# Rathbone PWA – installable app + Node.js backend

A production‑ready starter you can run locally or deploy. It scrapes the Rathbone Dining Hall menu, lets users review/rate/upload photos (including custom food combos), and tracks trends to recommend options. Built as a **Node.js/Express + MongoDB** API and a **React + Vite PWA** front end.

## Project structure

rathbone-pwa/  
├─ server/ # Node.js backend (Express)  
│ ├─ src/  
│ │ ├─ index.ts # server entry (ts-node-dev)  
│ │ ├─ env.ts # env loader & zod schema  
│ │ ├─ app.ts # express app  
│ │ ├─ routes/  
│ │ │ ├─ menu.ts # GET /api/menu  
│ │ │ ├─ reviews.ts # CRUD reviews + photos  
│ │ │ ├─ combos.ts # CRUD combos  
│ │ │ └─ recs.ts # GET /api/recommendations  
│ │ ├─ models/  
│ │ │ ├─ MenuItem.ts # menu item schema  
│ │ │ ├─ Review.ts # review schema  
│ │ │ ├─ Combo.ts # combo schema  
│ │ │ └─ User.ts # (anonymous device or email)  
│ │ ├─ scraper/  
│ │ │ ├─ rathbone.ts # Playwright powered scraper  
│ │ │ └─ selectors.ts # centralize CSS/xpath selectors  
│ │ ├─ services/  
│ │ │ ├─ trends.ts # rolling/decayed averages  
│ │ │ └─ storage.ts # local fs or S3 uploads  
│ │ ├─ middlewares/  
│ │ │ ├─ error.ts  
│ │ │ └─ rateLimit.ts  
│ │ ├─ utils/  
│ │ │ ├─ id.ts  
│ │ │ └─ time.ts  
│ │ └─ jobs/  
│ │ └─ nightly.ts # recompute trends daily (node-cron)  
│ ├─ package.json  
│ └─ tsconfig.json  
├─ web/ # React + Vite PWA front end  
│ ├─ index.html  
│ ├─ vite.config.ts  
│ ├─ package.json  
│ ├─ public/  
│ │ ├─ manifest.webmanifest  
│ │ └─ icons/\* # PWA icons (generated via realfavicongenerator)  
│ └─ src/  
│ ├─ main.tsx  
│ ├─ App.tsx  
│ ├─ api/client.ts # fetch helper & offline queue  
│ ├─ components/  
│ │ ├─ ItemCard.tsx  
│ │ ├─ ReviewModal.tsx  
│ │ ├─ ComboCard.tsx  
│ │ ├─ RatingStars.tsx  
│ │ ├─ TrendsChart.tsx  
│ │ └─ Toast.tsx  
│ ├─ pages/  
│ │ ├─ Home.tsx  
│ │ ├─ Combos.tsx  
│ │ └─ Trends.tsx  
│ ├─ hooks/useDeviceId.ts  
│ ├─ styles.css  
│ └─ sw.ts # custom service worker (workbox via plugin)  
├─ .env.example  
└─ README.md

## 1) Backend (Node.js/Express)

### server/package.json

{  
 "name": "rathbone-server",  
 "version": "1.0.0",  
 "type": "module",  
 "scripts": {  
 "dev": "ts-node-dev --respawn --transpileOnly src/index.ts",  
 "start": "node dist/index.js",  
 "build": "tsc"  
 },  
 "dependencies": {  
 "cors": "^2.8.5",  
 "dotenv": "^16.4.5",  
 "express": "^4.19.2",  
 "express-rate-limit": "^7.3.1",  
 "mongoose": "^8.6.0",  
 "morgan": "^1.10.0",  
 "multer": "^1.4.5-lts.1",  
 "nanoid": "^5.0.7",  
 "node-cron": "^3.0.3",  
 "pino": "^9.3.2",  
 "pino-pretty": "^11.2.2",  
 "playwright": "^1.47.2",  
 "zod": "^3.23.8"  
 },  
 "devDependencies": {  
 "@types/express": "^4.17.21",  
 "@types/morgan": "^1.9.9",  
 "@types/node": "^20.11.30",  
 "ts-node-dev": "^2.0.0",  
 "typescript": "^5.4.5"  
 }  
}

