**1. Add a “Book” Button to Your Website**

**Description:**  
The system provides a dedicated booking link which you can embed as a button on any website.

**React Example:**

// Paste this in your React site wherever you want the booking button  
  
const BOOKING\_URL = "https://your-business.fresha.com/booking-link"; // Replace with your link  
  
function BookNowButton() {  
 return (  
 <a  
 href={BOOKING\_URL}  
 target="\_blank"  
 rel="noopener noreferrer"  
 style={{  
 background: "#2979ff",  
 color: "#fff",  
 padding: "12px 28px",  
 borderRadius: "4px",  
 fontWeight: "bold",  
 textDecoration: "none",  
 display: "inline-block",  
 fontSize: "18px",  
 }}  
 >  
 Book Now  
 </a>  
 );  
}  
  
export default BookNowButton;

* To use in pure HTML:

<a href="https://your-business.fresha.com/booking-link"  
 target="\_blank"  
 style="background:#2979ff;color:#fff;padding:12px 28px;border-radius:4px;font-weight:bold;text-decoration:none;display:inline-block;font-size:18px;">  
 Book Now  
</a>

Replace with your actual booking link from Fresha Link Builder.[[1]](#fn1)

**2. Track Meta (Facebook) Pixel Ads**

**Description:**  
Insert the Meta Pixel tracking code (from your Facebook Ad Manager) into your booking website. Fresha supports key FB/Meta events.

**React Example:**

import { useEffect } from "react";  
  
const META\_PIXEL\_ID = "123456789012345"; // Replace with your Pixel ID  
  
function FacebookPixel() {  
 useEffect(() => {  
 if (window.fbq) return;  
 !(function(f, b, e, v, n, t, s) {  
 if (f.fbq) return; n = f.fbq = function() {n.callMethod ? n.callMethod.apply(n, arguments) : n.queue.push(arguments);}  
 if (!f.\_fbq) f.\_fbq = n; n.push = n; n.loaded = !0; n.version = '2.0';  
 n.queue = []; t = b.createElement(e); t.async = !0;  
 t.src = v; s = b.getElementsByTagName(e)[^1\_0];  
 s.parentNode.insertBefore(t, s)  
 })(window, document, 'script',  
 'https://connect.facebook.net/en\_US/fbevents.js');  
 window.fbq('init', META\_PIXEL\_ID);  
 window.fbq('track', 'PageView');  
 }, []);  
  
 return (  
 <noscript>  
 <img  
 height="1"  
 width="1"  
 style={{ display: "none" }}  
 src={`https://www.facebook.com/tr?id=${META\_PIXEL\_ID}&ev=PageView&noscript=1`}  
 alt=""  
 />  
 </noscript>  
 );  
}  
  
export default FacebookPixel;

**Insert <FacebookPixel />** into your site's <head> or root component.  
For events (like booking confirmed), call:

window.fbq('track', 'Reservation'); // 'Reservation' for bookings, see doc for more events

Make sure your Pixel ID is created in your Facebook Business settings.[[2]](#fn2)

**3. Create Your Online Profile**

**Description:**  
Automate the profile setup wizard for a business marketplace.

**Example process (Node/Express pseudo-code, but can adapt to FE/BE):**

// Backend endpoint to receive form data and save full business profile  
app.post('/api/profile', (req, res) => {  
 const {  
 name, contactEmail, phone, address, openingHours,  
 images, amenities, highlights, values, description  
 } = req.body;  
  
 // Validate/set required fields, then:  
 Profile.create({  
 name, contactEmail, phone, address, openingHours,  
 images, amenities, highlights, values, description  
 }).then(profile => res.json({ success: true, profile }));  
});

**React steps for FE:**

* Use a multi-step form wizard.
* Each step captures fields shown in the doc (name, location, photos, values, etc.).
* On final step, POST the full profile.

**4. Get Bookings from Google Search and Maps**

**Description:**  
This is done through “Reserve with Google” integration. Your backend must match data to Google Business Profile.

