```
/* ============================================== */
/*   GESTIONNAIRE DE TITRES YOUTUBE AVEC IA    */
/*   Version sécurisée - sans tokens exposés   */
/* ============================================== */

/* ===== FICHIER: server.js ===== */

const express = require('express');
const { google } = require('googleapis');
const path = require('path');

const app = express();
const PORT = 3005;

/* ====== CONFIGURATION (À PERSONNALISER) ====== */
const CLIENT_ID = 'YOUR_GOOGLE_CLIENT_ID.apps.googleusercontent.com';
const CLIENT_SECRET = 'YOUR_GOOGLE_CLIENT_SECRET';
const REDIRECT_URI = 'http://localhost:3005';
const REFRESH_TOKEN = 'YOUR_GOOGLE_REFRESH_TOKEN';

// Clé API OpenAI (remplacez par votre vraie clé)
const OPENAI_API_KEY = 'sk-proj-YOUR_OPENAI_API_KEY_HERE';

/* ====== OAuth & YouTube API ====== */
const oauth2Client = new google.auth.OAuth2(CLIENT_ID, CLIENT_SECRET, REDIRECT_URI);
if (REFRESH_TOKEN) oauth2Client.setCredentials({ refresh_token: REFRESH_TOKEN });

const youtube = google.youtube({ version: 'v3', auth: oauth2Client });

/* ====== Middlewares ====== */
app.use(express.json());
```

```
app.use(express.static(path.join(__dirname, 'public')));


/* ====== Helpers ====== */
async function ensureValidToken() {
  const access = await oauth2Client.getAccessToken();
  if (!access?.token) throw new Error('Impossible d\'obtenir un access token.');
}


/* ====== ChatGPT Integration ====== */
async function callChatGPT(prompt) {
  try {
    const response = await fetch('https://api.openai.com/v1/chat/completions', {
      method: 'POST',
      headers: {
        'Authorization': `Bearer ${OPENAI_API_KEY}`,
        'Content-Type': 'application/json',
      },
      body: JSON.stringify({
        model: 'gpt-3.5-turbo', // ou 'gpt-4' si vous avez accès
        messages: [
          {
            role: 'system',
            content: 'Tu es un expert en optimisation de titres YouTube. Tu dois créer des titres accrocheurs, optimisés SEO, qui respectent les meilleures pratiques YouTube tout en restant authentiques.'
          },
          {
            role: 'user',
            content: prompt
          }
        ],
        max_tokens: 100,
```

```
            temperature: 0.7,
          }),
        });

        if (!response.ok) {
          throw new Error(`OpenAI API Error: ${response.status} ${response.statusText}`);
        }

        const data = await response.json();
        return data.choices[0]?.message?.content?.trim() || '';
    } catch (error) {
        console.error('Erreur ChatGPT API:', error);
        throw error;
    }
}

function buildPrompt(originalTitle, style, category, language, views) {
    const styleDescriptions = {
        engaging: 'engageant avec des émojis et des questions qui suscitent la curiosité',
        professional: 'professionnel et informatif',
        clickbait: 'accrocheur mais éthique, qui donne envie de cliquer',
        educational: 'éducatif et didactique',
        tutorial: 'orienté tutoriel/guide pratique'
    };

    const categoryContext = {
        tech: 'technologie et informatique',
        gaming: 'jeux vidéo et gaming',
        lifestyle: 'style de vie et bien-être',
        education: 'éducation et apprentissage',
        entertainment: 'divertissement',
```

```
    cooking: 'cuisine et gastronomie',
    travel: 'voyage et découverte'
  };

  const viewsContext = views > 10000 ? 'Cette vidéo a déjà une bonne audience' : 'Cette vidéo pourrait bénéficier de plus de visibilité';

  return `
Améliore ce titre YouTube : "${originalTitle}"

Contraintes :
- Maximum 100 caractères
- Style : ${styleDescriptions[style] || 'engageant'}
- Catégorie : ${categoryContext[category] || 'général'}
- Langue : ${language === 'fr' ? 'français' : language === 'en' ? 'anglais' : 'espagnol'}
- ${viewsContext} (${views} vues actuellement)

Le nouveau titre doit :
1. Être plus accrocheur que l'original
2. Inclure des mots-clés pertinents pour le SEO
3. Susciter la curiosité
4. Rester fidèle au contenu original
5. Respecter le style demandé

Réponds uniquement avec le nouveau titre, sans guillemets ni explications.
  `.trim();
}

/* ====== Routes OAuth ====== */
const SCOPES = ['https://www.googleapis.com/auth/youtube.force-ssl'];
```

```
app.get('/auth', (req, res) => {
  const url = oauth2Client.generateAuthUrl({
    access_type: 'offline',
    prompt: 'consent',
    include_granted_scopes: true,
    scope: SCOPES,
  });
  res.redirect(url);
});

app.get('/oauth2callback', async (req, res) => {