### server/src/env.ts

import { z } from 'zod';  
import \* as dotenv from 'dotenv';  
  
dotenv.config();  
  
const EnvSchema = z.object({  
 PORT: z.string().default('4000'),  
 MONGODB\_URI: z.string().url(),  
 RATHBONE\_URL: z.string().url().default('https://lehigh.sodexomyway.com/en-us/locations/rathbone-dining-hall'),  
 UPLOAD\_DIR: z.string().default('uploads'),  
 STORAGE: z.enum(['fs', 's3']).default('fs'),  
 S3\_BUCKET: z.string().optional(),  
 S3\_REGION: z.string().optional(),  
 S3\_ACCESS\_KEY\_ID: z.string().optional(),  
 S3\_SECRET\_ACCESS\_KEY: z.string().optional()  
});  
  
export const env = EnvSchema.parse(process.env);

### server/src/index.ts

import { env } from './env';  
import { app } from './app';  
import mongoose from 'mongoose';  
import cron from 'node-cron';  
import { recomputeTrends } from './jobs/nightly';  
  
(async () => {  
 await mongoose.connect(env.MONGODB\_URI);  
 const server = app.listen(Number(env.PORT), () => {  
 console.log(`[api] listening on http://localhost:${env.PORT}`);  
 });  
  
 // Run nightly at 03:15 local time  
 cron.schedule('15 3 \* \* \*', async () => {  
 try { await recomputeTrends(); } catch (e) { console.error(e); }  
 });  
  
 // graceful  
 process.on('SIGINT', () => { server.close(() => process.exit(0)); });  
})();

### server/src/app.ts

import express from 'express';  
import cors from 'cors';  
import morgan from 'morgan';  
import path from 'node:path';  
import { env } from './env';  
import menuRouter from './routes/menu';  
import reviewRouter from './routes/reviews';  
import comboRouter from './routes/combos';  
import recsRouter from './routes/recs';  
import { rateLimiter } from './middlewares/rateLimit';  
  
export const app = express();  
app.use(cors());  
app.use(express.json({ limit: '2mb' }));  
app.use(morgan('dev'));  
app.use('/uploads', express.static(path.resolve(env.UPLOAD\_DIR)));  
app.use(rateLimiter);  
  
app.get('/health', (\_req, res) => res.json({ ok: true }));  
app.use('/api/menu', menuRouter);  
app.use('/api/reviews', reviewRouter);  
app.use('/api/combos', comboRouter);  
app.use('/api/recommendations', recsRouter);  
  
// 404  
app.use((\_req, res) => res.status(404).json({ error: 'Not found' }));

### Models

#### server/src/models/MenuItem.ts

import { Schema, model } from 'mongoose';  
  
const MenuItemSchema = new Schema({  
 key: { type: String, index: true, unique: true }, // station+name+date slug  
 name: String,  
 station: String,  
 mealPeriod: String, // Breakfast/Lunch/Dinner  
 date: { type: String, index: true }, // YYYY-MM-DD  
 calories: Number,  
 allergens: [String],  
 dietary: [String], // Vegan/Vegetarian/etc.  
 raw: Schema.Types.Mixed, // keep original scrape payload  
 createdAt: { type: Date, default: Date.now }  
});  
  
export default model('MenuItem', MenuItemSchema);

#### server/src/models/Review.ts

import { Schema, model } from 'mongoose';  
  
const ReviewSchema = new Schema({  
 itemKey: { type: String, index: true }, // links to MenuItem.key, or comboId  
 isCombo: { type: Boolean, default: false },  
 rating: { type: Number, min: 1, max: 5, required: true },  
 text: { type: String, default: '' },  
 images: [String], // URLs  
 deviceId: { type: String, index: true },  
 userId: { type: String, default: null },  
 createdAt: { type: Date, default: Date.now }  
});  
  
export default model('Review', ReviewSchema);

#### server/src/models/Combo.ts

import { Schema, model } from 'mongoose';  
  
const ComboSchema = new Schema({  
 comboId: { type: String, unique: true },  
 title: String,  
 description: String,  
 items: [String], // array of MenuItem.key or free text if ad-hoc  
 date: { type: String, index: true },  
 createdBy: String, // deviceId or userId  
 createdAt: { type: Date, default: Date.now }  
});  
export default model('Combo', ComboSchema);

### Scraper

#### server/src/scraper/selectors.ts

/\*\* Centralize selectors so you can tweak without touching scraper logic. \*/  
export const selectors = {  
 // Section with the daily menu  
 menuRootHeading: 'h2:has-text("Rathbone Dining Hall Menu")',  
 // Stations usually appear as headings before item lists  
 stationHeadings: 'h3, h4',  
 // Item rows under a station  
 itemRows: 'li, .menu-item, .list-unstyled li',  
 // Calories links typically clickable for nutrition  
 calories: 'a[aria-label\*="cal"], a[href\*="nutrition"], .calories',  
 // Allergen/diet icons often have alt text  
 iconImgs: 'img[alt]'  
};

