perfect—these shots cover: **payments (processors, manual methods, due dates, plans, tips)** and the **Booking Engine** (general copy, multiple-packages toggle, extra questions, unavailable notices, privacy consent). Below is a tight, drop-in set of schema + API routes + React pages to bring all of that online in your Replit scaffold right now.

# **0) Schema adds (append to /server/db/schema.sql)**

-- PAYMENT PROCESSORS (Stripe/Square/PayPal placeholders)

create table if not exists payment\_processors (

id uuid primary key default uuid\_generate\_v4(),

owner\_id uuid references users(id) on delete cascade,

kind text not null, -- 'stripe'|'square'|'paypal'

status text default 'disconnected', -- 'connected'|'disconnected'

public\_meta jsonb default '{}'::jsonb, -- e.g., merchant id label to display

secret\_meta jsonb default '{}'::jsonb, -- tokens/keys (MVP: store dummy)

created\_at timestamptz default now()

);

-- CUSTOMER PAYMENT OPTIONS (manual + card via processor)

create table if not exists payment\_methods (

id uuid primary key default uuid\_generate\_v4(),

owner\_id uuid references users(id) on delete cascade,

kind text not null, -- 'card'|'check'|'cash'|'venmo'|'zelle'|'cashapp'

label text not null, -- what shows on invoices

enabled boolean default false,

show\_on\_invoices boolean default true,

auto\_pay boolean default false, -- only meaningful for 'card'

sort\_index int default 0,

config jsonb default '{}'::jsonb -- handles venmo handle, cashapp $tag, etc.

);

-- PAYMENT SCHEDULE SETTINGS (global defaults)

create table if not exists payment\_settings (

owner\_id uuid primary key references users(id) on delete cascade,

deposit\_term text default 'retainer', -- display term for initial payment

deposit\_due\_days int default 0, -- 0 = due at booking, else N days before event

final\_due\_days int default 30, -- days before event

tips\_enabled boolean default true

);

-- PAYMENT PLANS (radio list in UI)

create table if not exists payment\_plans (

id uuid primary key default uuid\_generate\_v4(),

owner\_id uuid references users(id) on delete cascade,

name text not null, -- e.g., 'Retainer + final 30 days'

slug text unique, -- 'retainer\_final\_30', 'retainer\_monthly\_split', 'pay\_in\_full'

spec jsonb not null, -- {type:'retainer\_final', final\_due\_days:30} OR {type:'retainer\_monthly', months:3, final\_due\_days:30}

enabled boolean default true,

sort\_index int default 0

);

-- BOOKING ENGINE SETTINGS (per account)

create table if not exists booking\_engine\_settings (

owner\_id uuid primary key references users(id) on delete cascade,

header\_text text default '',

subheader\_text text default '',

cta\_text text default 'Choose Now',

allow\_multiple\_packages boolean default true,

enable\_captcha boolean default false

);

-- UNAVAILABLE NOTICES (default + per-category)

create table if not exists unavailable\_notices (

id uuid primary key default uuid\_generate\_v4(),

owner\_id uuid references users(id) on delete cascade,

category text default 'default', -- 'default' | 'photo\_booth' | etc.

body text not null

);

-- PRIVACY / CONSENT

create table if not exists privacy\_settings (

owner\_id uuid primary key references users(id) on delete cascade,

request\_consent boolean default true,

consent\_message text default '',

privacy\_policy\_url text default ''

);

-- BOOKING ENGINE EXTRA QUESTIONS (collected during booking)

create table if not exists booking\_questions (

id uuid primary key default uuid\_generate\_v4(),

owner\_id uuid references users(id) on delete cascade,

label text not null, -- 'Customer Address', 'Stair Setup', etc.

type text not null default 'text', -- 'text'|'textarea'|'select'|'checkbox'

options jsonb default '[]'::jsonb, -- for select/checkbox

placeholder text default '',

help\_text text default '',

required boolean default false,

collect\_during\_booking boolean default true,

internal\_only boolean default false, -- if just for staff view after booking

sort\_index int default 0

);

