.readFileSync("credentials.json"));
  const { client_secret, client_id, redirect_uris } = credentials.installed;
  const oAuth2Client = new google.auth.OAuth2(client_id, client_secret,
redirect_uris[0]);

  const TOKEN_PATH = "token.json";
```

```
  if (fs.existsSync(TOKEN_PATH)) {
    oAuth2Client.setCredentials(JSON.parse(fs.readFileSync(TOKEN_PATH)));
    return oAuth2Client;
  }
  throw new Error("❌ Token belum tersedia. Jalankan proses autentikasi terlebih
dahulu.");
}

// 🔍 Deteksi karakter fancy Unicode

function hasFancyUnicode(text) {
  for (const char of text) {
    const code = char.codePointAt(0);
    if (
      (code >= 0x1D400 && code <= 0x1D7FF) || // Mathematical Alphanumeric
Symbols
      (code >= 0xFF00 && code <= 0xFFEF)     || // Fullwidth Latin
      (code >= 0x2460 && code <= 0x24FF)     // Ⓜ Circled Latin Letters
(enclosed alphanumeric)
    ) {
      return true;
    }
  }
  return false;
}

// 🔍 Deteksi komentar dengan simbol mencurigakan

function isSymbolSpam(text) {
  const normalized = text.normalize("NFKD");
  const simplified = unidecode(normalized);

  const emojiRegex = /\p{Emoji}/gu;
  const allowedSymbols = /[\/',";:\}\]\{\[\+=_\-\)\(\*&\^%$#@!~`>²<]/g;

  const cleanedOriginal = normalized.replace(emojiRegex,
"").replace(allowedSymbols, "");
  const cleanedSimplified = simplified.replace(emojiRegex,
"").replace(allowedSymbols, "");

  return cleanedOriginal !== cleanedSimplified || hasFancyUnicode(text);
}

// 🔍 Hybrid Spam Filter

function isSpam(text) {
  const lowerText = text.toLowerCase();
  const blockedWords = JSON.parse(fs.readFileSync("blockedword.json"));

  const isSymbol = isSymbolSpam(text);
  const isBlocked = blockedWords.some(word =>
lowerText.includes(word.toLowerCase()));
  let isBayes = false;
  if (spamClassifier) {
    const prediction = spamClassifier.classify(text);
    isBayes = prediction === "spam";
  }

  return {
    isSpam: isSymbol || isBlocked || isBayes,
    isSymbolSpam: isSymbol,
    isBlockedWord: isBlocked,
    isNaiveBayesSpam: isBayes
```

```
  };
}

// 🔹 Ambil komentar dari video YouTube

async function getComments(auth, videoId) {
  const youtube = google.youtube({ version: "v3", auth });
  const comments = [];
  let nextPageToken = "";

  do {
    const res = await youtube.commentThreads.list({
      part: "snippet",
      videoId,
      maxResults: 100,
      pageToken: nextPageToken
    });

    res.data.items.forEach(item => {
      const snippet = item.snippet.topLevelComment.snippet;
      comments.push({
        videoId,
        author: snippet.authorDisplayName,
        text: snippet.textDisplay,
        commentId: item.snippet.topLevelComment.id
      });
    });

    nextPageToken = res.data.nextPageToken;
  } while (nextPageToken);

  return comments;
}

// 🔹 Sembunyikan komentar terpilih

async function hideSpamComments(auth, spamIds) {
  const youtube = google.youtube({ version: "v3", auth });
  for (const id of spamIds) {
    try {
      await youtube.comments.setModerationStatus({ id, moderationStatus:
"rejected" });
      console.log(`✅ Disembunyikan: ${id}`);
    } catch (err) {
      console.error(`❌ Gagal sembunyikan ${id}:`, err.message);
    }
  }
}

// 🔹 Ekspor komentar ke Excel

async function exportToExcel(comments) {
  const workbook = new ExcelJS.Workbook();
  const sheet = workbook.addWorksheet("Spam Comments");

  sheet.columns = [
    { header: "Video ID", key: "videoId", width: 20 },
    { header: "Author", key: "author", width: 25 },
    { header: "Comment", key: "text", width: 50 },
    { header: "Spam", key: "spam", width: 10 },
    { header: "Method", key: "method", width: 20 }
  ];
```

```
  comments.forEach(c => {
    let method = "Clean";
    if (c.isSymbolSpam) method = "Unicode Symbol";
    else if (c.isSymbolSpam && c.isNaiveBayesSpam) method = "Hybrid";
    else if (c.isBlockedWord) method = "Blocked Words";
    else if (c.isNaiveBayesSpam) method = "Naive Bayes";

    sheet.addRow({
      videoId: c.videoId,
      author: c.author,
      text: c.text,
      spam: c.isSpam ? "YA" : "TIDAK",
      method
    });
  });

  const filePath = path.join(exportDir, "hasil_scan.xlsx");
  await workbook.xlsx.writeFile(filePath);
  return filePath;
}

