Swing — تطبيق شبيه بـ TikTok (Node.js + Express + MongoDB + React)

الهدف: تطبيق محلي لتجربة رفع الفيديو، فيد **Swing**. باسم TikTok لنسخة تشبه MVP هذا مستودع تعليمي يعمل كـ إعجاب، وتعليقات بسيطة. مخصّص للتعلم والتطوير — ليس للإنتاج بدون تحسينات autoplay gswipe، كامل الشاشة مع أمنية وأداء.

بنية المشروع

```
swing-mvp/
├─ backend/
│  ├─ package.json
│  ├─ server.js
│  ├─ models/Video.js
│  ├─ routes/videos.js
│  └─ uploads/    <-- ملفات الفيديو المحفوظة محلياً
├─ frontend/
│  ├─ package.json
│  ├─ vite.config.js
│  ├─ src/
│  │  ├─ main.jsx
│  │  ├─ App.jsx
│  │  ├─ styles.css
│  │  └─ components/
│  │     ├─ FeedFullScreen.jsx
│  │     ├─ Upload.jsx
│  │     └─ VideoCardFull.jsx
└─ README.md
```

ملاحظات عامة

- اسم التطبيق: **Swing**.
- التخزين المحلي لـ backend تعليمي فقط. في بيئة إنتاج استخدم S3/Cloudinary.
- الواجهة مصممة لتجربة شبيهة بموبايل (full-screen feed، autoplay عند التمرير، swipe up/down للتنقل).

---

backend/package.json

```
{
  "name": "swing-backend",
  "version": "1.0.0",
  "main": "server.js",
  "scripts": {
    "start": "node server.js"
  },
  "dependencies": {
    "cors": "^2.8.5",
    "express": "^4.18.2",
    "mongoose": "^7.0.0",
    "multer": "^1.4.5",
    "uuid": "^9.0.0"
  }
}
```

backend/models/Video.js

```
const mongoose = require('mongoose');

const VideoSchema = new mongoose.Schema({
  filename: String,
  originalname: String,
  title: String,
  description: String,
  likes: { type: Number, default: 0 },
  comments: [
    { text: String, createdAt: { type: Date, default: Date.now } }
  ],
  createdAt: { type: Date, default: Date.now }
});

module.exports = mongoose.model('Video', VideoSchema);
```

backend/routes/videos.js

```
const express = require('express');
const router = express.Router();
const multer = require('multer');
const { v4: uuidv4 } = require('uuid');
const Video = require('../models/Video');
const path = require('path');

const storage = multer.diskStorage({
  destination: (req, file, cb) => cb(null, path.join(__dirname, '..',
'uploads')),
  filename: (req, file, cb) => cb(null, uuidv4() +
path.extname(file.originalname))
});

const upload = multer({ storage, limits: { fileSize: 200 * 1024 * 1024 } });

router.post('/upload', upload.single('video'), async (req, res) => {
  try {
    const { title, description } = req.body;
    const video = new Video({
      filename: req.file.filename,
      originalname: req.file.originalname,
      title,
      description
    });
    await video.save();
    res.json({ success: true, video });
  } catch (err) {
    console.error(err);
    res.status(500).json({ success: false, message: 'Server error' });
  }
});

router.get('/feed', async (req, res) => {
  try {
    const videos = await Video.find().sort({ createdAt: -1 }).limit(100);
    res.json({ success: true, videos });
  } catch (err) {
    res.status(500).json({ success: false });
  }
});

router.post('/:id/like', async (req, res) => {
  try {
    const video = await Video.findById(req.params.id);
    if (!video) return res.status(404).json({ success: false });
    video.likes += 1;
    await video.save();
    res.json({ success: true, likes: video.likes });
  } catch (err) {
    res.status(500).json({ success: false });
  }
});
```

```
router.post('/:id/comment', async (req, res) => {
  try {
    const { text } = req.body;
    const video = await Video.findById(req.params.id);
    if (!video) return res.status(404).json({ success: false });
    video.comments.push({ text });
    await video.save();
    res.json({ success: true, comments: video.comments });
  } catch (err) {
    res.status(500).json({ success: false });
  }
});

module.exports = router;
```