**Automated Integration (Pseudo-Python):**

import requests  
  
def enable\_google\_reserve(fresha\_api\_token, business\_id):  
 # Where available, you'd enable with Fresha API (otherwise, headless browser automation)  
 url = f"https://api.fresha.com/businesses/{business\_id}/addons/google-reserve/enable"  
 headers = {"Authorization": f"Bearer {fresha\_api\_token}"}  
 resp = requests.post(url, headers=headers)  
 return resp.json()  
  
# Use above when business profile is set up and match details as per doc references[^1\_4].

* Sync business name/address with Google Business Profile for one-click enable.

**5. Set Up Facebook and Instagram Bookings**

**Description:**  
This enables "Book Now" buttons on social profiles, integrating via the Meta API and Fresha backend.

**Steps (FE/BE):**

* **React/Next.js**: Use a redirect flow to connect Meta business account
* **Express Example** (/api/connect-facebook endpoint)

app.get('/api/connect-facebook', (req, res) => {  
 // Redirect to Facebook's authorization page with Fresha as partner  
 const params =  
 `client\_id=${META\_APP\_ID}&redirect\_uri=${YOUR\_REDIRECT\_URI}&scope=pages\_show\_list,pages\_manage\_metadata`;  
 res.redirect(`https://www.facebook.com/v10.0/dialog/oauth?${params}`);  
});

After Meta auth, connect Fresha with the returned token and list business pages for "Book Now" buttons.[[3]](#fn3)

**Further integrations for “bundles”, “service add-ons”, “extra time”, “variants” etc. can all be scaffolded with robust FE forms saving to your backend using the data keys described in each document (use database schemas based on the field names found in the summaries).**

* For all booking or service objects, use arrays and linkage (e.g., bundles: array of service IDs, etc.)
* Use switch logic for handling variants, durations, pricing types, and extras.

Here’s how to implement **all booking/marketplace features with BOTH frontend and backend code** for the main business flows.

**1. Single Service Creation**

**Frontend (React):**

// src/components/CreateServiceForm.js  
import { useState } from "react";  
  
function CreateServiceForm() {  
 const [form, setForm] = useState({  
 name: "",  
 description: "",  
 price: "",  
 duration: "",  
 teamMembers: [],  
 locations: [],  
 resources: [],  
 category: "",  
 gender: "any",  
 isOnlineBookable: true,  
 });  
  
 function handleChange(e) {  
 setForm({ ...form, [e.target.name]: e.target.value });  
 }  
  
 function handleSubmit(e) {  
 e.preventDefault();  
 fetch("/api/services", {  
 method: "POST",  
 headers: { "Content-Type": "application/json" },  
 body: JSON.stringify(form),  
 })  
 .then((res) => res.json())  
 .then((data) => alert("Service created!"));  
 }  
  
 return (  
 <form onSubmit={handleSubmit}>  
 <input name="name" value={form.name} onChange={handleChange} placeholder="Service Name" />  
 <textarea name="description" value={form.description} onChange={handleChange} placeholder="Description" />  
 <input name="price" type="number" value={form.price} onChange={handleChange} placeholder="Price" />  
 <input name="duration" type="number" value={form.duration} onChange={handleChange} placeholder="Duration (min)" />  
 {/\* Add dropdowns for teamMembers, locations, resources, category as needed \*/}  
 <button type="submit">Create Service</button>  
 </form>  
 );  
}  
  
export default CreateServiceForm;

**Backend (Node/Express):**

// routes/services.js  
const express = require('express');  
const router = express.Router();  
const Service = require('../models/service');  
  
router.post('/', async (req, res) => {  
 const service = new Service(req.body);  
 await service.save();  
 res.json({ success: true, service });  
});  
  
module.exports = router;  
  
// models/service.js (Mongoose)  
const mongoose = require('mongoose');  
  
const ServiceSchema = new mongoose.Schema({  
 name: String,  
 description: String,  
 price: Number,  
 duration: Number,  
 teamMembers: [String],  
 locations: [String],  
 resources: [String],  
 category: String,  
 gender: String,  
 isOnlineBookable: Boolean,  
});  
  
module.exports = mongoose.model('Service', ServiceSchema);