  try {
    const { code } = req.query;
    const { tokens } = await oauth2Client.getToken(code);
    oauth2Client.setCredentials(tokens);
    res.send(` <h2> ✅ Tokens reçus</h2><pre>${JSON.stringify(tokens, null, 2)}</pre>
<p>Copie le <b>refresh_token</b> dans REFRESH_TOKEN puis redémarre.</p>`);
  } catch (e) {
    res.status(500).send('Erreur OAuth2: ' + e.message);
  }
});

/* ====== API Routes ====== */

/** GET /api/my-videos - liste toutes les vidéos via la playlist d'uploads */
app.get('/api/my-videos', async (_req, res) => {
  try {
    await ensureValidToken();

    const ch = await youtube.channels.list({ part: 'contentDetails', mine: true });
    const items = ch.data.items || [];
```

```
if (!items.length) return res.json({ success: true, total: 0, videos: [] });

const uploadsId = items[0].contentDetails?.relatedPlaylists?.uploads;
if (!uploadsId) return res.json({ success: true, total: 0, videos: [] });

let videoIds = [];
let nextPageToken = '';
do {
  const pl = await youtube.playlistItems.list({
    part: 'contentDetails',
    playlistId: uploadsId,
    maxResults: 50,
    pageToken: nextPageToken,
  });
  const batch = (pl.data.items || [])
    .map(it => it.contentDetails?.videoId)
    .filter(Boolean);
  videoIds.push(...batch);
  nextPageToken = pl.data.nextPageToken;
} while (nextPageToken);

if (!videoIds.length) return res.json({ success: true, total: 0, videos: [] });

const detailed = [];
for (let i = 0; i < videoIds.length; i += 50) {
  const chunk = videoIds.slice(i, i + 50);
  const vd = await youtube.videos.list({
    part: 'snippet,statistics,status',
    id: chunk.join(','),
  });
  if (vd.data.items?.length) detailed.push(...vd.data.items);
```

```
    }

    const videos = detailed.map(v => ({
      id: v.id,
      title: v.snippet?.title || '',
      views: Number(v.statistics?.viewCount || 0),
      privacyStatus: v.status?.privacyStatus || 'public',
    }));

    res.json({ success: true, total: videos.length, videos });
  } catch (error) {
    console.error('Erreur /api/my-videos:', error?.response?.data || error.message);
    res.status(500).json({
      success: false,
      error: error?.response?.data?.error?.message || error.message,
    });
  }
});

/** PUT /api/videos/:videoId/title - met à jour le titre d'une vidéo */
app.put('/api/videos/:videoId/title', async (req, res) => {
  try {
    await ensureValidToken();

    const { videoId } = req.params;
    const { title } = req.body;

    if (!title?.trim()) return res.status(400).json({ success: false, error: 'Le titre ne peut pas être vide' });
    if (title.length > 100) return res.status(400).json({ success: false, error: 'Le titre ne doit pas dépasser 100 caractères' });
```

```
    const vr = await youtube.videos.list({ part: 'snippet', id: videoId });
    if (!vr.data.items?.length) return res.status(404).json({ success: false, error: 'Vidéo non trouvée' });

    const snippet = vr.data.items[0].snippet;
    const oldTitle = snippet.title;

    const ur = await youtube.videos.update({
      part: 'snippet',
      requestBody: { id: videoId, snippet: { ...snippet, title: title.trim() } },
    });

    res.json({ success: true, message: 'Titre modifié', oldTitle, newTitle: title.trim(), videoId, data: ur.data });
  } catch (error) {
    console.error('Erreur /api/videos/:videoId/title:', error?.response?.data || error.message);
    res.status(500).json({
      success: false,
      error: error?.response?.data?.error?.message || error.message,
    });
  }
});

/** POST /api/enhance-titles - améliore les titres avec ChatGPT */
app.post('/api/enhance-titles', async (req, res) => {
  try {
    if (!OPENAI_API_KEY || OPENAI_API_KEY === 'sk-proj-YOUR_OPENAI_API_KEY_HERE') {
      return res.status(400).json({
        success: false,
        error: 'Clé API OpenAI non configurée. Ajoutez votre clé dans OPENAI_API_KEY.'
      });
    }
```

```
const { videos, style = 'engaging', category = 'auto', language = 'fr' } = req.body;

if (!videos || !Array.isArray(videos)) {
  return res.status(400).json({
    success: false,
    error: 'Le paramètre "videos" doit être un tableau de vidéos'
  });
}

const enhancedTitles = {};
const errors = [];

// Traiter chaque vidéo
for (const video of videos) {
  try {
    console.log(`🤖 Amélioration du titre: "${video.title}"`);

    const prompt = buildPrompt(video.title, style, category, language, video.views);
    const enhancedTitle = await callChatGPT(prompt);