backend/server.js

```
const express = require('express');
const mongoose = require('mongoose');
const cors = require('cors');
const path = require('path');

const videosRouter = require('./routes/videos');

const app = express();
app.use(cors());
app.use(express.json());

app.use('/uploads', express.static(path.join(__dirname, 'uploads')));
app.use('/api/videos', videosRouter);

const MONGO = process.env.MONGO_URI || 'mongodb://localhost:27017/swing-mvp';
mongoose.connect(MONGO).then(() => console.log('mongo connected'))
  .catch(err => console.error(err));

const PORT = process.env.PORT || 4000;
app.listen(PORT, () => console.log('Swing backend running on', PORT));
```

---

frontend/package.json

```
{
  "name": "swing-frontend",
  "version": "1.0.0",
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "preview": "vite preview"
  },
  "dependencies": {
    "react": "^18.2.0",
    "react-dom": "^18.2.0"
  },
  "devDependencies": {
    "vite": "^5.0.0"
  }
}
```

frontend/src/styles.css

```
html,body,#root{height:100%;margin:0}
body{font-family:Inter, system-ui, sans-serif;background:#000;color:#fff}
.app{max-width:720px;margin:0 auto;height:100vh;display:flex;flex-
direction:column}
.topbar{padding:12px;text-align:center;background:#0a0a0a}
.feed{flex:1;overflow:hidden;position:relative}
.video-slide{position:absolute;inset:0;display:flex;align-items:center;justify-
content:center}
.video-meta{position:absolute;right:12px;bottom:24px;z-index:20;text-
align:right}
.controls{position:absolute;left:12px;bottom:80px;display:flex;flex-
direction:column;gap:12px}
.upload-form{padding:12px;background:#111}
```

frontend/src/main.jsx

```
import React from 'react'
import { createRoot } from 'react-dom/client'
import App from './App'
import './styles.css'

createRoot(document.getElementById('root')).render(<App />)
```

frontend/src/App.jsx

```
import React from 'react'
import FeedFullScreen from './components/FeedFullScreen'
import Upload from './components/Upload'

export default function App(){
  return (
    <div className="app">
      <div className="topbar">Swing □ استعرض الفيديوهات</div>
      <FeedFullScreen />
      <Upload />
    </div>
  )
}
```

frontend/src/components/FeedFullScreen.jsx

```
import React, { useEffect, useRef, useState } from 'react'
import VideoCardFull from './VideoCardFull'

export default function FeedFullScreen(){
  const [videos, setVideos] = useState([])
  const [index, setIndex] = useState(0)
  const containerRef = useRef(null)

  useEffect(()=>{ load() }, [])

  const load = async ()=>{
    const res = await fetch('http://localhost:4000/api/videos/feed')
    const j = await res.json()
    if(j.success) setVideos(j.videos)
  }

  // wheel / touch handling for navigation (swipe up/down)
  useEffect(()=>{
    let startY = 0
    const el = containerRef.current
    if(!el) return

    const touchStart = (e)=>{ startY = e.touches ? e.touches[0].clientY :
e.clientY }
    const touchEnd = (e)=>{
      const endY = e.changedTouches ? e.changedTouches[0].clientY : e.clientY
      const dy = endY - startY
      if(Math.abs(dy) > 50){
        if(dy < 0) next() // swipe up -> next
        else prev()
      }
    }

    el.addEventListener('touchstart', touchStart)
    el.addEventListener('touchend', touchEnd)
    el.addEventListener('mousedown', touchStart)
    el.addEventListener('mouseup', touchEnd)

    return ()=>{
      el.removeEventListener('touchstart', touchStart)
      el.removeEventListener('touchend', touchEnd)
      el.removeEventListener('mousedown', touchStart)
      el.removeEventListener('mouseup', touchEnd)
    }
  }, [index, videos])

  const next = ()=> setIndex(i => Math.min(i+1, videos.length-1))
  const prev = ()=> setIndex(i => Math.max(i-1, 0))

  return (
    <div className="feed" ref={containerRef}>
      {videos.map((v,i)=> (
        <div key={v._id} className="video-slide" style={{transform:
`translateY(${(i-index)*100}%)`, transition:'transform 300ms'}}>
          <VideoCardFull video={v} active={i===index} onNext={next} />
        </div>
      ))}
    </div>
  )
}
```

