「TinyLingo · 微世界英语」(Next.js + TS + Tailwind)

你是一名全栈工程师，请实现一个名为 TinyLingo – 微世界英语 的 Web 应用。技术栈：Next.js (App Router) + TypeScript + Tailwind CSS；前端画布用 react-konva 或 Fabric.js；后端用 Next.js Route Handlers（Node）+ 一个 Python 微服务 负责去背景（rembg + PIL）。要求像素级还原如下信息架构与交互。

0. 关键产品逻辑（已改）

成本控制逻辑：  
上传一张场景图后，只做基础去背景 + 前端像素级分割。  
用户在前端预览/选择要保存的物品后，才触发 AI 识别/命名/TTS/风格统一（对被选中的连通域进行处理）。

去背景：使用 remove.bg（可选）或 Python rembg 服务（默认）。

分割：前端 Canvas API + BFS，对去背景图像按像素透明度识别连通区域，每个区域生成独立 Canvas 与 PNG 贴纸预览。

等距风格与英语学习随后置于“AI识别”阶段处理，仅对用户选中的区域执行。

1. 站点结构与页面

顶部导航（全站一致）

左：Logo TinyLingo

中：Home | My stickers | My Worlds | Explore | Create World

右：用户头像+ “Hi, {username}”

页脚（全站一致）

深色底三列：  
左：About Us / Projects / Contacts  
中：Privacy Policy / Terms of Use  
右：Facebook / Instagram / LinkedIn / Back to Home

右上有 Inquire → 按钮

1) Home

巨幅标题：BUILD IT, LEARN IT

副标：Where little words shape great worlds

世界卡片网格（封面、名称、Words 数、收藏/喜欢）

主按钮：Create World Now!

2) Explore

标题：EXPLORE + 搜索框（World/Sticker 切换）

瀑布流卡片（封面、名称、作者、计数）

“View More”/分页

3) My Stickers

标题：MY STICKERS

Sorted（按标签分组）/ Unsorted

批量选择、打标签、下载 ZIP

4) My Worlds

标题：MY WORLDS，副标同 Home

卡片列表（含 Create New World 特殊卡片）

搜索/排序/分享

5) Create World（核心页面）

左：等距 2.5D 房间/背板（画布）

右：侧栏 Tabs：Stickers | Background | Uploads

Uploads 流程（本项目新增/重写）

上传图片（JPG/PNG/WebP ≤10MB）

调用后端：先去背景（/bg/remove），返回透明 PNG

前端分割：对透明 PNG 用 Canvas + BFS 找连通区域（基于 alpha>阈值）

在侧栏网格显示每个候选贴纸（小缩略图+面积/像素计数）

用户勾选要保存的物品（也可框选/全选/反选）

点击 “识别并保存所选” → 才调用 AI：

命名（英语名）、音标/释义（可简化）、TTS 生成

风格统一（2.5D 等距 45°、干净背景、统一光照）

仅对被选中区域进行

批量生成 透明 PNG 贴纸与可选背板 → 保存到微库（同时可 ZIP 下载）

画布交互

导入背板（作为底层）

拖拽贴纸、缩放、旋转、层级、吸附、撤销/重做

移动贴纸时自动播放英语发音（拖入或移动≥12px触发；对象 1.2s 冷却）

2. 数据模型（TS 接口）

// 用户

type User = {

id: string; name: string; email: string; avatarUrl?: string;

};

// 场景（一次上传处理记录）

type Scene = {

id: string;

ownerId: string;

originalImageUrl: string;

bgRemovedImageUrl?: string; // 去背景后的透明 PNG

candidates: Candidate[]; // 前端分割产生的候选区域（仅元数据）

createdAt: string;

status: 'uploaded' | 'bgRemoved' | 'aiProcessing' | 'ready' | 'failed';

};

type Candidate = {

id: string;

bbox: { x: number; y: number; w: number; h: number };

area: number; // 像素数量

previewUrl?: string; // 用于侧栏缩略图（dataURL）

maskRle?: string; // 可选：行程编码的掩膜（节省体积）

selected?: boolean;

};

// 贴纸

type Sticker = {

id: string; ownerId: string;

name: string; // 'Donut'

ipa?: string;

ttsUrl?: string;

category?: 'Food'|'Pet'|'Furniture'|'Other';

imageUrl: string; // 透明 PNG

width: number; height: number;

sourceSceneId?: string; sourceCandidateId?: string;

createdAt: string; tags: string[];

};

// 背板

type Board = {

id: string; ownerId: string; name: string;

imageUrl: string; sourceSceneId?: string; createdAt: string;

};

// 世界与画布节点

type World = {

id: string; ownerId: string; name: string;

boardId?: string; nodes: WorldNode[];

wordCount: number; likes: number; favorites: number;

coverUrl?: string; createdAt: string; updatedAt: string;

};

type WorldNode = {

id: string; stickerId: string;

x: number; y: number; scale: number; rotation: number; zIndex: number;

lastSpokenAt?: string;

};