**2. Bundle Creation**

**Frontend (React):**

// src/components/CreateBundleForm.js  
import { useState } from "react";  
  
function CreateBundleForm({ services }) {  
 const [bundle, setBundle] = useState({  
 name: "",  
 description: "",  
 services: [],  
 priceType: "custom", // or "service"  
 customPrice: "",  
 category: "",  
 gender: "any",  
 isOnlineBookable: true,  
 });  
  
 function handleChange(e) {  
 setBundle({ ...bundle, [e.target.name]: e.target.value });  
 }  
  
 function handleServiceSelect(svcId) {  
 setBundle(prev => ({  
 ...prev,  
 services: prev.services.includes(svcId)  
 ? prev.services.filter(id => id !== svcId)  
 : [...prev.services, svcId],  
 }));  
 }  
  
 function handleSubmit(e) {  
 e.preventDefault();  
 fetch("/api/bundles", {  
 method: "POST",  
 headers: { "Content-Type": "application/json" },  
 body: JSON.stringify(bundle),  
 }).then(res => res.json()).then(() => alert("Bundle created!"));  
 }  
  
 return (  
 <form onSubmit={handleSubmit}>  
 <input name="name" value={bundle.name} onChange={handleChange} placeholder="Bundle Name" />  
 <textarea name="description" value={bundle.description} onChange={handleChange} placeholder="Description" />  
 {/\* List all services with checkboxes \*/}  
 {services.map(svc => (  
 <label key={svc.\_id}>  
 <input  
 type="checkbox"  
 checked={bundle.services.includes(svc.\_id)}  
 onChange={() => handleServiceSelect(svc.\_id)}  
 />  
 {svc.name}  
 </label>  
 ))}  
 <input name="customPrice" type="number" value={bundle.customPrice} onChange={handleChange} placeholder="Custom Price" />  
 {/\* Add dropdowns for category, gender \*/}  
 <button type="submit">Create Bundle</button>  
 </form>  
 );  
}  
  
export default CreateBundleForm;

**Backend (Node/Express):**

// routes/bundles.js  
const express = require('express');  
const router = express.Router();  
const Bundle = require('../models/bundle');  
  
router.post('/', async (req, res) => {  
 const bundle = new Bundle(req.body);  
 await bundle.save();  
 res.json({ success: true, bundle });  
});  
  
module.exports = router;  
  
// models/bundle.js  
const mongoose = require('mongoose');  
  
const BundleSchema = new mongoose.Schema({  
 name: String,  
 description: String,  
 services: [mongoose.Schema.Types.ObjectId], // references Service  
 priceType: String,  
 customPrice: Number,  
 category: String,  
 gender: String,  
 isOnlineBookable: Boolean,  
});  
  
module.exports = mongoose.model('Bundle', BundleSchema);

**3. Service Add-ons**

**Frontend (React):**

function AddonManager({ serviceId }) {  
 const [groupName, setGroupName] = useState("");  
 const [prompt, setPrompt] = useState("");  
 const [options, setOptions] = useState([]);  
 const [newOption, setNewOption] = useState({ name: "", price: 0, description: "" });  
  
 function addOption() {  
 setOptions([...options, newOption]);  
 setNewOption({ name: "", price: 0, description: "" });  
 }  
  
 function saveAddons() {  
 fetch(`/api/services/${serviceId}/addons`, {  
 method: "POST",  
 headers: { "Content-Type": "application/json" },  
 body: JSON.stringify({ groupName, prompt, options }),  
 });  
 }  
  
 return (  
 <div>  
 <input value={groupName} onChange={e => setGroupName(e.target.value)} placeholder="Group name" />  
 <input value={prompt} onChange={e => setPrompt(e.target.value)} placeholder="Prompt to client" />  
 {/\* Options (list and add) \*/}  
 <div>  
 <input value={newOption.name} onChange={e => setNewOption({...newOption, name: e.target.value })} placeholder="Option name" />  
 <input value={newOption.price} onChange={e => setNewOption({...newOption, price: Number(e.target.value) })} type="number" placeholder="Price" />  
 <input value={newOption.description} onChange={e => setNewOption({...newOption, description: e.target.value })} placeholder="Description" />  
 <button onClick={addOption}>Add Option</button>  
 </div>  
 <button onClick={saveAddons}>Save Add-ons</button>  
 <ul>{options.map((opt,i)=><li key={i}>{opt.name} - {opt.price}</li>)}</ul>  
 </div>  
 );  
}