frontend/src/components/VideoCardFull.jsx

```
import React, { useEffect, useRef, useState } from 'react'

export default function VideoCardFull({ video, active }){
  const vidRef = useRef(null)
  const [likes, setLikes] = useState(video.likes || 0)
  const [comments, setComments] = useState(video.comments || [])

  useEffect(()=>{
    const v = vidRef.current
    if(!v) return
    if(active){
      v.play().catch(()=>{})
    } else {
      v.pause()
      v.currentTime = 0
    }
  }, [active])

  const handleLike = async ()=>{
    await fetch(`http://localhost:4000/api/videos/${video._id}/like`, { method:
'POST' })
    setLikes(l => l+1)
  }

  const handleComment = async (text)=>{
    await fetch(`http://localhost:4000/api/videos/${video._id}/comment`, {
method: 'POST', headers: {'Content-Type':'application/json'}, body:
JSON.stringify({ text }) })
    setComments(c => [...c, { text }])
  }

  return (
    <div style={{width:'100%',height:'100%',position:'relative'}}>
      <video ref={vidRef} src={`http://localhost:4000/uploads/${video.filename}
`} style={{width:'100%',height:'100%',objectFit:'cover'}} loop muted
playsInline />

      <div className="video-meta">
        <h3>{video.title}</h3>
        <p>{video.description}</p>
      </div>

      <div className="controls">
        <button onClick={handleLike}>أعجبني ({likes})</button>
        <details style={{background:'#111',padding:8,borderRadius:8}}>
          <summary>تعليقات ({comments.length})</summary>
          <ul>
            {comments.map((c,i)=>(<li key={i}>{c.text}</li>))}
          </ul>
          <form onSubmit={(e)=>{e.preventDefault(); const
t=e.target.elements.c.value; handleComment(t); e.target.reset();}}>
            <input name="c" placeholder="اكتب تعليق..." />
            <button type="submit">إرسال</button>
          </form>
        </details>
      </div>
    </div>
  )
}
```

frontend/src/components/Upload.jsx

```
import React, { useState } from 'react'

export default function Upload(){
  const [file, setFile] = useState(null)
  const [title, setTitle] = useState('')
  const [desc, setDesc] = useState('')

  const submit = async (e)=>{
    e.preventDefault()
    if(!file) return alert('اختر فيديو')
    const fd = new FormData()
    fd.append('video', file)
    fd.append('title', title)
    fd.append('description', desc)
    const res = await fetch('http://localhost:4000/api/videos/upload', {
method:'POST', body: fd })
    const j = await res.json()
    if(j.success) { alert('تم الرفع'); window.location.reload() }
    else alert('فشل')
  }

  return (
    <form className="upload-form" onSubmit={submit}>
      <input type="file" accept="video/*"
onChange={e=>setFile(e.target.files[0])} />
      <input placeholder="عنوان" value={title}
onChange={e=>setTitle(e.target.value)} />
      <input placeholder="وصف" value={desc}
onChange={e=>setDesc(e.target.value)} />
      <button type="submit">رفع إلى Swing</button>
    </form>
  )
}
```

تشغيل محلي

1. شغّل MongoDB.
2. في مجلد `backend`:

3. `npm install`
4. `mkdir uploads`
5. `npm start`
6. في مجلد `frontend`:
7. `npm install`
8. `npm run dev`

خطوات مقترحة لاحقاً

- تحويل الفيديوهات عبر FFmpeg لإنشاء preview / ترميزات أصغر.
- رفع مباشر إلى Cloudinary أو S3 مع signed uploads.
- إضافة مصادقة (JWT) وحسابات مستخدمين، وحفظ منشنات/تاريخ المشاهدة.
- تحسين تجربة الواجهة (preload التالي، lazy loading، إيماءات أكثر، هدر أقل).

أبدأ — (أو تحويل الفيديو تلقائياً Cloudinary، مثلاً: دعم تسجيل دخول، رفع لـ) — إذا تبغى أعدل أي جزء بالتحديد أطبّقها مباشرة في المستند.