    // Vérifier que le titre respecte les contraintes
    if (enhancedTitle && enhancedTitle.length <= 100) {
      enhancedTitles[video.id] = enhancedTitle;
      console.log(`✅ "${video.title}" → "${enhancedTitle}"`);
    } else {
      // Fallback si le titre est trop long
      const truncated = enhancedTitle.substring(0, 97) + '...';
      enhancedTitles[video.id] = truncated;
      console.log(`⚠️ Titre tronqué pour ${video.id}`);
    }
```

```
        // Petit délai pour éviter de surcharger l'API OpenAI
        await new Promise(resolve => setTimeout(resolve, 1000));

      } catch (error) {
        console.error(`❌ Erreur pour la vidéo ${video.id}:`, error.message);
        errors.push({
          videoId: video.id,
          originalTitle: video.title,
          error: error.message
        });

        // En cas d'erreur, garder le titre original
        enhancedTitles[video.id] = video.title;
      }
    }

    res.json({
      success: true,
      enhancedTitles,
      processed: videos.length,
      errors: errors.length > 0 ? errors : undefined,
      message: `${Object.keys(enhancedTitles).length} titres améliorés${errors.length > 0 ? `
(${errors.length} erreurs)` : ''}`
    });

  } catch (error) {
    console.error('❌ /api/enhance-titles:', error);
    res.status(500).json({
      success: false,
      error: error.message
```

```
    });
  }
});

/* ====== Route de base ====== */
app.get('/', (req, res) => {
  res.sendFile(path.join(__dirname, 'index.html'));
});

/* ====== Démarrage du serveur ====== */
app.listen(PORT, () => {
  console.log(`
🎬 YOUTUBE TITLE MANAGER AVEC IA
================================
🌐 Interface  : http://localhost:${PORT}/
🔐 OAuth      : http://localhost:${PORT}/auth

🚀 APIs disponibles:
- GET  /api/my-videos
- PUT  /api/videos/:videoId/title
- POST /api/enhance-titles

⚠️ CONFIGURATION REQUISE:
${CLIENT_ID.includes('YOUR_') ? '❌ CLIENT_ID non configuré' : '✅ CLIENT_ID configuré'}
${CLIENT_SECRET.includes('YOUR_') ? '❌ CLIENT_SECRET non configuré' : '✅ CLIENT_SECRET configuré'}
${REFRESH_TOKEN.includes('YOUR_') ? '❌ REFRESH_TOKEN non configuré' : '✅ REFRESH_TOKEN configuré'}
${OPENAI_API_KEY.includes('YOUR_') ? '❌ OPENAI_API_KEY non configuré' : '✅ OPENAI_API_KEY configuré'}
```

```
🔧 Pour configurer: Modifiez les constantes en haut de ce fichier
`);
});

/* ================================================ */

/* ===== FICHIER: index.html ===== */

<!doctype html>
<html lang="fr">
<head>
  <meta charset="utf-8"/>
  <title>Gestion des titres YouTube avec IA</title>
  <meta name="viewport" content="width=device-width, initial-scale=1"/>

  <style>
    /* Design System */
    :root{
      --bg: #0b1020;
      --bg-elev: #0f152a;
      --card: #121a34;
      --muted: #9aa4bf;
      --text: #e9ecf8;
      --border: #1e2a55;
      --accent: #6ea8ff;
      --accent-2: #8b5cf6;
      --danger: #ff6b6b;
      --success: #4ade80;
      --warning: #fbbf24;
      --shadow: 0 10px 30px rgba(0,0,0,.35);
      --radius: 14px;
```

```
    --radius-sm: 10px;
    --ring: 0 0 0 3px rgba(110,168,255,.35);
  }

  @media (prefers-color-scheme: light){
    :root{
      --bg:#f7f9ff; --bg-elev:#ffffff; --card:#ffffff; --text:#0f1222; --muted:#5b6174; --
border:#e6e9f5; --shadow: 0 10px 25px rgba(17, 23, 61, 0.12);
    }
  }

  /* Base */
  *{box-sizing:border-box}
  html,body{height:100%}
  body{
    margin:0;
    font-family: ui-sans-serif, system-ui, -apple-system, Segoe UI, Roboto, Arial;
    color:var(--text);
    background:
      radial-gradient(1200px 600px at 10% -10%, rgba(139,92,246,.20), transparent 60%),
      radial-gradient(900px 500px at 110% 10%, rgba(110,168,255,.18), transparent 60%),
      var(--bg);
  }

  .container{max-width:1100px;margin:32px auto;padding:0 20px}

  h1{
    margin:0 0 18px;
    font-weight:800;
    letter-spacing: .2px;
    font-size: clamp(26px, 4vw, 38px);
```

```
    line-height:1.1;
    background: linear-gradient(90deg, var(--accent), var(--accent-2));
    -webkit-background-clip:text; background-clip:text; color:transparent;
  }
  .subtitle{margin-top:-4px;color:var(--muted);font-size:14px}

  /* Toolbar */
  .toolbar{
    position:sticky; top:0; z-index:5;
    display:flex; gap:10px; flex-wrap:wrap; align-items:center;
    padding:12px; margin:18px 0 14px;
    background: linear-gradient(180deg, rgba(255,255,255,.02), rgba(255,255,255,0)) , var(--bg-elev);
    border:1px solid var(--border); border-radius: var(--radius);
    box-shadow: var(--shadow);
    backdrop-filter: blur(8px);