**Backend (Node/Express):**

// routes/services.js (add to existing file)  
router.post('/:serviceId/addons', async (req, res) => {  
 const { groupName, prompt, options } = req.body;  
 const service = await Service.findById(req.params.serviceId);  
 service.addons.push({ groupName, prompt, options });  
 await service.save();  
 res.json({ success: true });  
});  
  
// models/service.js (extend schema)  
const AddonSchema = new mongoose.Schema({  
 groupName: String,  
 prompt: String,  
 options: [{ name: String, price: Number, description: String }],  
});  
ServiceSchema.addons = [AddonSchema];

**4. Service Variant Creation (Pricing & Duration per Staff/Location)**

**Frontend:**  
Present a UI to add multiple variants for a service, each with its own name, price, duration, SKU, and location or team member pricing.

**Backend:**  
Add to ServiceSchema:

const VariantSchema = new mongoose.Schema({  
 name: String,  
 duration: Number,  
 priceType: String,  
 price: Number,  
 sku: String,  
 staffPricing: [{  
 teamMember: String,  
 price: Number  
 }],  
 locationPricing: [{  
 location: String,  
 price: Number  
 }]  
});  
ServiceSchema.variants = [VariantSchema];  
  
router.post('/:serviceId/variants', async (req, res) => {  
 const { variant } = req.body;  
 const service = await Service.findById(req.params.serviceId);  
 service.variants.push(variant);  
 await service.save();  
 res.json({ success: true });  
});

**5. Facebook/Instagram/Google Integration**

**Frontend:**  
Display a connection button. On click, redirect to the OAuth endpoint (Meta/Google) for admin authentication and integration.

**Backend Example:**

* Use passport.js or direct OAuth for authentication.
* Store OAuth tokens to allow your backend to list your business/profile and auto-generate “book now” links/buttons.

**6. Tracking Integrations (Meta Pixel, Google Analytics)**

**Frontend:**  
Add tracking scripts to booking flows (dynamically insert based on admin settings).

**Backend:**  
Provide APIs for admins to enter their Meta Pixel ID and Google Analytics Measurement ID, save in config.  
Serve booking flows with tracking scripts/IDs injected.

**Example React Component:**

function PixelAndAnalytics({ pixelId, gaId }) {  
 useEffect(() => {  
 if (pixelId) {/\* Insert FB Pixel script \*/}  
 if (gaId) {/\* Insert GA script \*/}  
 }, [pixelId, gaId]);  
 return null;  
}

**7. Extra Time for Services/Appointments**

**Frontend:**  
UI fields for Process/Blocked/Extra Time when adding or editing a service/appointment.

**Backend:**  
Add extraTime field to Service and Appointment schema (type, duration, etc.).  
Update appointment duration calculations to consider extra times.

Here's a complete frontend and backend architecture for **ALL marketplace business flows, including service setup, profile, add-ons, bundles, booking links, integrations, and analytics**. You can use **React for frontend** and **Node/Express with MongoDB for backend** (adaptable for Django/Python—just ask!), covering everything listed in the docs:

**1. Marketplace Business Profile Setup**

**Frontend: Multi-step React Form**

// ProfileForm.js  
import { useState } from "react";  
  
const defaultProfile = {  
 displayName: "",  
 phone: "",  
 email: "",  
 address: "",  
 hours: {}, // { Mon:"10:00-19:00", ... }  
 images: [],  
 amenities: [],  
 highlights: [],  
 values: [],  
 description: "",  
 paymentInfo: {},  
 googleBusinessId: "",  
};  
  
export default function ProfileForm() {  
 const [profile, setProfile] = useState(defaultProfile);  
  
 function handleChange(e) {  
 setProfile({...profile, [e.target.name]: e.target.value });  
 }  
  
 function handleSave() {  
 fetch("/api/profile", {  
 method:"POST",  
 headers:{"Content-Type":"application/json"},  
 body:JSON.stringify(profile),  
 }).then(res=>res.json()).then(data=>alert("Profile saved!"));  
 }  
  
