```html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>AiDub English Lite (Free)</title>
<style>
  body { font-family: system-ui, Arial, sans-serif; margin: 16px; background:#f5f6fa; color:#111; }
  h1 { font-size:1.4rem; margin-bottom:8px; }
  .card { background:white; border-radius:10px; padding:14px; margin-bottom:12px; box-shadow:0 6px 18px rgba(0,0,0,0.06); }
  label { display:block; margin-top:10px; font-weight:600; }
  input, select, textarea, button { font-size:0.95rem; margin-top:6px; }
  textarea { width:100%; min-height:80px; border-radius:8px; border:1px solid #ccc; padding:6px; resize:vertical; }
  button { padding:6px 12px; border-radius:8px; border:none; background:#2b6ef6; color:white; cursor:pointer; }
  button.secondary { background:#e0e6f5; color:#1b3bb6; }
  video, audio { width:100%; margin-top:6px; border-radius:8px; background:black; }
  .pill { display:inline-block; padding:4px 8px; border-radius:999px; background:#eef2ff; color:#1d3ad6; font-weight:600; margin-left:6px; }
</style>
</head>
<body>

<h1>AiDub English Lite — Free Video Translator & Dub</h1>

<div class="card">
  <label>1) Choose a video</label>
  <input type="file" id="videoFile" accept="video/*">
  <video id="video" controls></video>
</div>

<div class="card">
  <label>2) Transcribe (via microphone)</label>
  <button id="startTrans">Start Transcription</button>
  <button id="stopTrans" class="secondary">Stop</button>
  <span class="pill" id="recIndicator">Idle</span>
  <textarea id="transcript" placeholder="Transcript will appear here..."></textarea>
  <button id="clearTranscript" class="secondary">Clear Transcript</button>
</div>

<div class="card">
  <label>3) Translate to English</label>
  <button id="translateBtn">Translate</button>
  <textarea id="translated" placeholder="Translated text will appear here..."></textarea>
```

```
</div>

<div class="card">
  <label>4) Synthesize Dubbed English Audio</label>
  <select id="voiceSelect"></select>
  <select id="rate">
    <option value="0.9">Rate 0.9</option>
    <option value="1" selected>Rate 1</option>
    <option value="1.1">Rate 1.1</option>
    <option value="1.2">Rate 1.2</option>
  </select>
  <button id="speakBtn">Speak (Preview)</button>
  <button id="startRecord">Start Recording</button>
  <button id="stopRecord" class="secondary">Stop Recording</button>
  <span class="pill" id="recTtsIndicator">Idle</span>
  <audio id="dubAudio" controls></audio>
  <a id="downloadLink" download="dubbed-audio.webm" class="secondary">Download Dubbed Audio</a>
</div>

<script>
const videoFile = document.getElementById('videoFile');
const video = document.getElementById('video');
const startTrans = document.getElementById('startTrans');
const stopTrans = document.getElementById('stopTrans');
const recIndicator = document.getElementById('recIndicator');
const transcriptTA = document.getElementById('transcript');
const clearTranscript = document.getElementById('clearTranscript');

const translateBtn = document.getElementById('translateBtn');
const translatedTA = document.getElementById('translated');

const voiceSelect = document.getElementById('voiceSelect');
const rateSelect = document.getElementById('rate');
const speakBtn = document.getElementById('speakBtn');

const startRecord = document.getElementById('startRecord');
const stopRecord = document.getElementById('stopRecord');
const recTtsIndicator = document.getElementById('recTtsIndicator');
const dubAudio = document.getElementById('dubAudio');
const downloadLink = document.getElementById('downloadLink');

let recognition=null;
let mediaRecorder=null;
let recordedChunks=[];
```

```
// Load video
videoFile.addEventListener('change', e=>{
  const f = e.target.files[0];
  if(f) video.src = URL.createObjectURL(f);
});

// Speech Recognition (via mic)
function createRecognition(){
  const SpeechRecognition = window.SpeechRecognition || window.webkitSpeechRecognition;