  }

  button{
    appearance:none; border:0; cursor:pointer; font-weight:600;
    padding:10px 14px; border-radius: var(--radius-sm);
    transition: transform .08s ease, box-shadow .2s ease, background .2s ease;
    box-shadow: 0 6px 16px rgba(0,0,0,.18);
  }
  button:active{transform:translateY(1px)}
  button:disabled{opacity:.5;cursor:not-allowed;transform:none!important}

  .primary{
    color:#07132a; background: linear-gradient(135deg, var(--accent), #a0c2ff 70%);
  }
  .primary:hover:not(:disabled){box-shadow: 0 10px 22px rgba(110,168,255,.35)}
```

```
.success{
  color:#0a2e0a; background: linear-gradient(135deg, var(--success), #6ee76e 70%);
}
.success:hover:not(:disabled){box-shadow: 0 10px 22px rgba(74,222,128,.35)}

.warning{
  color:#422006; background: linear-gradient(135deg, var(--warning), #fcd34d 70%);
}
.warning:hover:not(:disabled){box-shadow: 0 10px 22px rgba(251,191,36,.35)}

.ghost{
  background:transparent; color:var(--text);
  border:1px solid var(--border);
}
.ghost:hover{background:rgba(255,255,255,.04)}

input[type="search"], input.title, select{
  flex:1; min-width:200px;
  background: var(--bg-elev);
  color: var(--text);
  border:1px solid var(--border);
  border-radius: var(--radius-sm);
  padding:10px 12px;
  outline:none;
  transition: box-shadow .15s ease, border-color .15s ease, background .2s ease;
}
input::placeholder{color:color-mix(in oklab, var(--muted) 70%, transparent)}
input:focus, select:focus{box-shadow: var(--ring); border-color: color-mix(in oklab, var(--accent) 60%, var(--border))}
```

```
/* Enhancement Panel */
.enhancement-panel{
  background: linear-gradient(180deg, rgba(255,255,255,.02), rgba(255,255,255,0)), var(--card);
  border:1px solid var(--border);
  border-radius: var(--radius);
  padding:16px;
  margin:16px 0;
  box-shadow: var(--shadow);
}
.enhancement-options{
  display:grid;
  grid-template-columns: repeat(auto-fit, minmax(280px, 1fr));
  gap:12px;
  margin-bottom:16px;
}
.option-group{
  display:flex;
  flex-direction:column;
  gap:6px;
}
.option-group label{
  font-weight:600;
  font-size:14px;
  color:var(--text);
}

/* Cards */
.badge{display:inline-flex;align-items:center;gap:6px;padding:4px 8px;border-radius:999px;font-size:12px;border:1px solid var(--border);background:rgba(255,255,255,.04)}
.views{font-variant-numeric:tabular-nums}
```

```
.changed{border-color:var(--success)!important;background:rgba(74,222,128,.1)!important}
.preview{color:var(--warning);font-style:italic;margin-top:4px;font-size:13px}

/* Progress */
.progress{
  width:100%;
  height:6px;
  background:var(--border);
  border-radius:3px;
  overflow:hidden;
  margin:10px 0;
}
.progress-bar{
  height:100%;
  background:linear-gradient(90deg, var(--accent), var(--accent-2));
  width:0%;
  transition:width .3s ease;
}

/* List & Cards */
#list{display:grid; grid-template-columns: repeat(auto-fill, minmax(350px, 1fr)); gap:14px}

.card{
  border:1px solid var(--border);
  border-radius: var(--radius);
  background: linear-gradient(180deg, rgba(255,255,255,.02), rgba(255,255,255,0)), var(--card);
  padding:16px; box-shadow: var(--shadow);
  transition: transform .2s ease, box-shadow .2s ease, border-color .2s ease;
  animation: fadeIn .35s ease both;
}
```

```
    .card:hover{transform: translateY(-3px); box-shadow: 0 18px 36px rgba(0,0,0,.28); border-color: color-mix(in oklab, var(--accent) 40%, var(--border))}

    .row{display:flex;gap:10px;align-items:center;flex-wrap:wrap}
    .row + .row{margin-top:10px}

    /* Counters & Errors */
    #count{margin:8px 2px}
    #err{color:var(--danger); background: color-mix(in oklab, var(--danger) 10%, transparent); border:1px solid color-mix(in oklab, var(--danger) 30%, transparent); padding:10px 12px; border-radius: var(--radius-sm); display:none}
    #err.show{display:block}

    /* Animations */
    @keyframes fadeIn{from{opacity:0; transform:translateY(6px)}to{opacity:1; transform:translateY(0)}}
    @keyframes pulse{0%,100%{opacity:1}50%{opacity:.5}}
    .processing{animation:pulse 2s infinite}

    .small{color:var(--muted);font-size:12px}
  </style>
</head>

<body>
<div class="container">
  <h1>Gestion des titres YouTube</h1>
  <p class="subtitle">Renommez rapidement vos vidéos, avec filtre en direct et sauvegarde instantanée.</p>

  <div class="toolbar">
    <button class="primary" onclick="load()">Charger les vidéos</button>
    <input id="q" type="search" placeholder="Filtrer par titre… (live)" oninput="filterRows()"/>