#### server/src/scraper/rathbone.ts

import { chromium } from 'playwright';  
import { selectors } from './selectors';  
import slugify from '../utils/id';  
import { toISODate } from '../utils/time';  
  
type Item = {  
 key: string; name: string; station: string; mealPeriod?: string; date: string;  
 calories?: number; allergens: string[]; dietary: string[]; raw: any;  
};  
  
export async function fetchMenu(date = toISODate(new Date())): Promise<Item[]> {  
 const browser = await chromium.launch({ args: ['--no-sandbox'] });  
 const page = await browser.newPage({ userAgent: 'Mozilla/5.0 RathbonePWA/1.0' });  
 await page.goto(process.env.RATHBONE\_URL!, { waitUntil: 'domcontentloaded' });  
  
 // If a date picker exists, try to navigate to the requested date by clicking it  
 try {  
 // quick attempt: look for the date number and click it  
 const dayNum = new Date(date).getDate().toString();  
 const dateBtn = page.locator('button, [role="button"]').filter({ hasText: dayNum }).first();  
 if (await dateBtn.count()) { await dateBtn.click({ trial: false }).catch(() => {}); }  
 } catch {}  
  
 // Wait for menu region to render  
 await page.waitForTimeout(1500);  
  
 // Scrape  
 const items = await page.evaluate((sel, dateStr) => {  
 const root = document.querySelector(sel.menuRootHeading)?.parentElement || document.body;  
 const out: any[] = [];  
  
 function text(el: Element | null | undefined) { return (el?.textContent || '').trim(); }  
  
 // Find station blocks heuristically: heading + following list  
 const stations = Array.from(root.querySelectorAll(sel.stationHeadings)) as HTMLElement[];  
 stations.forEach(st => {  
 const stationName = text(st);  
 let cursor: Element | null = st.nextElementSibling;  
 const seen = new Set<string>();  
 for (let hops = 0; cursor && hops < 10; hops++) {  
 const rows = cursor.querySelectorAll(sel.itemRows);  
 rows.forEach((row: Element) => {  
 const name = text(row).replace(/\s+\d+\s\*cal.\*/i, '').replace(/\s{2,}/g, ' ').trim();  
 if (!name || seen.has(name)) return;  
 seen.add(name);  
 const calEl = row.querySelector(sel.calories);  
 const cal = calEl ? parseInt((calEl as HTMLElement).innerText.replace(/\D/g, ''), 10) : undefined;  
 const icons = Array.from(row.querySelectorAll(sel.iconImgs)) as HTMLImageElement[];  
 const allergDiet = icons.map(i => (i.alt || '').trim()).filter(Boolean);  
 const allergens: string[] = [];  
 const dietary: string[] = [];  
 allergDiet.forEach(a => {  
 if (/vegan|vegetarian|mindful|plant/i.test(a)) dietary.push(a);  
 else allergens.push(a);  
 });  
 out.push({  
 key: `${stationName}:${name}:${dateStr}`.toLowerCase(),  
 name,  
 station: stationName,  
 date: dateStr,  
 calories: isFinite(cal as number) ? cal : undefined,  
 allergens,  
 dietary,  
 raw: { stationName, html: row.outerHTML }  
 });  
 });  
 cursor = cursor.nextElementSibling;  
 }  
 });  
 return out;  
 }, selectors, date);  
  
 await browser.close();  
  
 // normalize keys  
 return items.map(i => ({ ...i, key: slugify(i.key) }));  
}

#### server/src/utils/id.ts

export default function slugify(s: string) {  
 return s  
 .toLowerCase()  
 .replace(/[^a-z0-9]+/g, '-')  
 .replace(/(^-|-$)+/g, '');  
}

#### server/src/utils/time.ts

export function toISODate(d: Date) {  
 const tz = new Date(d.getTime() - d.getTimezoneOffset() \* 60000);  
 return tz.toISOString().slice(0, 10); // YYYY-MM-DD  
}

### Routes

#### server/src/routes/menu.ts

import { Router } from 'express';  
import MenuItem from '../models/MenuItem';  
import { fetchMenu } from '../scraper/rathbone';  
import { toISODate } from '../utils/time';  
  
const router = Router();  
  
router.get('/', async (req, res) => {  
 const date = (req.query.date as string) || toISODate(new Date());  
 const existing = await MenuItem.find({ date }).lean();  
 if (existing.length) return res.json(existing);  
  
 const items = await fetchMenu(date);  
 if (!items.length) return res.status(503).json({ error: 'No menu found for date' });  
  
 await MenuItem.bulkWrite(items.map((i) => ({  
 updateOne: { filter: { key: i.key }, update: { $set: i }, upsert: true }  
 })));  
 res.json(items);  
});  
  
export default router;