 // Render inputs for each section (displayName, hours, amenities, etc.)...  
 // Render file uploader for images  
 return /\* JSX form \*/;  
}

**Backend:**

// models/Profile.js  
const mongoose = require('mongoose');  
const ProfileSchema = new mongoose.Schema({  
 displayName: String,  
 phone: String,  
 email: String,  
 address: String,  
 hours: Object,  
 images: [String],  
 amenities: [String],  
 highlights: [String],  
 values: [String],  
 description: String,  
 paymentInfo: Object,  
 googleBusinessId: String,  
});  
module.exports = mongoose.model('Profile', ProfileSchema);  
  
// routes/profile.js  
const Profile = require('../models/Profile');  
router.post('/', async (req,res)=>{  
 const profile = await Profile.findOneAndUpdate(  
 {}, req.body, {upsert:true, new:true}  
 );  
 res.json({ success:true, profile });  
});

**2. Service, Bundle, Variant, Add-on, and Pricing Management**

**Frontend:**  
Build React forms for:

* Create single service
* Create bundle (multi-service)
* Create variants (price/duration for each team/location)
* Create add-ons (extra client options per service)

Example Service Form:

function ServiceForm({ submitURL }) {  
 // Handles name, type, duration, teamMembers, locations, price, add-ons, variants, resources  
 // Use multiple useState hooks  
 return (  
 <form onSubmit={handleSubmit}>  
 {/\* Fields for name, type, duration, etc. \*/}  
 </form>  
 );  
}

Bundle, Add-on, and Variant forms should have multi-select and dynamic “add option” inputs.

**Backend:**

// models/Service.js  
const mongoose = require('mongoose');  
  
const VariantSchema = new mongoose.Schema({  
 name: String, duration: Number, price: Number, sku: String,  
 teamPricing: [{ teamMember: String, price: Number }],  
 locPricing: [{ location: String, price: Number }]  
});  
  
const AddonSchema = new mongoose.Schema({  
 groupName: String, prompt: String,  
 options: [{ name: String, price: Number, description: String, extraTime: Number }]  
});  
  
const ServiceSchema = new mongoose.Schema({  
 name: String,  
 type: String,  
 category: String,  
 description: String,  
 duration: Number,  
 price: Number,  
 teamMembers: [String],  
 locations: [String],  
 resources: [String],  
 variants: [VariantSchema],  
 addons: [AddonSchema],  
 gender: String,  
 isOnlineBookable: Boolean,  
 extraTime: [{type:"processing",duration:Number},{type:"blocked",duration:Number}]  
});  
  
module.exports = mongoose.model('Service', ServiceSchema);  
  
// routes/services.js  
const Service = require('../models/Service');  
router.post('/', async (req,res)=>{  
 const service = new Service(req.body); await service.save();  
 res.json({ success:true, service });  
});  
router.post('/:serviceId/variant', async (req,res)=>{  
 const service = await Service.findById(req.params.serviceId);  
 service.variants.push(req.body); await service.save();  
 res.json({ success:true });  
});  
router.post('/:serviceId/addon', async (req,res)=>{  
 const service = await Service.findById(req.params.serviceId);  
 service.addons.push(req.body); await service.save();  
 res.json({ success:true });  
});

Create similar models for **Bundle** and **Appointment**.

**3. Online Booking Link and QR Integration**

**Frontend:**  
Show the booking button/link anywhere:

function BookingButton({ businessUrl }) {  
 return (  
 <a href={businessUrl} target="\_blank" rel="noopener noreferrer"  
 style={{  
 background:"#2979FF",color:"#fff",padding:"14px 30px",borderRadius:"6px",fontWeight:"bold",  
 fontSize:"20px", display:"inline-block",  
 }}>  
 Book Now  
 </a>  
 );  
}

Generate QR codes using a package (e.g., qrcode.react).

**Backend:**  
Booking links are part of the business profile. Optionally, build a QR endpoint that returns a PNG using a library.

**4. Google Search/Maps & Reserve Integration**

**Frontend:**  
Admin UI for entering Google Business ID and matching service/location names.

**Backend:**  
On save, trigger/schedule a call to Google Reserve API (or manual instructions if integration isn't public) to sync profile and show the Book button on Google (see docs for field requirements—ensure names and addresses match exactly).