    <a class="ghost" href="/auth">Autoriser l'application (si pas de refresh token)</a>
```

```
  </div>


  <div class="enhancement-panel">
    <h3 style="margin:0 0 12px;color:var(--text);font-size:18px">🚀 Amélioration automatique des titres</h3>

    <div class="enhancement-options">
      <div class="option-group">
        <label>Style de titre</label>
        <select id="titleStyle">
          <option value="engaging">Engageant (émojis, questions)</option>
          <option value="professional">Professionnel</option>
          <option value="clickbait">Accrocheur (clickbait éthique)</option>
          <option value="educational">Éducatif</option>
          <option value="tutorial">Tutoriel/Guide</option>
        </select>
      </div>

      <div class="option-group">
        <label>Catégorie de contenu</label>
        <select id="contentCategory">
          <option value="auto">Auto-détection</option>
          <option value="tech">Technologie</option>
          <option value="gaming">Gaming</option>
          <option value="lifestyle">Lifestyle</option>
          <option value="education">Éducation</option>
          <option value="entertainment">Divertissement</option>
          <option value="cooking">Cuisine</option>
          <option value="travel">Voyage</option>
        </select>
      </div>
```

```
        <div class="option-group">
          <label>Langue cible</label>
          <select id="targetLanguage">
            <option value="fr">Français</option>
            <option value="en">Anglais</option>
            <option value="es">Espagnol</option>
          </select>
        </div>
      </div>

      <div class="row">
        <button class="success" onclick="enhanceAllTitles()" id="enhanceBtn">✨ Améliorer tous les titres</button>
        <button class="warning" onclick="previewEnhancements()" id="previewBtn">👁️ Aperçu des changements</button>
        <button class="ghost" onclick="resetAllTitles()" id="resetBtn">↻ Réinitialiser</button>
      </div>

      <div id="progress" class="progress" style="display:none">
        <div id="progressBar" class="progress-bar"></div>
      </div>
      <div id="progressText" class="small" style="display:none"></div>
    </div>

    <div id="err"></div>
    <div id="count" class="small"></div>
    <div id="list"></div>
  </div>

  <script>
    let rows = []; // {id,title,views,privacyStatus}
```

```
  let filtered = [];
  let originalTitles = {}; // backup des titres originaux
  let enhancedTitles = {}; // titres améliorés en preview

  function formatViews(n){ n=Number(n||0); if(n>=1e6) return (n/1e6).toFixed(1)+'M'; if(n>=1e3) return (n/1e3).toFixed(1)+'K'; return n.toString(); }
  function escapeHtml(s){ return s.replace(/&/g,'&amp;').replace(/</g,'&lt;').replace(/>/g,'&gt;').replace(/"/g,'&quot;'); }

  // Appel à l'API ChatGPT pour améliorer les titres
  async function enhanceTitlesWithAI(videos, style, category, language) {
    try {
      const response = await fetch('/api/enhance-titles', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
        },
        body: JSON.stringify({
          videos: videos,
          style: style,
          category: category,
          language: language
        })
      });

      const result = await response.json();

      if (!result.success) {
        throw new Error(result.error || 'Erreur lors de l\'amélioration des titres');
      }

      return result;
```

```
  } catch (error) {
    console.error('Erreur API enhance-titles:', error);
    throw error;
  }
}

async function load(){
  const err = document.getElementById('err');
  const list = document.getElementById('list');
  err.classList.remove('show'); err.textContent = '';
  list.innerHTML = '<div class="small">Chargement…</div>';
  try{
    const res = await fetch('/api/my-videos');
    const data = await res.json();
    if(!data.success){
      list.innerHTML = '';
      err.textContent = 'Erreur API: ' + (data.error || 'inconnue');
      err.classList.add('show');
      return;
    }
    rows = data.videos || [];
    filtered = rows;

    // Sauvegarder les titres originaux
    originalTitles = {};
    rows.forEach(v => originalTitles[v.id] = v.title);

    render();
  }catch(e){
    list.innerHTML = '';
    err.textContent = 'Erreur réseau: ' + e.message;
```

```
        err.classList.add('show');
      }
    }

    function render(){
      const list = document.getElementById('list');
      list.innerHTML = '';