3. 后端 API 设计

后端采用 Next.js Route Handlers；另起 Python 微服务（FastAPI）提供 /bg/remove。

Node（Next.js）API

POST /api/scene  
body: { originalImage: multipart } → 返回 { sceneId, originalImageUrl }

POST /api/bg/remove  
body: { sceneId, imageUrl } → 反向代理到 Python 服务，拿到 bgRemovedImageUrl

POST /api/scene/:id/candidates  
body: { candidates: Candidate[] } // 前端分割完成后把元数据回传（或省略，纯前端存）

POST /api/ai/recognize  
body: { sceneId, candidateIds: string[] }  
→ 对选中的候选进行：命名/风格统一/生成 PNG/可选背板/TTS  
→ 返回 { stickers: Sticker[], board?: Board }

POST /api/library/batch-save  
body: { stickers: Sticker[], board?: Board } → 持久化并返回 ID

GET /api/stickers?query=...、GET /api/worlds、POST /api/worlds、PATCH /api/worlds/:id

POST /api/zip/:sceneId → 打包所选贴纸下载

Python（FastAPI）去背景服务

依赖：from PIL import Image，import rembg

POST /bg/remove  
入参：上传二进制或远程 URL  
处理：rembg.remove → 透明 PNG  
返回：文件 URL（或直接字节流）

若需用 remove.bg 官方 API，可在 Node 侧提供 POST /api/bg/remove，用 fetch('https://api.remove.bg/v1.0/removebg') 代理；默认优先 rembg。

4. 前端 Canvas 分割（BFS 连通域）

思路

对去背景后的 PNG 绘制到隐藏 Canvas，取 ImageData。

以 alpha > α阈值(如 16) 作为“前景像素”，其余视为背景。

用 BFS/队列遍历，找出所有连通域（4 邻域或 8 邻域任选，建议 8 邻域）。

每个连通域记录 minX/minY/maxX/maxY/area。

为每个连通域新建独立 Canvas，拷贝像素（非该域像素 alpha 置 0），导出 dataURL 作为候选贴纸预览。

过滤太小的区域（如 area < 300 px）避免噪点；提供“合并小岛”开关（将邻近小区域并入最大邻域）。

TypeScript 伪代码（放在 useSegmentation.ts）

type Region = { id: number; bbox: {x:number;y:number;w:number;h:number}; area: number; mask: Uint8Array };

export function segmentByBFS(source: HTMLImageElement, alphaThresh = 16): Region[] {

const w = source.width, h = source.height;

const srcCanvas = document.createElement('canvas');

srcCanvas.width = w; srcCanvas.height = h;

const ctx = srcCanvas.getContext('2d')!;

ctx.drawImage(source, 0, 0);

const { data } = ctx.getImageData(0, 0, w, h); // RGBA

const visited = new Uint8Array(w \* h);

const regions: Region[] = [];

const inb = (x:number,y:number)=> x>=0 && y>=0 && x<w && y<h;

const idx = (x:number,y:number)=> (y\*w+x);

const dirs = [[1,0],[-1,0],[0,1],[0,-1],[1,1],[1,-1],[-1,1],[-1,-1]];

for (let y=0;y<h;y++){

for (let x=0;x<w;x++){

const i = idx(x,y);

const a = data[i\*4+3];

if (visited[i] || a<=alphaThresh) continue;

// BFS

const q = [[x,y]];

visited[i] = 1;

let minX=x,maxX=x,minY=y,maxY=y, area=0;

const mask = new Uint8Array(w\*h); // 1 = belongs to region

while(q.length){

const [cx,cy] = q.shift()!;

const ci = idx(cx,cy);

mask[ci] = 1; area++;

minX=Math.min(minX,cx); maxX=Math.max(maxX,cx);

minY=Math.min(minY,cy); maxY=Math.max(maxY,cy);

for (const [dx,dy] of dirs){

const nx=cx+dx, ny=cy+dy; if(!inb(nx,ny)) continue;

const ni=idx(nx,ny);

if(visited[ni]) continue;

const na = data[ni\*4+3];

if(na>alphaThresh){

visited[ni]=1; q.push([nx,ny]);

}

}

}

const region: Region = {

id: regions.length+1,

bbox: { x:minX, y:minY, w:maxX-minX+1, h:maxY-minY+1 },

area, mask

};

regions.push(region);

}

}

return regions.sort((a,b)=>b.area-a.area);

}

// 将 region 导出为独立贴纸 Canvas

export function regionToCanvas(region: Region, srcImage: HTMLImageElement): HTMLCanvasElement {

const {x,y,w,h} = region.bbox;

const canvas = document.createElement('canvas'); canvas.width=w; canvas.height=h;

const ctx = canvas.getContext('2d')!;

// 把原图对应区域绘制过来，再用 mask 抠掉

const tmp = document.createElement('canvas'); tmp.width = srcImage.width; tmp.height = srcImage.height;

const tctx = tmp.getContext('2d')!; tctx.drawImage(srcImage,0,0);

const imgData = tctx.getImageData(x,y,w,h); // 只取 bbox

const d = imgData.data; const W = srcImage.width;