#### server/src/routes/reviews.ts

import { Router } from 'express';  
import multer from 'multer';  
import path from 'node:path';  
import { nanoid } from 'nanoid';  
import Review from '../models/Review';  
import MenuItem from '../models/MenuItem';  
  
const upload = multer({ dest: process.env.UPLOAD\_DIR || 'uploads' });  
const router = Router();  
  
router.get('/', async (req, res) => {  
 const { itemKey } = req.query;  
 const q: any = {};  
 if (itemKey) q.itemKey = itemKey;  
 const reviews = await Review.find(q).sort({ createdAt: -1 }).lean();  
 res.json(reviews);  
});  
  
router.post('/', upload.array('images', 4), async (req, res) => {  
 const { itemKey, rating, text, isCombo } = req.body;  
 if (!itemKey || !rating) return res.status(400).json({ error: 'itemKey and rating required' });  
 // Ensure target exists (either menu item or combo id)  
 if (!isCombo) {  
 const exists = await MenuItem.findOne({ key: itemKey });  
 if (!exists) return res.status(404).json({ error: 'Menu item not found' });  
 }  
 const imageUrls = (req.files as any[] | undefined)?.map((f) => `/${f.destination}/${f.filename}`) || [];  
 const review = await Review.create({  
 itemKey,  
 isCombo: isCombo === 'true',  
 rating: Number(rating),  
 text: text || '',  
 images: imageUrls,  
 deviceId: req.header('x-device-id') || nanoid()  
 });  
 res.status(201).json(review);  
});  
  
export default router;

#### server/src/routes/combos.ts

import { Router } from 'express';  
import { nanoid } from 'nanoid';  
import Combo from '../models/Combo';  
  
const router = Router();  
  
router.get('/', async (\_req, res) => {  
 const combos = await Combo.find().sort({ createdAt: -1 }).lean();  
 res.json(combos);  
});  
  
router.post('/', async (req, res) => {  
 const id = nanoid();  
 const { title, description, items, date } = req.body;  
 if (!title) return res.status(400).json({ error: 'title required' });  
 const combo = await Combo.create({ comboId: id, title, description, items: items || [], date });  
 res.status(201).json(combo);  
});  
  
export default router;

#### server/src/services/trends.ts

import Review from '../models/Review';  
  
export async function trendingScores({ sinceDays = 21 } = {}) {  
 const since = new Date(Date.now() - sinceDays \* 864e5);  
 const reviews = await Review.aggregate([  
 { $match: { createdAt: { $gte: since }, isCombo: false } },  
 { $group: { \_id: '$itemKey', count: { $sum: 1 }, avg: { $avg: '$rating' }, last: { $max: '$createdAt' } } },  
 { $sort: { last: -1 } }  
 ]);  
 // exponential time decay weight: half-life 10 days  
 const lambda = Math.log(2) / 10;  
 const now = Date.now();  
 return reviews.map(r => {  
 const ageDays = (now - new Date(r.last).getTime()) / 864e5;  
 const weight = Math.exp(-lambda \* ageDays);  
 const score = (r.avg \* weight) + Math.log(1 + r.count);  
 return { itemKey: r.\_id, score, avg: r.avg, count: r.count, last: r.last };  
 }).sort((a, b) => b.score - a.score);  
}

#### server/src/routes/recs.ts

import { Router } from 'express';  
import MenuItem from '../models/MenuItem';  
import Review from '../models/Review';  
import { trendingScores } from '../services/trends';  
  
const router = Router();  
  
router.get('/', async (req, res) => {  
 const { date } = req.query as { date?: string };  
 const today = date || new Date().toISOString().slice(0, 10);  
 const [menu, scores] = await Promise.all([  
 MenuItem.find({ date: today }).lean(),  
 trendingScores({ sinceDays: 21 })  
 ]);  
 const scoreMap = new Map(scores.map(s => [s.itemKey, s]));  
 const ranked = menu  
 .map(m => ({ ...m, trend: scoreMap.get(m.key) }))  
 .sort((a, b) => (b.trend?.score || 0) - (a.trend?.score || 0));  
 res.json(ranked);  
});  
  
export default router;

#### server/src/middlewares/rateLimit.ts

import rateLimit from 'express-rate-limit';  
  
export const rateLimiter = rateLimit({  
 windowMs: 60\_000,  
 limit: 120,  
 standardHeaders: true,  
 legacyHeaders: false  
});

#### server/src/jobs/nightly.ts

import { trendingScores } from '../services/trends';  
  
export async function recomputeTrends() {  
 // warm cache by calling trendingScores (stored in-memory if you add a cache layer)  
 await trendingScores();  
}