# **1) Server routes**

## **a) Payments — /server/routes/payments.js**

import { Router } from 'express'

import { pool } from '../index.js'

const r = Router()

const ownerId = process.env.DEMO\_OWNER\_ID

// PROCESSORS

r.get('/processors', async (\_req,res,next)=>{

try{

const {rows} = await pool.query('select \* from payment\_processors where owner\_id=$1 order by created\_at asc',[ownerId])

res.json(rows)

}catch(e){ next(e) }

})

// connect/disconnect mock endpoints (UI parity; you can wire actual APIs later)

r.post('/processors/:kind/connect', async (req,res,next)=>{

try{

const kind = req.params.kind

const {rows:[p]} = await pool.query(

`insert into payment\_processors (owner\_id,kind,status,public\_meta)

values ($1,$2,'connected', $3)

on conflict do nothing

returning \*`,

[ownerId, kind, {display:`Connected to ${kind}`}]

)

res.json(p || {ok:true})

}catch(e){ next(e) }

})

r.post('/processors/:kind/disconnect', async (req,res,next)=>{

try{

await pool.query('update payment\_processors set status=$1 where owner\_id=$2 and kind=$3',['disconnected', ownerId, req.params.kind])

res.json({ok:true})

}catch(e){ next(e) }

})

// METHODS

r.get('/methods', async (\_req,res,next)=>{

try{

const {rows} = await pool.query('select \* from payment\_methods where owner\_id=$1 order by sort\_index asc',[ownerId])

res.json(rows)

}catch(e){ next(e) }

})

r.post('/methods', async (req,res,next)=>{

try{

const { kind, label, enabled=false, show\_on\_invoices=true, auto\_pay=false, config={}, sort\_index=0 } = req.body

const {rows:[m]} = await pool.query(

`insert into payment\_methods (owner\_id,kind,label,enabled,show\_on\_invoices,auto\_pay,config,sort\_index)

values ($1,$2,$3,$4,$5,$6,$7,$8) returning \*`,

[ownerId, kind, label, enabled, show\_on\_invoices, auto\_pay, config, sort\_index]

)

res.status(201).json(m)

}catch(e){ next(e) }

})

r.post('/methods/:id', async (req,res,next)=>{

try{

const fields = ['label','enabled','show\_on\_invoices','auto\_pay','config','sort\_index']

const sets=[], vals=[]

fields.forEach((k,i)=>{ if(k in req.body){ sets.push(`${k}=$${i+1}`); vals.push(req.body[k]) }})

vals.push(req.params.id, ownerId)

await pool.query(`update payment\_methods set ${sets.join(',')} where id=$${vals.length-1} and owner\_id=$${vals.length}`, vals)

res.json({ok:true})

}catch(e){ next(e) }

})

// SETTINGS (due dates, tips)

r.get('/settings', async (\_req,res,next)=>{

try{

const {rows:[s]} = await pool.query('select \* from payment\_settings where owner\_id=$1',[ownerId])

res.json(s)

}catch(e){ next(e) }

})

r.post('/settings', async (req,res,next)=>{

try{

const f = ['deposit\_term','deposit\_due\_days','final\_due\_days','tips\_enabled']

const set=[], vals=[]

f.forEach((k,i)=>{ if(k in req.body){ set.push(`${k}=$${i+1}`); vals.push(req.body[k]) }})

vals.push(ownerId)

await pool.query(

`insert into payment\_settings (owner\_id) values ($${vals.length})

on conflict (owner\_id) do update set ${set.length?set.join(','):'owner\_id=excluded.owner\_id'}`, vals)

res.json({ok:true})

}catch(e){ next(e) }

})

// PLANS

r.get('/plans', async (\_req,res,next)=>{

try{

const {rows} = await pool.query('select \* from payment\_plans where owner\_id=$1 order by sort\_index asc',[ownerId])

res.json(rows)

}catch(e){ next(e) }

})

r.post('/plans', async (req,res,next)=>{

try{

const { name, slug, spec, enabled=true, sort\_index=0 } = req.body

const {rows:[p]} = await pool.query(

`insert into payment\_plans (owner\_id,name,slug,spec,enabled,sort\_index)

values ($1,$2,$3,$4,$5,$6) returning \*`,

[ownerId, name, slug, spec, enabled, sort\_index]