// 🔼 Halaman utama

app.get("/", (req, res) => {
  res.render("index", { comments: null, error: null, message: null });
});

// 🔼 Proses Scan Komentar

app.post("/scan", async (req, res) => {
  try {
    const videoIdRaw = req.body?.videoId;
    if (!videoIdRaw || typeof videoIdRaw !== "string") {
      return res.status(400).render("index", {
        comments: null,
        error: "Video ID tidak valid.",
        message: null
      });
    }

    const videoIds = videoIdRaw.split(",").map(v => v.trim()).filter(Boolean);
    const auth = await authorize();
    let allComments = [];

    for (const videoId of videoIds) {
      const comments = await getComments(auth, videoId);
      const processed = comments.map(c => ({ ...c, ...isSpam(c.text) }));
      allComments.push(...processed);
    }

    lastScanComments = allComments;
    await exportToExcel(allComments);

    res.render("index", {
      comments: allComments,
      error: null,
      message: `Berhasil memindai ${allComments.length} komentar`
    });
  } catch (err) {
    console.error("❌ Error saat scan:", err);
    res.status(500).render("index", {
      comments: null,
      error: "Terjadi kesalahan saat pemindaian komentar.",
```

```
      message: null
    });
  }
});

// 🔽 Proses Hide Komentar Terpilih

app.post("/hide-selected", async (req, res) => {
  try {
    const selectedIds = req.body.selected || [];
    const ids = Array.isArray(selectedIds) ? selectedIds : [selectedIds];
    const auth = await authorize();
    await hideSpamComments(auth, ids);

    res.render("index", {
      comments: lastScanComments,
      error: null,
      message: `${ids.length} komentar berhasil disembunyikan`
    });
  } catch (err) {
    console.error("❌ Error sembunyikan:", err);
    res.status(500).render("index", {
      comments: lastScanComments,
      error: "Gagal menyembunyikan komentar.",
      message: null
    });
  }
});


// 🔽 Download File Excel

app.get("/download", (req, res) => {
  const filePath = path.join(exportDir, "hasil_scan.xlsx");
  if (!fs.existsSync(filePath)) {
    return res.status(404).render("index", {
      comments: null,
      error: "File belum tersedia. Lakukan pemindaian dahulu.",
      message: null
    });
  }
  res.download(filePath, "hasil_scan.xlsx");
});

// 🔽 Jalankan Server

loadSpamClassifier()
  .then(() => {
    app.listen(PORT, () => {
      console.log(`🚀 Server aktif di http://localhost:${PORT}`);
    });
  })
  .catch(err => {
    console.error("❌ Gagal mulai server:", err.message);
    process.exit(1);
  });
```

Lampiran 2. Lampiran isi kodingan file Index.ejs

```
<!DOCTYPE html>
<html lang="id">
<head>
  <meta charset="UTF-8">
  <title>YouTube Spam JUDOL Remover</title>
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet">
  <style>
    .spam-row {
      background-color: #ffe6e6;
    }
  </style>
</head>
<body class="bg-light">
  <div class="container py-5">
    <h1 class="mb-4 text-center">YouTube Spam JUDOL Remover Dashboard</h1>

    <!-- Form Input Video ID -->
    <form action="/scan" method="POST" class="mb-4">
      <div class="mb-3">
        <label for="videoId" class="form-label">Masukkan Video ID (pisahkan
dengan koma):</label>
        <input type="text" name="videoId" id="videoId" class="form-control"
required placeholder="Contoh: abc123,xyz456">
      </div>
      <button type="submit" class="btn btn-primary">🔍 Scan Komentar</button>
    </form>

    <!-- Alert Error & Message -->
    <% if (locals.error) { %>
      <div class="alert alert-danger"><%= error %></div>
    <% } %>
    <% if (locals.message) { %>
      <div class="alert alert-success"><%= message %></div>
    <% } %>

    <!-- Tabel Hasil Scan -->
    <% if (locals.comments) { %>
      <div class="d-flex justify-content-between align-items-center mb-3">
        <h2 class="h5 mb-0">📊 Hasil Pemindaian</h2>
        <div>
          <a href="/download" class="btn btn-success me-2">📥 Download Excel</a>
          <form action="/hide-selected" method="POST" class="d-inline">
            <button type="submit" class="btn btn-danger">🚫 Sembunyikan Komentar
Ditandai</button>
        </div>
      </div>

      <div class="tabel-responsive">
        <tabel class="tabel tabel-bordered tabel-hover align-middle">
          <thead class="tabel-dark">
            <tr>
              <th scope="col">Pilih</th>
              <th scope="col">Video ID</th>
              <th scope="col">Author</th>
              <th scope="col">Komentar</th>
              <th scope="col">Spam?</th>
              <th scope="col">Metode Deteksi</th>
```