// 注意：region.mask 是全图尺寸，这里对应偏移

for(let j=0;j<h;j++){

for(let i=0;i<w;i++){

const gi = ( (y+j)\*W + (x+i) );

if (region.mask[gi] !== 1) {

d[(j\*w+i)\*4+3] = 0; // 非连通域像素 alpha 清零

}

}

}

ctx.putImageData(imgData,0,0);

return canvas;

}

侧栏展示时，直接 canvas.toDataURL('image/png') 作为候选缩略图。

5. 交互与流程（Create World 详细）

用户打开 Create World → 右侧 Uploads

选择图片 → POST /api/scene（存原图）

去背景：前端调用 POST /api/bg/remove → 得到透明 PNG URL（或字节）

前端加载透明 PNG → segmentByBFS → 生成 候选列表（可在内存，不必先存后端）

侧栏显示每个候选缩略图（面积、边框、复选框）

过滤面积过小的候选；支持“最小面积阈值”滑杆

框选/多选/全选/反选

用户点击 “识别并保存所选”：

仅上传被选中的候选（其 bbox PNG 或 RLE 掩膜）到后端

POST /api/ai/recognize →

识别英文名、生成 TTS（或延迟生成）

2.5D 等距 45° 风格统一（若需调用第三方/自研）

输出透明 PNG（自动裁剪并留 8–16px 内边距，统一光照&阴影）

后端返回 stickers[] 与可选 board

前端弹出“保存到微库”确认 → POST /api/library/batch-save

用户可一键 导入背板到画布；将贴纸拖入、移动时自动 TTS 播放

自动生成的英语名计入世界的 wordCount（同名去重）

6. 英语发音（TTS）

新贴纸创建时后台生成 TTS（mp3/wav），字段 sticker.ttsUrl；

前端拖入或移动≥12px触发播放；同对象 1.2s 冷却；世界内并行最多 2 路。

7. 等距风格规范（AI 阶段硬性）

艺术风格：2.5D 等距45°，无正面视角；

设计特征：可平铺、平面矢量感、分层有深度、干净背景、紧凑“场景贴纸”美学；

光照统一自左上；阴影与描边一致；

输出：透明 PNG（可选 SVG）。

8. 目录与关键组件（建议）

app/

(routes)/

page.tsx // Home

explore/page.tsx

my-stickers/page.tsx

my-worlds/page.tsx

create-world/page.tsx // 核心：上传、分割、选择、识别、画布

api/

scene/route.ts // POST 创建scene

bg/remove/route.ts // 代理到 Python 或 remove.bg

ai/recognize/route.ts // 仅处理选中的候选

library/batch-save/route.ts

worlds/route.ts

worlds/[id]/route.ts

components/

Nav.tsx Footer.tsx CardWorld.tsx CardSticker.tsx

CanvasEditor/Editor.tsx // react-konva/fabric 实现

Uploads/UploadPanel.tsx // 去背景 + 分割 + 选择

Uploads/CandidateItem.tsx

useSegmentation.ts // BFS 算法（见上）

useTTS.ts // 播放与节流

lib/

types.ts api.ts

9. Python 去背景微服务（FastAPI 示例）

# main.py

from fastapi import FastAPI, UploadFile, File

from fastapi.responses import Response

from PIL import Image

import io

import rembg

app = FastAPI()

@app.post("/bg/remove")

async def remove\_bg(file: UploadFile = File(...)):

content = await file.read()

result = rembg.remove(content) # bytes in -> bytes out (PNG with alpha)

return Response(content=result, media\_type="image/png")

Next.js 侧通过 form-data 将图片转发到该服务；或直接接收 URL，Node 拉流→转发。

10. 验收用例（必须通过）

上传一张含多个不重叠物体、背景清晰的图 → 去背景成功；前端 BFS 自动生成 ≥2 个候选贴纸预览。

勾选 2 个候选，点“识别并保存所选” → 仅这 2 个被命名+TTS+风格统一，并保存到微库与 ZIP 导出。

在 Create World 导入背板，拖入贴纸，移动时自动发音（1.2s 冷却）。

所有输出贴纸满足 2.5D 等距 45° 风格一致性与透明 PNG 规范。

11. 额外细节

去背景失败或候选<2：弹出指引（请上传“多个不重叠物体、背景清晰分隔”的图片）。

分割参数可调（α阈值、最小面积、8 邻域开关）。

可选“合并邻近小区域”功能。

安全：上传大小/类型校验；用户鉴权（JWT + HttpOnly Cookie）。

无障碍：键盘微调、ARIA、可读名称（学单词）。

请按以上规范搭建项目骨架、实现页面与核心流程。重点完成：

Python rembg 去背景接口；

前端 Canvas+BFS 分割与候选导出；

“先选择、后识别”的成本控制流程；

创建世界页面的画布交互与自动发音。

完成后请提供：运行说明、环境变量示例、主要代码位置说明与演示数据。