)

res.status(201).json(p)

}catch(e){ next(e) }

})

export default r

Mount in server/index.js:

import payments from './routes/payments.js'

app.use('/api/payments', payments)

## **b) Booking Engine — /server/routes/bookingEngine.js**

import { Router } from 'express'

import { pool } from '../index.js'

const r = Router()

const ownerId = process.env.DEMO\_OWNER\_ID

// GENERAL

r.get('/general', async (\_req,res,next)=>{

try{ const {rows:[g]} = await pool.query('select \* from booking\_engine\_settings where owner\_id=$1',[ownerId]); res.json(g) }

catch(e){ next(e) }

})

r.post('/general', async (req,res,next)=>{

try{

const f = ['header\_text','subheader\_text','cta\_text','allow\_multiple\_packages','enable\_captcha']

const set=[], vals=[]

f.forEach((k,i)=>{ if(k in req.body){ set.push(`${k}=$${i+1}`); vals.push(req.body[k]) }})

vals.push(ownerId)

await pool.query(

`insert into booking\_engine\_settings (owner\_id) values ($${vals.length})

on conflict (owner\_id) do update set ${set.length?set.join(','):'owner\_id=excluded.owner\_id'}`, vals)

res.json({ok:true})

}catch(e){ next(e) }

})

// UNAVAILABLE NOTICES

r.get('/notices', async (\_req,res,next)=>{

try{ const {rows}=await pool.query('select \* from unavailable\_notices where owner\_id=$1 order by category asc',[ownerId]); res.json(rows) }

catch(e){ next(e) }

})

r.post('/notices', async (req,res,next)=>{

try{

const { category='default', body } = req.body

const {rows:[n]} = await pool.query(

`insert into unavailable\_notices (owner\_id,category,body) values ($1,$2,$3)

on conflict do nothing returning \*`, [ownerId, category, body])

res.status(201).json(n || {ok:true})

}catch(e){ next(e) }

})

r.post('/notices/:id', async (req,res,next)=>{

try{

await pool.query('update unavailable\_notices set body=$1 where id=$2 and owner\_id=$3',[req.body.body, req.params.id, ownerId])

res.json({ok:true})

}catch(e){ next(e) }

})

// PRIVACY

r.get('/privacy', async (\_req,res,next)=>{

try{ const {rows:[p]}=await pool.query('select \* from privacy\_settings where owner\_id=$1',[ownerId]); res.json(p) }

catch(e){ next(e) }

})

r.post('/privacy', async (req,res,next)=>{

try{

const f=['request\_consent','consent\_message','privacy\_policy\_url']

const set=[], vals=[]

f.forEach((k,i)=>{ if(k in req.body){ set.push(`${k}=$${i+1}`); vals.push(req.body[k]) }})

vals.push(ownerId)

await pool.query(

`insert into privacy\_settings (owner\_id) values ($${vals.length})

on conflict (owner\_id) do update set ${set.length?set.join(','):'owner\_id=excluded.owner\_id'}`, vals)

res.json({ok:true})

}catch(e){ next(e) }

})

// EXTRA QUESTIONS

r.get('/questions', async (\_req,res,next)=>{

try{ const {rows}=await pool.query('select \* from booking\_questions where owner\_id=$1 order by sort\_index asc',[ownerId]); res.json(rows) }

catch(e){ next(e) }

})

r.post('/questions', async (req,res,next)=>{

try{

const { label, type='text', options=[], placeholder='', help\_text='', required=false, collect\_during\_booking=true, internal\_only=false, sort\_index=0 } = req.body

const {rows:[q]} = await pool.query(

`insert into booking\_questions (owner\_id,label,type,options,placeholder,help\_text,required,collect\_during\_booking,internal\_only,sort\_index)

values ($1,$2,$3,$4,$5,$6,$7,$8,$9,$10) returning \*`,

[ownerId,label,type,options,placeholder,help\_text,required,collect\_during\_booking,internal\_only,sort\_index]