## 2) Frontend (React + Vite PWA)

### web/package.json

{  
 "name": "rathbone-web",  
 "version": "1.0.0",  
 "private": true,  
 "type": "module",  
 "scripts": {  
 "dev": "vite",  
 "build": "vite build",  
 "preview": "vite preview"  
 },  
 "dependencies": {  
 "react": "^18.3.1",  
 "react-dom": "^18.3.1",  
 "recharts": "^2.12.7"  
 },  
 "devDependencies": {  
 "@vitejs/plugin-react": "^4.3.1",  
 "vite": "^5.4.6",  
 "vite-plugin-pwa": "^0.20.5",  
 "typescript": "^5.4.5"  
 }  
}

### web/vite.config.ts

import { defineConfig } from 'vite';  
import react from '@vitejs/plugin-react';  
import { VitePWA } from 'vite-plugin-pwa';  
  
export default defineConfig({  
 plugins: [  
 react(),  
 VitePWA({  
 registerType: 'autoUpdate',  
 includeAssets: ['favicon.svg'],  
 manifest: {  
 name: 'Rathbone – What’s Good Today?',  
 short\_name: 'Rathbone',  
 start\_url: '/',  
 display: 'standalone',  
 background\_color: '#0b132b',  
 theme\_color: '#2ec4b6',  
 icons: [  
 { src: '/icons/pwa-192.png', sizes: '192x192', type: 'image/png' },  
 { src: '/icons/pwa-512.png', sizes: '512x512', type: 'image/png' },  
 { src: '/icons/maskable.png', sizes: '512x512', type: 'image/png', purpose: 'maskable' }  
 ]  
 },  
 workbox: {  
 globPatterns: ['\*\*/\*.{js,css,html,ico,png,svg}'],  
 runtimeCaching: [  
 { urlPattern: /\/api\/menu.\*/, handler: 'NetworkFirst', options: { cacheName: 'menu' } },  
 { urlPattern: /\/api\/reviews.\*/, handler: 'NetworkOnly' },  
 { urlPattern: /\/uploads\//, handler: 'CacheFirst', options: { cacheName: 'uploads' } }  
 ]  
 }  
 })  
 ],  
 server: { port: 5173, proxy: { '/api': 'http://localhost:4000', '/uploads': 'http://localhost:4000' } }  
});

### web/public/manifest.webmanifest

{  
 "name": "Rathbone – What’s Good Today?",  
 "short\_name": "Rathbone",  
 "start\_url": "/",  
 "display": "standalone",  
 "background\_color": "#0b132b",  
 "theme\_color": "#2ec4b6",  
 "icons": [  
 { "src": "/icons/pwa-192.png", "sizes": "192x192", "type": "image/png" },  
 { "src": "/icons/pwa-512.png", "sizes": "512x512", "type": "image/png" },  
 { "src": "/icons/maskable.png", "sizes": "512x512", "type": "image/png", "purpose": "maskable" }  
 ]  
}

### web/src/api/client.ts

const BASE = '';  
  
export async function api<T>(path: string, opts: RequestInit = {}): Promise<T> {  
 const deviceId = localStorage.getItem('deviceId') || crypto.randomUUID();  
 localStorage.setItem('deviceId', deviceId);  
 const res = await fetch(BASE + path, { ...opts, headers: { 'x-device-id': deviceId, ...(opts.headers || {}) } });  
 if (!res.ok) throw new Error(await res.text());  
 return res.json();  
}  
  
export async function getMenu(date?: string) {  
 const q = date ? `?date=${date}` : '';  
 return api<any[]>(`/api/menu${q}`);  
}  
export async function getReviews(itemKey: string) {  
 return api<any[]>(`/api/reviews?itemKey=${encodeURIComponent(itemKey)}`);  
}  
export async function postReview(form: FormData) {  
 return api<any>(`/api/reviews`, { method: 'POST', body: form });  
}  
export async function getCombos() { return api<any[]>(`/api/combos`); }  
export async function postCombo(payload: any) {  
 return api<any>(`/api/combos`, { method: 'POST', headers: { 'Content-Type': 'application/json' }, body: JSON.stringify(payload) });  
}  
export async function getRecs(date?: string) {  
 const q = date ? `?date=${date}` : '';  
 return api<any[]>(`/api/recommendations${q}`);  
}

### web/src/main.tsx

import React from 'react';  
import ReactDOM from 'react-dom/client';  
import App from './App';  
import './styles.css';  
  