      document.getElementById('count').textContent = filtered.length ? ` ${filtered.length} vidéos` : 'Aucune vidéo';

      filtered.forEach(v=>{
        const hasPreview = enhancedTitles[v.id];
        const isChanged = hasPreview && enhancedTitles[v.id] !== originalTitles[v.id];

        const card = document.createElement('div');
        card.className = 'card';
        card.innerHTML = `
        <div class="row">
          <span class="badge"><b>ID</b>: <code>${v.id}</code></span>
          <span class="badge views">${formatViews(v.views)} vues</span>
          <span class="badge">${v.privacyStatus}</span>
        </div>
        <div class="row">
          <input class="title ${isChanged ? 'changed' : ''}" id="t-${v.id}" value="${escapeHtml(hasPreview ? enhancedTitles[v.id] : v.title)}" />
          <button class="primary" onclick="save('${v.id}')">Enregist

/* ========================================== */
/*   SUITE DU FICHIER index.html (JavaScript)  */
/* ========================================== */

        <div class="row">
```

```
          <input class="title ${isChanged ? 'changed' : ''}" id="t-${v.id}"
value="${escapeHtml(hasPreview ? enhancedTitles[v.id] : v.title)}" />
          <button class="primary" onclick="save('${v.id}')">Enregistrer</button>
        </div>
        ${hasPreview && isChanged ? `<div class="preview">Original:
"${escapeHtml(originalTitles[v.id])}"</div>` : ''}
      `;
      list.appendChild(card);
    });
  }

  function filterRows(){
    const q = document.getElementById('q').value.toLowerCase();
    filtered = rows.filter(r => r.title.toLowerCase().includes(q));
    render();
  }

  async function save(id){
    const el = document.getElementById('t-'+id);
    const title = el.value.trim();
    if(!title) return alert('Titre vide');
    if(title.length > 100) return alert('Max 100 caractères');
    try{
      const r = await fetch('/api/videos/'+id+'/title', {
        method:'PUT',
        headers:{'Content-Type':'application/json'},
        body: JSON.stringify({ title })
      });
      const out = await r.json();
      if(out.success){
        // Mettre à jour le titre dans les données locales
        const video = rows.find(v => v.id === id);
```

```
        if(video) video.title = title;
        alert('✅ Titre enregistré');
      }else{
        alert('❌ ' + (out.error || 'Erreur inconnue'));
      }
    }catch(e){
      alert('❌ Erreur réseau: ' + e.message);
    }
}

async function previewEnhancements() {
  if (!rows.length) {
    return alert('⚠️ Chargez d\'abord vos vidéos avec le bouton "Charger les vidéos"');
  }

  const style = document.getElementById('titleStyle').value;
  const category = document.getElementById('contentCategory').value;
  const language = document.getElementById('targetLanguage').value;

  const btn = document.getElementById('previewBtn');
  const progress = document.getElementById('progress');
  const progressBar = document.getElementById('progressBar');
  const progressText = document.getElementById('progressText');

  btn.disabled = true;
  btn.textContent = '🤖 ChatGPT travaille...';
  progress.style.display = 'block';
  progressText.style.display = 'block';
  progressText.textContent = 'Connexion à ChatGPT...';

  try {
```

```
        // Utiliser les vidéos filtrées ou toutes les vidéos
        const videosToProcess = filtered.length > 0 ? filtered : rows;

        progressText.textContent = `Amélioration de ${videosToProcess.length} titres avec ChatGPT...`;
        progressBar.style.width = '50%';

        const result = await enhanceTitlesWithAI(videosToProcess, style, category, language);

        enhancedTitles = result.enhancedTitles || {};
        progressBar.style.width = '100%';

        render();

        let message = `✅ ${result.processed} titres améliorés par ChatGPT !`;
        if (result.errors && result.errors.length > 0) {
          message += `\n⚠️ ${result.errors.length} erreurs (titres conservés)`;
          console.warn('Erreurs lors de l\'amélioration:', result.errors);
        }

        alert(message + '\nLes titres modifiés sont surlignés en vert.');

      } catch (error) {
        console.error('Erreur:', error);
        alert('❌ Erreur lors de l\'amélioration des titres: ' + error.message);
      } finally {
        btn.disabled = false;
        btn.textContent = '👁️ Aperçu des changements';
        progress.style.display = 'none';
        progressText.style.display = 'none';
        progressBar.style.width = '0%';
```

```
    }
  }

  async function enhanceAllTitles() {