)

res.status(201).json(q)

}catch(e){ next(e) }

})

r.post('/questions/:id', async (req,res,next)=>{

try{

const f = ['label','type','options','placeholder','help\_text','required','collect\_during\_booking','internal\_only','sort\_index']

const set=[], vals=[]

f.forEach((k,i)=>{ if(k in req.body){ set.push(`${k}=$${i+1}`); vals.push(req.body[k]) }})

vals.push(req.params.id, ownerId)

await pool.query(`update booking\_questions set ${set.join(',')} where id=$${vals.length-1} and owner\_id=$${vals.length}`, vals)

res.json({ok:true})

}catch(e){ next(e) }

})

export default r

Mount:

import bookingEngine from './routes/bookingEngine.js'

app.use('/api/booking-engine', bookingEngine)

# **2) Client UI**

## **a) Payments page — /client/src/pages/Payments.jsx**

import { useEffect, useState } from 'react'

import axios from 'axios'

const DEFAULT\_METHODS = [

{kind:'card', label:'Credit Card', auto\_pay:true, enabled:true, sort\_index:1},

{kind:'check', label:'Check', enabled:false, sort\_index:2},

{kind:'cash', label:'Cash', enabled:false, sort\_index:3},

{kind:'venmo', label:'Venmo', enabled:true, sort\_index:4, config:{handle:'@yourhandle'}},

{kind:'zelle', label:'Zelle', enabled:false, sort\_index:5},

{kind:'cashapp',label:'Cashapp', enabled:true, sort\_index:6, config:{tag:'$yourtag'}},

]

export default function Payments(){

const [processors,setProcessors]=useState([])

const [methods,setMethods]=useState([])

const [settings,setSettings]=useState(null)

const [plans,setPlans]=useState([])

useEffect(()=>{

Promise.all([

axios.get('/api/payments/processors'),

axios.get('/api/payments/methods'),

axios.get('/api/payments/settings'),

axios.get('/api/payments/plans')

]).then(([p,m,s,pl])=>{

setProcessors(p.data); setMethods(m.data); setSettings(s.data); setPlans(pl.data)

if(!m.data.length){ // seed defaults first time

DEFAULT\_METHODS.forEach(x=>axios.post('/api/payments/methods', x).then(()=>axios.get('/api/payments/methods').then(r=>setMethods(r.data))))

}

if(!pl.data.length){

seedPlans()

}

})

},[])

function seedPlans(){

const seeds = [

{name:'Retainer + final 30 days', slug:'retainer\_final\_30', spec:{type:'retainer\_final', final\_due\_days:30}, sort\_index:1},

{name:'Retainer + monthly split + final 30 days', slug:'retainer\_monthly\_split', spec:{type:'retainer\_monthly', months:3, final\_due\_days:30}, sort\_index:2},

{name:'Pay in full', slug:'pay\_in\_full', spec:{type:'pay\_in\_full'}, sort\_index:3}

]

seeds.forEach(x=>axios.post('/api/payments/plans', x).then(()=>axios.get('/api/payments/plans').then(r=>setPlans(r.data))))

}

async function toggleMethod(m, patch){ await axios.post(`/api/payments/methods/${m.id}`, patch); refreshMethods() }

async function refreshMethods(){ const {data}=await axios.get('/api/payments/methods'); setMethods(data) }

async function updateSettings(p){ await axios.post('/api/payments/settings', p); const {data}=await axios.get('/api/payments/settings'); setSettings(data) }

return (

<div className="grid grid-cols-3 gap-6">

<section className="col-span-2 space-y-6">

<div className="bg-white border rounded-2xl p-4">

<h2 className="font-display text-xl mb-2">Payment Processing</h2>

<div className="grid grid-cols-2 gap-4">

{['paypal','square'].map(k=>{

const p = processors.find(x=>x.kind===k)

const connected = p?.status==='connected'

return (

<div key={k} className="border rounded-xl p-3">

<div className="font-medium capitalize">{k}</div>

<div className="text-sm text-neutral-500">{connected?'Connected':'Connect to accept payments'}</div>