ReactDOM.createRoot(document.getElementById('root')!).render(  
 <React.StrictMode>  
 <App />  
 </React.StrictMode>  
);  
  
if ('serviceWorker' in navigator) {  
 window.addEventListener('load', () => navigator.serviceWorker.register('/sw.js'));  
}

### web/src/App.tsx

import { useEffect, useState } from 'react';  
import Home from './pages/Home';  
import Combos from './pages/Combos';  
import Trends from './pages/Trends';  
  
export default function App() {  
 const [tab, setTab] = useState<'home'|'combos'|'trends'>('home');  
  
 useEffect(() => { document.title = 'Rathbone – What\'s Good Today?'; }, []);  
  
 return (  
 <div className="container">  
 <header>  
 <h1>Rathbone</h1>  
 <nav>  
 <button className={tab==='home'?'active':''} onClick={() => setTab('home')}>Today</button>  
 <button className={tab==='combos'?'active':''} onClick={() => setTab('combos')}>Combos</button>  
 <button className={tab==='trends'?'active':''} onClick={() => setTab('trends')}>Trends</button>  
 </nav>  
 </header>  
 {tab === 'home' && <Home/>}  
 {tab === 'combos' && <Combos/>}  
 {tab === 'trends' && <Trends/>}  
 </div>  
 );  
}

### web/src/pages/Home.tsx

import { useEffect, useState } from 'react';  
import { getMenu, getReviews } from '../api/client';  
import ItemCard from '../components/ItemCard';  
  
export default function Home() {  
 const [menu, setMenu] = useState<any[]>([]);  
 const [loading, setLoading] = useState(true);  
  
 useEffect(() => {  
 (async () => {  
 try { setMenu(await getMenu()); } finally { setLoading(false); }  
 })();  
 }, []);  
  
 if (loading) return <p className="muted">Loading today\'s menu…</p>;  
 if (!menu.length) return <p>No menu found for today.</p>;  
  
 return (  
 <div className="grid">  
 {menu.map(item => <ItemCard key={item.key} item={item} />)}  
 </div>  
 );  
}

### web/src/components/ItemCard.tsx

import { useEffect, useState } from 'react';  
import RatingStars from './RatingStars';  
import ReviewModal from './ReviewModal';  
import { getReviews } from '../api/client';  
  
export default function ItemCard({ item }: { item: any }) {  
 const [reviews, setReviews] = useState<any[]>([]);  
 const [open, setOpen] = useState(false);  
  
 useEffect(() => { (async () => setReviews(await getReviews(item.key)))(); }, [item.key]);  
 const avg = reviews.length ? (reviews.reduce((a, r) => a + r.rating, 0) / reviews.length).toFixed(1) : '—';  
  
 return (  
 <div className="card">  
 <div className="card-head">  
 <div>  
 <h3>{item.name}</h3>  
 <div className="meta">{item.station} • {item.calories ? `${item.calories} cal` : 'cal n/a'}</div>  
 </div>  
 <div className="avg"><span>{avg}</span><RatingStars value={Number(avg)||0} readOnly/></div>  
 </div>  
 <div className="tags">  
 {(item.dietary||[]).map((t: string) => <span key={t} className="tag diet">{t}</span>)}  
 {(item.allergens||[]).map((t: string) => <span key={t} className="tag warn">{t}</span>)}  
 </div>  
 <button onClick={() => setOpen(true)}>Rate / Review</button>  
 {open && <ReviewModal item={item} onClose={() => setOpen(false)} onSubmitted={() => (async()=>setReviews(await getReviews(item.key)))()} />}  
 </div>  
 );  
}

### web/src/components/ReviewModal.tsx

import { useRef, useState } from 'react';  
import { postReview } from '../api/client';  
import RatingStars from './RatingStars';  
  
export default function ReviewModal({ item, onClose, onSubmitted }: any) {  
 const [rating, setRating] = useState(5);  
 const [text, setText] = useState('');  
 const fileRef = useRef<HTMLInputElement>(null);  
 const [busy, setBusy] = useState(false);  
  
 async function submit() {  
 setBusy(true);  
 try {  
 const fd = new FormData();  
 fd.append('itemKey', item.key);  
 fd.append('rating', String(rating));  
 fd.append('text', text);  
 for (const f of (fileRef.current?.files || [])) fd.append('images', f);  
 await postReview(fd);  
 onSubmitted?.();  
 onClose();  
 } catch (e) {  
 alert('Failed to post review');  
 } finally { setBusy(false); }  
 }  
  
 return (  
 <div className="modal">  
 <div className="modal-card">  
 <h4>Review: {item.name}</h4>  
 <RatingStars value={rating} onChange={setRating}/>  
 <textarea placeholder="How was it? Any tips?" value={text} onChange={e=>setText(e.target.value)} />  
 <input ref={fileRef} type="file" accept="image/\*" multiple />  
 <div className="row">  
 <button onClick={onClose} className="secondary">Cancel</button>  
 <button onClick={submit} disabled={busy}>{busy?'Posting…':'Post'}</button>  
 </div>  
 </div>  
 </div>  
 );  
}