    if(!Object.keys(enhancedTitles).length) {
      return alert('⚠️ Générez d\'abord un aperçu avec le bouton "Aperçu des changements"');
    }

    if(!confirm('Êtes-vous sûr de vouloir appliquer ces améliorations à tous vos titres YouTube ?')) {
      return;
    }

    const btn = document.getElementById('enhanceBtn');
    const progress = document.getElementById('progress');
    const progressBar = document.getElementById('progressBar');
    const progressText = document.getElementById('progressText');

    btn.disabled = true;
    btn.textContent = '⚙️ Traitement...';
    progress.style.display = 'block';
    progressText.style.display = 'block';

    const videosToUpdate = Object.keys(enhancedTitles);
    let completed = 0;
    let successful = 0;

    try {
      for(const videoId of videosToUpdate) {
        const newTitle = enhancedTitles[videoId];
        progressText.textContent = `Mise à jour ${completed + 1}/${videosToUpdate.length}: ${newTitle.substring(0, 50)}...`;
```

```
            try {
                const response = await fetch(` /api/videos/${videoId}/title` , {
                    method: 'PUT',
                    headers: {'Content-Type': 'application/json'},
                    body: JSON.stringify({ title: newTitle })
                });

                const result = await response.json();
                if(result.success) {
                    successful++;
                    // Mettre à jour les données locales
                    const video = rows.find(v => v.id === videoId);
                    if(video) video.title = newTitle;
                }

            } catch(e) {
                console.error('Erreur pour la vidéo', videoId, e);
            }

            completed++;
            progressBar.style.width = (completed / videosToUpdate.length * 100) + '%';

            // Petit délai pour ne pas surcharger l'API
            await new Promise(resolve => setTimeout(resolve, 200));
        }

        alert(` ✅ Traitement terminé !\n${successful}/${videosToUpdate.length} titres mis à jour avec succès.` );

        // Reset des previews
```

```
        enhancedTitles = {};
        render();

      } finally {
        btn.disabled = false;
        btn.textContent = '✨ Améliorer tous les titres';
        progress.style.display = 'none';
        progressText.style.display = 'none';
        progressBar.style.width = '0%';
      }
    }

    function resetAllTitles() {
      if(!Object.keys(enhancedTitles).length) {
        return alert('Aucun changement à annuler');
      }

      enhancedTitles = {};
      render();
      alert('✅ Tous les changements ont été annulés');
    }
</script>
</body>
</html>

/* ============================================== */
/*          FICHIER: package.json          */
/* ============================================== */

{
  "name": "youtube-title-manager-ai",
```

```
"version": "1.0.0",
"description": "Gestionnaire de titres YouTube avec amélioration par intelligence artificielle",
"main": "server.js",
"scripts": {
  "start": "node server.js",
  "dev": "nodemon server.js",
  "test": "echo \"No tests specified\" && exit 0"
},
"keywords": [
  "youtube",
  "api",
  "chatgpt",
  "openai",
  "title-optimization",
  "seo",
  "video-management",
  "artificial-intelligence"
],
"author": "Votre Nom",
"license": "MIT",
"dependencies": {
  "express": "^4.18.2",
  "googleapis": "^128.0.0"
},
"devDependencies": {
  "nodemon": "^3.0.1"
},
"engines": {
  "node": ">=16.0.0"
},
"repository": {
```

```
  "type": "git",
  "url": "https://github.com/votre-username/youtube-title-manager-ai.git"
 },
 "bugs": {
  "url": "https://github.com/votre-username/youtube-title-manager-ai/issues"
 },
 "homepage": "https://github.com/votre-username/youtube-title-manager-ai#readme"
}

/* ============================================== */
/*          FICHIER: .env (OPTIONNEL)      */
/* ============================================== */

# Configuration Google OAuth 2.0
GOOGLE_CLIENT_ID=YOUR_GOOGLE_CLIENT_ID.apps.googleusercontent.com
GOOGLE_CLIENT_SECRET=YOUR_GOOGLE_CLIENT_SECRET
GOOGLE_REFRESH_TOKEN=YOUR_GOOGLE_REFRESH_TOKEN

# Configuration OpenAI
OPENAI_API_KEY=sk-proj-YOUR_OPENAI_API_KEY_HERE

# Configuration Serveur
PORT=3005
REDIRECT_URI=http://localhost:3005

# Environnement (development, production)
NODE_ENV=development

/* ============================================== */
/*        GUIDE D'INSTALLATION ET USAGE     */
/* ============================================== */
```