**5. Facebook & Instagram Book Now Integrations**

**Frontend:**  
Provide a “Connect Facebook/Instagram” button, triggering OAuth pop-up and redirect (see Meta docs).

**Backend:**  
OAuth endpoints (with passport-facebook or Meta’s own endpoint). Store resulting tokens, manage bookings created via API webhooks. Add method to generate “book now” button and update the social media pages via API.

**6. Service Add-ons**

See section #2 above. Add-on groups and options link directly to service objects in MongoDB.

**7. Extra Time in Services/Appointments**

**Frontend:**  
When creating/editing a service, show controls to add “processing,” “blocked,” or “servicing” extra times (multi-select, ordered).

**Backend:**  
Service schema’s extraTime array as above. Also store on appointments.

**8. Google Analytics & Meta Pixel Tracking**

**Frontend:**  
Inject tracking code into booking/checkout flows:

// For Meta Pixel  
useEffect(()=>{  
 window.fbq('init', pixelId);  
 window.fbq('track', 'PageView');  
},[]);  
  
// For GA  
useEffect(()=>{  
 // Insert GA script with ID  
},[]);

Add hooks for events ('Reservation', 'Purchase', etc.).

**Backend:**  
Profile/settings Schema stores tracking IDs. Serve them to frontend.

**9. Google Review Boost**

**Frontend:**  
Show prompt after booking/review for clients to post the same review on Google (if trigger thresholds met).

**Backend:**  
After successful Fresha review, check rating, and present option/link to Google Review page (store Google Business Profile URLs).

**1. Business Profile Setup Page**

// pages/ProfileSetupPage.js  
import React, { useState } from "react";  
  
export default function ProfileSetupPage() {  
 const [profile, setProfile] = useState({  
 displayName: "",  
 phone: "",  
 email: "",  
 address: "",  
 hours: "",  
 description: "",  
 highlights: "",  
 amenities: "",  
 values: "",  
 googleBusinessId: "",  
 paymentInfo: "",  
 images: [],  
 });  
  
 function handleChange(e) {  
 setProfile({ ...profile, [e.target.name]: e.target.value });  
 }  
 function handleFile(e) {  
 setProfile({ ...profile, images: [...profile.images, ...e.target.files] });  
 }  
 function handleSubmit(e) {  
 e.preventDefault();  
 fetch("/api/profile", {  
 method:"POST",  
 body:JSON.stringify(profile),  
 headers:{ "Content-Type": "application/json" }  
 }).then(r=>alert("Profile saved successfully!"));  
 }  
 return (  
 <form onSubmit={handleSubmit}>  
 <input name="displayName" value={profile.displayName} onChange={handleChange} placeholder="Business Name" required />  
 <input name="phone" value={profile.phone} onChange={handleChange} placeholder="Phone" required />  
 <input name="email" value={profile.email} onChange={handleChange} placeholder="Email" required />  
 <input name="address" value={profile.address} onChange={handleChange} placeholder="Address" required />  
 <input name="hours" value={profile.hours} onChange={handleChange} placeholder="Opening hours" />  
 <textarea name="description" value={profile.description} onChange={handleChange} placeholder="Description" />  
 <input name="highlights" value={profile.highlights} onChange={handleChange} placeholder="Highlights (comma separated)" />  
 <input name="amenities" value={profile.amenities} onChange={handleChange} placeholder="Amenities (comma separated)" />  
 <input name="values" value={profile.values} onChange={handleChange} placeholder="Values (comma separated)" />  
 <input name="googleBusinessId" value={profile.googleBusinessId} onChange={handleChange} placeholder="Google Business ID" />  
 <input name="paymentInfo" value={profile.paymentInfo} onChange={handleChange} placeholder="Payment Info" />  
 <input type="file" multiple accept="image/\*" onChange={handleFile} />  
 <button type="submit">Save Profile</button>  
 </form>  
 );  
}