### web/src/components/RatingStars.tsx

export default function RatingStars({ value, onChange, readOnly=false }: { value: number; onChange?: (v:number)=>void; readOnly?: boolean }) {  
 const stars = [1,2,3,4,5];  
 return (  
 <div className="stars">  
 {stars.map(s => (  
 <button key={s} className={s <= value ? 'star on' : 'star'} onClick={()=>!readOnly&&onChange?.(s)} aria-label={`${s} star`}>★</button>  
 ))}  
 </div>  
 );  
}

### web/src/pages/Combos.tsx

import { useEffect, useState } from 'react';  
import { getCombos, postCombo } from '../api/client';  
  
export default function Combos() {  
 const [combos, setCombos] = useState<any[]>([]);  
 const [title, setTitle] = useState('');  
 const [desc, setDesc] = useState('');  
 const [items, setItems] = useState('');  
 useEffect(()=>{ (async()=>setCombos(await getCombos()))(); },[]);  
  
 async function submit() {  
 if (!title) return;  
 await postCombo({ title, description: desc, items: items.split(',').map(s=>s.trim()), date: new Date().toISOString().slice(0,10) });  
 setTitle(''); setDesc(''); setItems('');  
 setCombos(await getCombos());  
 }  
  
 return (  
 <div className="narrow">  
 <h2>Create a combo</h2>  
 <input placeholder="Title (e.g., Spicy Chickpea Bowl)" value={title} onChange={e=>setTitle(e.target.value)} />  
 <textarea placeholder="What did you mix?" value={desc} onChange={e=>setDesc(e.target.value)} />  
 <input placeholder="Related items (comma-separated)" value={items} onChange={e=>setItems(e.target.value)} />  
 <button onClick={submit}>Post combo</button>  
  
 <h2>Latest combos</h2>  
 {combos.map(c => (  
 <div className="card" key={c.comboId}>  
 <h3>{c.title}</h3>  
 <div className="meta">{c.date}</div>  
 <p>{c.description}</p>  
 {c.items?.length ? <div className="tags">{c.items.map((i:string)=><span key={i} className="tag">{i}</span>)}</div>: null}  
 </div>  
 ))}  
 </div>  
 );  
}

### web/src/pages/Trends.tsx

import { useEffect, useState } from 'react';  
import { getRecs } from '../api/client';  
import TrendsChart from '../components/TrendsChart';  
  
export default function Trends() {  
 const [items, setItems] = useState<any[]>([]);  
 useEffect(()=>{ (async()=>setItems(await getRecs()))(); },[]);  
  
 return (  
 <div>  
 <h2>Trending now</h2>  
 <p className="muted">Ranked by recency‑weighted ratings and review count.</p>  
 <TrendsChart data={items.slice(0,10).map(i=>({ name: i.name, score: i.trend?.score||0 }))} />  
 <div className="grid">  
 {items.map(i => (  
 <div key={i.key} className="card">  
 <h3>{i.name}</h3>  
 <div className="meta">{i.station}</div>  
 <div className="meta">{i.trend ? `${i.trend.avg.toFixed(1)}★ · ${i.trend.count} reviews` : 'no recent reviews'}</div>  
 </div>  
 ))}  
 </div>  
 </div>  
 );  
}

### web/src/components/TrendsChart.tsx

import { ResponsiveContainer, BarChart, Bar, XAxis, YAxis, Tooltip } from 'recharts';  
  
export default function TrendsChart({ data }: { data: { name: string; score: number }[] }) {  
 return (  
 <div style={{ width: '100%', height: 280 }}>  
 <ResponsiveContainer>  
 <BarChart data={data}>  
 <XAxis dataKey="name" hide={false} angle={-20} height={60} interval={0}/>  
 <YAxis />  
 <Tooltip />  
 <Bar dataKey="score" />  
 </BarChart>  
 </ResponsiveContainer>  
 </div>  
 );  
}