=== ÉTAPE 1: Prérequis ===

- Node.js version 16+ installé
- Compte Google Cloud Console
- Compte OpenAI avec crédits
- Un canal YouTube avec des vidéos

=== ÉTAPE 2: Configuration Google Cloud ===

1. Allez sur https://console.cloud.google.com
2. Créez un nouveau projet ou sélectionnez-en un
3. Activez l'API "YouTube Data API v3"
4. Créez des identifiants OAuth 2.0:
   - Type d'application: Application Web
   - URI de redirection: http://localhost:3005
5. Notez le CLIENT_ID et CLIENT_SECRET

=== ÉTAPE 3: Configuration OpenAI ===

1. Créez un compte sur https://platform.openai.com
2. Ajoutez des crédits de facturation
3. Générez une clé API dans la section "API keys"
4. Copiez la clé (elle ne s'affiche qu'une fois)

=== ÉTAPE 4: Installation ===

1. Créez un dossier pour votre projet
2. Copiez tous les fichiers dedans
3. Ouvrez un terminal dans ce dossier
4. Installez les dépendances:
   npm install

=== ÉTAPE 5: Configuration ===

Méthode A - Modification directe:

- Ouvrez server.js
- Remplacez YOUR_GOOGLE_CLIENT_ID par votre vrai CLIENT_ID
- Remplacez YOUR_GOOGLE_CLIENT_SECRET par votre vrai SECRET
- Remplacez YOUR_OPENAI_API_KEY par votre vraie clé OpenAI
- Laissez YOUR_GOOGLE_REFRESH_TOKEN pour l'instant

Méthode B - Fichier .env (recommandé):
- Installez dotenv: npm install dotenv
- Créez un fichier .env avec vos vraies valeurs
- Ajoutez en haut de server.js: require('dotenv').config();
- Utilisez process.env.GOOGLE_CLIENT_ID au lieu des constantes

=== ÉTAPE 6: Première authentification ===
1. Lancez le serveur: npm start
2. Ouvrez http://localhost:3005/auth
3. Autorisez l'application à accéder à votre YouTube
4. Copiez le refresh_token affiché sur la page
5. Remplacez YOUR_GOOGLE_REFRESH_TOKEN par ce token
6. Relancez le serveur

=== ÉTAPE 7: Utilisation ===
1. Ouvrez http://localhost:3005
2. Cliquez "Charger les vidéos" pour voir vos vidéos
3. Configurez le style et la catégorie souhaités
4. Cliquez "Aperçu des changements" pour voir les améliorations IA
5. Vérifiez les suggestions et cliquez "Améliorer tous les titres"
6. Ou modifiez individuellement et cliquez "Enregistrer"

=== SÉCURITÉ IMPORTANTE ===
- Ne partagez jamais vos clés API publiquement
- Ajoutez .env à votre .gitignore si vous utilisez Git

- Surveillez vos quotas Google Cloud et OpenAI

- Testez d'abord sur quelques vidéos avant traitement en masse

- Sauvegardez vos titres originaux importants

=== COÛTS À SURVEILLER ===

- YouTube Data API: 10,000 unités/jour gratuites

- OpenAI: Facturation par token (~$0.002/1000 tokens)

- Estimation: ~50 vidéos = $0.10-0.30 selon la complexité

=== RÉSOLUTION DE PROBLÈMES ===

Erreur "Token invalide":

→ Refaites l'authentification sur /auth

Erreur "Quota dépassé":

→ Attendez le lendemain ou demandez une augmentation

Erreur OpenAI:

→ Vérifiez vos crédits et la validité de la clé

Titres trop longs:

→ L'app tronque automatiquement à 100 caractères

Pas de vidéos affichées:

→ Vérifiez que votre canal a bien des vidéos publiques

=== DÉVELOPPEMENT AVANCÉ ===

Pour personnaliser l'application:

- Modifiez les prompts dans buildPrompt()

- Ajoutez des styles de titres dans styleDescriptions

- Changez le design en modifiant le CSS

- Ajoutez des langues dans les options

- Implémentez des templates personnalisés

=== COMMANDES UTILES ===

npm start   # Lancer en production

npm run dev   # Lancer avec auto-restart (nécessite nodemon)

node server.js   # Lancer directement

curl localhost:3005/api/my-videos # Tester l'API

L'application est maintenant prête à optimiser vos titres YouTube avec l'IA !

/* == */

/*   FIN DU PROJET   */

/* == */