**2. Service Creation Page**

// pages/CreateServicePage.js  
import React, { useState } from "react";  
  
export default function CreateServicePage() {  
 const [form, setForm] = useState({  
 name: "",  
 type: "",  
 category: "",  
 description: "",  
 duration: 60,  
 price: 0,  
 teamMembers: [],  
 locations: [],  
 resources: [],  
 gender: "any",  
 isOnlineBookable: true,  
 extraTime: [],  
 });  
  
 function handleChange(e) {  
 setForm({ ...form, [e.target.name]: e.target.value });  
 }  
 function handleArrChange(name, value) {  
 setForm({ ...form, [name]: value.split(",") });  
 }  
 function handleSubmit(e) {  
 e.preventDefault();  
 fetch("/api/services", {  
 method: "POST",  
 body: JSON.stringify(form),  
 headers: { "Content-Type": "application/json" }  
 }).then(r=>alert("Service created!"));  
 }  
 return (  
 <form onSubmit={handleSubmit}>  
 <input name="name" value={form.name} onChange={handleChange} placeholder="Service Name" required />  
 <input name="type" value={form.type} onChange={handleChange} placeholder="Type" />  
 <input name="category" value={form.category} onChange={handleChange} placeholder="Category" />  
 <textarea name="description" value={form.description} onChange={handleChange} placeholder="Description" />  
 <input name="duration" value={form.duration} type="number" onChange={handleChange} placeholder="Duration (min)" />  
 <input name="price" value={form.price} type="number" onChange={handleChange} placeholder="Price" />  
 <input name="teamMembers" value={form.teamMembers.join(",")} onChange={e=>handleArrChange("teamMembers", e.target.value)} placeholder="Team Members (comma separated)" />  
 <input name="locations" value={form.locations.join(",")} onChange={e=>handleArrChange("locations", e.target.value)} placeholder="Locations (comma separated)" />  
 <input name="resources" value={form.resources.join(",")} onChange={e=>handleArrChange("resources", e.target.value)} placeholder="Resources (comma separated)" />  
 <input name="gender" value={form.gender} onChange={handleChange} placeholder="Gender" />  
 <button type="submit">Create Service</button>  
 </form>  
 );  
}

**3. Bundle Creation Page**

// pages/CreateBundlePage.js  
import React, { useState } from "react";  
  
export default function CreateBundlePage({ services = [] }) {  
 const [form, setForm] = useState({  
 name: "",  
 description: "",  
 services: [],  
 priceType: "custom",  
 customPrice: "",  
 category: "",  
 gender: "any",  
 isOnlineBookable: true,  
 });  
  
 function handleChange(e) {  
 setForm({ ...form, [e.target.name]: e.target.value });  
 }  
 function handleServiceSelect(svcId) {  
 setForm(prev => ({  
 ...prev,  
 services: prev.services.includes(svcId)  
 ? prev.services.filter(id => id !== svcId)  
 : [...prev.services, svcId],  
 }));  
 }  
 function handleSubmit(e) {  
 e.preventDefault();  
 fetch("/api/bundles", {  
 method: "POST",  
 body: JSON.stringify(form),  
 headers: { "Content-Type": "application/json" }  
 }).then(r=>{  
 alert("Bundle created!");  
 });  
 }  
 return (  
 <form onSubmit={handleSubmit}>  
 <input name="name" value={form.name} onChange={handleChange} placeholder="Bundle Name" />  
 <textarea name="description" value={form.description} onChange={handleChange} placeholder="Description" />  
 <div>  
 <label>Services:</label>  
 {services.map(svc => (  
 <label key={svc.\_id}>  
 <input  
 type="checkbox"  
 checked={form.services.includes(svc.\_id)}  
 onChange={() => handleServiceSelect(svc.\_id)}  
 />  
 {svc.name}  
 </label>  
 ))}  
 </div>  
 <input name="customPrice" type="number" value={form.customPrice} onChange={handleChange} placeholder="Custom Price" />  
 <input name="category" value={form.category} onChange={handleChange} placeholder="Category" />  
 <input name="gender" value={form.gender} onChange={handleChange} placeholder="Gender" />  
 <button type="submit">Create Bundle</button>  
 </form>  
 );  
}