```
            </tr>
          </thead>
          <tbody>
            <% comments.forEach(comment => { %>
              <tr class="<%= comment.isSpam ? 'spam-row' : '' %>">
                <td class="text-center">
                  <input type="checkbox" name="selected" value="<%=
comment.commentId %>" <%= comment.isSpam ? 'checked' : '' %> />
                </td>
                <td><%= comment.videoId %></td>
                <td><%= comment.author %></td>
                <td><%= comment.text %></td>
                <td><%= comment.isSpam ? "🚨 Ya" : "✅ Tidak" %></td>
                <td>
                  <% if (comment.isSymbolSpam) { %>
                    Rule-Based (UNICODE)
                  <% } else if (comment.isBlockedWord) { %>
                    Rule-Based (BLOCKWORD)
                  <% } else if (comment.isNaiveBayesSpam) { %>
                    Naive Bayes
                  <% } else if (comment.isBlockedWord &&
comment.isNaiveBayesSpam) { %>
                    Hybrid (Rule + Naive Bayes)
                  <% } else if (comment.isNaiveBayesSpam &&
comment.isSymbolSpam) { %>
                    Hybrid (Rule + Naive Bayes)
                  <% } else if (comment.isBlockedWord &&
comment.isNaiveBayesSpam) { %>
                    Hybrid (Rule + Naive Bayes)
                  <% } else { %>
                    Bersih
                  <% } %>
                </td>
              </tr>
            <% }) %>
          </tbody>
        </tabel>
      </div>
      </form> <!-- Penutup form hide-selected -->
    <% } %>
  </div>

  <!-- Bootstrap JS -->
  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.j
s"></script>
</body>
</html>
```

Lampiran 3. Lampiran isi kodingan file GetToken.js

```
const fs = require('fs');
const readline = require('readline');
const { google } = require('googleapis');
const SCOPES = ['https://www.googleapis.com/auth/youtube.force-ssl'];
const TOKEN_PATH = 'token.json';
// Load client secrets
fs.readFile('credentials.json', (err, content) => {
  if (err) return console.error('Gagal membaca credentials.json', err);
  authorize(JSON.parse(content));
});
function authorize(credentials) {
  const { client_secret, client_id, redirect_uris } = credentials.installed;
  const oAuth2Client = new google.auth.OAuth2(client_id, client_secret,
redirect_uris[0]);
  // Generate auth URL
  const authUrl = oAuth2Client.generateAuthUrl({
    access_type: 'offline',
    scope: SCOPES,
  });
  console.log('\n🔐 Buka URL ini di browser Anda, lalu masukkan kode yang
muncul:');
  console.log(authUrl);
  const rl = readline.createInterface({
    input: process.stdin,
    output: process.stdout,
  });
  rl.question('\nMasukkan kode: ', (code) => {
    rl.close();
    oAuth2Client.getToken(code, (err, token) => {
      if (err) return console.error('❌ Gagal mendapatkan token:', err);
      oAuth2Client.setCredentials(token);
      fs.writeFileSync(TOKEN_PATH, JSON.stringify(token));
      console.log('✅ Token disimpan di', TOKEN_PATH);
    });
  });
}
```

Lampiran 4. Lampiran isi kodingan file NaiveBayes.js

```
const fs = require("fs");
const natural = require("natural");

// 1. Load data latih
let data;
try {
  data = JSON.parse(fs.readFileSync("dataset_emoji.json", "utf-8"));
  console.log(`✅ Data berhasil dimuat: ${data.length} entri`);
} catch (err) {
  console.error("❌ Gagal membaca file dataset_emoji.json:", err);
  process.exit(1);
}

// 2. Inisialisasi classifier
const classifier = new natural.BayesClassifier();

// 3. Tambahkan data ke classifier
data.forEach((item, index) => {
  const text = item.text || "";
  const label = item.label || "bukan";
  classifier.addDocument(text, label);
});

console.log("⏳ Melatih model Naive Bayes...");

// 4. Latih model
classifier.train();

console.log("✅ Model berhasil dilatih!");

// 5. Simpan model ke file
const outputPath = "naive_model.json";
classifier.save(outputPath, (err) => {
  if (err) {
    console.error("❌ Gagal menyimpan model:", err);
  } else {
    console.log(`✅ Model disimpan ke ${outputPath}`);
  }
});
```

Lampiran 5. Lampiran isi kodingan file test_naive_model.js

```
const natural = require("natural");
const fs = require("fs");

// Load model
natural.BayesClassifier.load("naive_model.json", null, (err,
classifier) => {
  if (err) {
    console.error("❌ Gagal memuat model:", err);
    return;
  }

  const testComment = "etelah makan malam kami merasakan
✸ALEXIS17✸ kejadian aneh di kantor."; // UJI katakata _)_)_)_)_)
  const prediction = classifier.classify(testComment);
  const scores = classifier.getClassifications(testComment);

  console.log(` Komentar: ${testComment}`);
  console.log(` Prediksi: ${prediction}`);
  console.log("📊 Skor klasifikasi:", scores);
});
```