  if(!SpeechRecognition){ alert("SpeechRecognition not supported on this browser."); return null; }
  const r = new SpeechRecognition();
  r.lang = 'auto';
  r.interimResults = true;
  r.continuous = true;
  r.onstart = ()=> recIndicator.textContent='Listening...';
  r.onend = ()=> recIndicator.textContent='Idle';
  r.onerror = (ev)=> { console.warn(ev); recIndicator.textContent='Error'; };
  r.onresult = (ev)=>{
    let interim='',finalText='';
    for(let i=ev.resultIndex;i<ev.results.length;i++){
      const res = ev.results[i];
      if(res.isFinal) finalText+=res[0].transcript+' ';
      else interim+=res[0].transcript;
    }
    if(finalText) transcriptTA.value += finalText;
  };
  return r;
}

startTrans.addEventListener('click', ()=>{
  if(!recognition) recognition = createRecognition();
  if(recognition) recognition.start();
});
stopTrans.addEventListener('click', ()=>{ if(recognition) recognition.stop(); });

// Clear transcript
clearTranscript.addEventListener('click', ()=> transcriptTA.value='');

// Translate via LibreTranslate
async function translateToEnglish(text){
  const url='https://libretranslate.de/translate';
  const resp = await fetch(url,{
    method:'POST',
    headers:{'Content-Type':'application/json'},
    body:JSON.stringify({q:text,source:'auto',target:'en',format:'text'})
  });
  const data = await resp.json();
  return data.translatedText;
}

translateBtn.addEventListener('click', async ()=>{
  const text = transcriptTA.value.trim();
  if(!text){ alert('No transcript'); return; }
  translateBtn.textContent='Translating...'; translateBtn.disabled=true;
  try {
    const translated = await translateToEnglish(text);
    translatedTA.value = translated;
  } catch(err){ alert('Translation failed: '+err.message);}
  translateBtn.disabled=false; translateBtn.textContent='Translate';
});

// TTS Voices
function populateVoices(){
  const voices = speechSynthesis.getVoices();
  voiceSelect.innerHTML='';
  voices.forEach(v=>{
    const opt = document.createElement('option');
    opt.value=v.name;
    opt.textContent=v.name+' ('+v.lang+')';
    voiceSelect.appendChild(opt);
  });
}
speechSynthesis.onvoiceschanged = populateVoices;
populateVoices();

// Speak (preview)
speakBtn.addEventListener('click', ()=>{
  const text = translatedTA.value.trim();
  if(!text) return alert('No translated text');
  const utter = new SpeechSynthesisUtterance(text);
  const v = speechSynthesis.getVoices().find(x=>x.name===voiceSelect.value);
  if(v) utter.voice = v;
  utter.rate = parseFloat(rateSelect.value) || 1;
  speechSynthesis.cancel(); speechSynthesis.speak(utter);
});

// Record TTS audio
startRecord.addEventListener('click', async ()=>{
  recordedChunks=[];
  try{
    const stream = await navigator.mediaDevices.getDisplayMedia({video:false,audio:true});
    mediaRecorder = new MediaRecorder(stream);
    mediaRecorder.ondataavailable = e=>{ if(e.data.size) recordedChunks.push(e.data); };
    mediaRecorder.onstop = ()=>{
      const blob = new Blob(recordedChunks,{type:'audio/webm'});
      const url = URL.createObjectURL(blob);
      dubAudio.src=url;
      downloadLink.href=url;
      downloadLink.download='dubbed-audio.webm';
      recTtsIndicator.textContent='Recorded';
      stream.getTracks().forEach(t=>t.stop());
    };
    mediaRecorder.start();
    recTtsIndicator.textContent='Recording... (press Speak now)';
    startRecord.disabled=true; stopRecord.disabled=false;
  }catch(err){
    alert('Recording failed. Chrome desktop with "Share tab audio" is recommended.');
    console.error(err);
  }
});

stopRecord.addEventListener('click', ()=>{
  if(mediaRecorder && mediaRecorder.state==='recording') mediaRecorder.stop();
  startRecord.disabled=false; stopRecord.disabled=true;
  recTtsIndicator.textContent='Processing...';
});
</script>

</body>
</html>
```