**4. Service Add-ons Manager**

// pages/AddonsManagerPage.js  
import React, { useState } from "react";  
  
export default function AddonsManagerPage({ serviceId }) {  
 const [groupName, setGroupName] = useState("");  
 const [prompt, setPrompt] = useState("");  
 const [options, setOptions] = useState([]);  
 const [newOption, setNewOption] = useState({ name: "", price: 0, description: "" });  
  
 function addOption() {  
 setOptions([...options, newOption]);  
 setNewOption({ name: "", price: 0, description: "" });  
 }  
 function saveAddons() {  
 fetch(`/api/services/${serviceId}/addon`, {  
 method: "POST",  
 headers: { "Content-Type": "application/json" },  
 body: JSON.stringify({ groupName, prompt, options }),  
 }).then(r=>alert("Add-ons saved!"));  
 }  
 return (  
 <div>  
 <input value={groupName} onChange={e => setGroupName(e.target.value)} placeholder="Group name" />  
 <input value={prompt} onChange={e => setPrompt(e.target.value)} placeholder="Prompt to client" />  
 <div>  
 <input value={newOption.name} onChange={e => setNewOption({ ...newOption, name: e.target.value })} placeholder="Option name" />  
 <input value={newOption.price} type="number" onChange={e => setNewOption({ ...newOption, price: Number(e.target.value) })} placeholder="Price" />  
 <input value={newOption.description} onChange={e => setNewOption({ ...newOption, description: e.target.value })} placeholder="Description" />  
 <button onClick={addOption}>Add Option</button>  
 </div>  
 <button onClick={saveAddons}>Save Add-ons</button>  
 <ul>  
 {options.map((opt, i) => (  
 <li key={i}>{opt.name} - {opt.price}</li>  
 ))}  
 </ul>  
 </div>  
 );  
}

**5. Booking Button Component**

// components/BookingButton.js  
import React from "react";  
export default function BookingButton({ businessUrl }) {  
 return (  
 <a href={businessUrl} target="\_blank" rel="noopener noreferrer"  
 style={{  
 background:"#2979FF",color:"#fff",padding:"14px 30px",borderRadius:"6px",fontWeight:"bold",  
 fontSize:"20px",display:"inline-block",  
 }}>  
 Book Now  
 </a>  
 );  
}

**6. Integrations/Tracking Setup Page**

// pages/IntegrationsSetupPage.js  
import React, { useState } from "react";  
export default function IntegrationsSetupPage() {  
 const [pixelId, setPixelId] = useState("");  
 const [gaId, setGaId] = useState("");  
 function handleSubmit(e) {  
 e.preventDefault();  
 fetch("/api/settings", {  
 method:"POST",  
 body:JSON.stringify({ pixelId, gaId }),  
 headers:{ "Content-Type": "application/json" }  
 }).then(r=>alert("IDs Saved!"));  
 }  
 return (  
 <form onSubmit={handleSubmit}>  
 <h3>Meta Pixel</h3>  
 <input type="text" value={pixelId} onChange={e=>setPixelId(e.target.value)} placeholder="Meta Pixel ID" />  
 <h3>Google Analytics</h3>  
 <input type="text" value={gaId} onChange={e=>setGaId(e.target.value)} placeholder="GA Measurement ID" />  
 <button type="submit">Save IDs</button>  
 </form>  
 );  
}

**7. Google Review Boost Prompt Component**

// components/GoogleReviewPrompt.js  
import React from "react";  
export default function GoogleReviewPrompt({ showPrompt, googleUrl }) {  
 if (!showPrompt) return null;  
 return (  
 <div style={{ padding: 20, borderRadius: 8, background: "#f6f6f6", margin: "20px 0" }}>  
 <h2>Enjoyed your service?</h2>  
 <p>Share your feedback with others on Google!</p>  
 <a href={googleUrl} target="\_blank" rel="noopener noreferrer"  
 style={{ color: "#2979FF", fontWeight: "bold" }}>Leave a Google Review</a>  
 </div>  
 );  
}

**Navigation/Routing:**  
For full pages, use React Router routing in your app to navigate between pages (i.e. /profile, /services/new, /bundle/new, /settings/integrations, etc.).

Let me know if you need more specialized business logic, error handling, or style frameworks (Material UI, Ant Design, etc.), or want code for advanced flows like appointment booking UX, OAuth handoffs, or any other admin marketplace feature!