### web/src/styles.css

:root { color-scheme: light dark; --bg:#0b132b; --fg:#e0fbfc; --brand:#2ec4b6; --muted:#94a3b8; }  
\*{box-sizing:border-box} body{margin:0; font-family: ui-sans-serif, system-ui; background:#0b132b; color:#e2e8f0}  
.container{max-width:900px; margin:0 auto; padding:16px}  
header{display:flex; align-items:center; justify-content:space-between; gap:12px; position:sticky; top:0; background:linear-gradient(#0b132b, #0b132bf0); backdrop-filter:saturate(120%) blur(6px); padding:8px 0;}  
h1{font-size:22px; margin:0}  
nav button{background:transparent; border:1px solid #334155; color:#e2e8f0; padding:6px 10px; border-radius:999px; margin-left:8px}  
nav button.active{border-color:var(--brand)}  
.grid{display:grid; grid-template-columns:repeat(auto-fill, minmax(260px, 1fr)); gap:12px}  
.card{background:#111827; border:1px solid #1f2937; border-radius:16px; padding:12px}  
.card-head{display:flex; align-items:center; justify-content:space-between; gap:8px}  
.meta{color:var(--muted); font-size:12px}  
.tag{display:inline-block; margin:4px 6px 0 0; font-size:11px; padding:3px 8px; border-radius:10px; border:1px solid #334155}  
.tag.diet{border-color:#10b981}  
.tag.warn{border-color:#f59e0b}  
.avg{display:flex; align-items:center; gap:8px}  
.stars{display:inline-flex; gap:4px}  
.star{background:none; border:none; font-size:18px; filter:grayscale(1); opacity:.6}  
.star.on{filter:none; opacity:1}  
button{cursor:pointer}  
button.secondary{background:#0b132b; border:1px solid #334155; color:#e2e8f0}  
.modal{position:fixed; inset:0; background:rgba(0,0,0,.6); display:grid; place-items:center}  
.modal-card{width:min(560px, 92vw); background:#0b132b; border:1px solid #233041; border-radius:16px; padding:16px}  
textarea, input{width:100%; background:#0f172a; border:1px solid #233041; color:#e2e8f0; border-radius:10px; padding:10px; margin:6px 0}  
.row{display:flex; justify-content:flex-end; gap:10px}  
.muted{color:var(--muted)}  
.narrow{max-width:680px; margin:0 auto}

## 3) README & setup

### .env.example

# Server  
PORT=4000  
MONGODB\_URI=mongodb://127.0.0.1:27017/rathbone  
# Optional: change if Lehigh Dining updates their URL  
RATHBONE\_URL=https://lehigh.sodexomyway.com/en-us/locations/rathbone-dining-hall  
UPLOAD\_DIR=uploads  
STORAGE=fs  
# If you want S3 for photos  
S3\_BUCKET=your-bucket  
S3\_REGION=us-east-1  
S3\_ACCESS\_KEY\_ID=...  
S3\_SECRET\_ACCESS\_KEY=...

### Running locally

# 1) API  
cd server  
cp .env.example .env  
pnpm i # or npm i / yarn  
pnpm dev  
  
# 2) Frontend  
cd ../web  
pnpm i  
pnpm dev  
# open http://localhost:5173 and “Install App” from your browser’s menu

### Deploy notes

* **API**: Render, Fly.io, Railway, or Heroku (enable Playwright/Chromium buildpack; disable sandbox or add --no-sandbox).
* **DB**: MongoDB Atlas free tier.
* **Web**: Vercel/Netlify. Set proxy to your API base or serve both from one domain to simplify cookies and PWA scope.

### Data & scraping ethics

* Cache the daily menu in Mongo to minimize scraping.
* Respect robots.txt and the site’s Terms; if Lehigh Dining exposes an official JSON feed later, point the scraper to that endpoint for stability.

### How recommendations work

* Reviews store a 1–5 rating.
* A nightly job computes **recency‑weighted scores** (exponential decay; half‑life 10 days) and adds a +log(1+count) popularity bonus.
* The **Today** page shows current menu with each item’s live average; the **Trends** page ranks items by the score above.

## 4) Extras you can add later

* Google/Lehigh SSO (or passwordless email links) if you want verified users.
* Basic moderation: image size/type check, profanity filter, per‑device rate limits.
* Dietary filters (vegan/halal/allergens) directly on the home screen.
* Background Sync in the service worker to queue reviews offline and auto‑send when online.
* Campus‑wide feed that merges combos + reviews from multiple dining halls.

### Quick API examples

GET /api/menu?date=2025-10-18 # today’s items, scraped & cached  
GET /api/reviews?itemKey=<slug> # list reviews for an item  
POST /api/reviews (multipart/form-data)  
 fields: itemKey, rating, text, images[]  
POST /api/combos { title, description, items[], date }  
GET /api/recommendations?date=YYYY-MM-DD

If the Rathbone HTML structure changes, just tweak server/src/scraper/selectors.ts.