<div className="mt-2 flex gap-2">

{!connected && <button onClick={()=>axios.post(`/api/payments/processors/${k}/connect`).then(()=>axios.get('/api/payments/processors').then(r=>setProcessors(r.data)))} className="px-3 py-2 rounded-lg text-white" style={{background:'#3c0b43'}}>Connect</button>}

{connected && <button onClick={()=>axios.post(`/api/payments/processors/${k}/disconnect`).then(()=>axios.get('/api/payments/processors').then(r=>setProcessors(r.data)))} className="px-3 py-2 rounded-lg border">Disconnect</button>}

</div>

</div>

)

})}

</div>

</div>

<div className="bg-white border rounded-2xl p-4">

<h3 className="font-medium mb-2">Additional Payment Settings</h3>

<div className="space-y-3">

<div className="border rounded-xl p-3">

<div className="font-semibold">Due Dates</div>

<div className="text-sm">Initial payment label: <code>{settings?.deposit\_term || 'retainer'}</code></div>

<div className="grid grid-cols-3 gap-2 mt-2">

<input className="input" placeholder="Deposit term" defaultValue={settings?.deposit\_term || 'retainer'} onBlur={e=>updateSettings({deposit\_term:e.target.value})}/>

<input className="input" type="number" placeholder="Deposit due days (0=now)" defaultValue={settings?.deposit\_due\_days ?? 0} onBlur={e=>updateSettings({deposit\_due\_days:Number(e.target.value)})}/>

<input className="input" type="number" placeholder="Final due days" defaultValue={settings?.final\_due\_days ?? 30} onBlur={e=>updateSettings({final\_due\_days:Number(e.target.value)})}/>

</div>

</div>

<div className="border rounded-xl p-3">

<div className="font-semibold">Payment Plans</div>

<ul className="text-sm mt-2">{plans.map(p=><li key={p.id}>• {p.name}</li>)}</ul>

</div>

<div className="border rounded-xl p-3">

<div className="font-semibold">Tips</div>

<label className="inline-flex items-center gap-2 mt-1">

<input type="checkbox" defaultChecked={settings?.tips\_enabled} onChange={e=>updateSettings({tips\_enabled:e.target.checked})}/>

<span>Tips are currently {settings?.tips\_enabled?'enabled':'disabled'}</span>

</label>

</div>

</div>

</div>

</section>

<aside className="col-span-1">

<div className="bg-white border rounded-2xl p-4">

<div className="font-medium mb-2">Customer Payment Options</div>

<div className="space-y-2">

{methods.sort((a,b)=>a.sort\_index-b.sort\_index).map(m=>(

<div key={m.id} className="border rounded-xl p-2 flex items-center justify-between">

<div>

<div className="font-semibold">{m.label}</div>

<div className="text-xs text-neutral-500">{m.kind}{m.auto\_pay?' • Auto Pay':''}</div>

</div>

<div className="flex gap-2">

<button className={`px-2 py-1 rounded ${m.enabled?'bg-emerald-500 text-white':'border'}`} onClick={()=>toggleMethod(m,{enabled:!m.enabled})}>{m.enabled?'Enabled':'Disabled'}</button>

<button className="px-2 py-1 rounded border" onClick={()=>toggleMethod(m,{show\_on\_invoices:!m.show\_on\_invoices})}>{m.show\_on\_invoices?'Shown':'Hidden'}</button>

</div>

</div>

))}

</div>

</div>

</aside>

</div>

)

}

Route: <Route path="/settings/payments" element={<Payments/>} />

## **b) Booking Engine page with tabs — /client/src/pages/BookingEngine.jsx**

import { useEffect, useState } from 'react'

import axios from 'axios'

function Section({title,children}){ return <div className="bg-white border rounded-2xl p-4 mb-6"><h3 className="font-medium mb-2">{title}</h3>{children}</div> }

export default function BookingEngine(){

const [tab,setTab]=useState('General')

return (

<div>

<h1 className="text-2xl font-display mb-4">Booking Engine</h1>

<div className="flex gap-2 mb-4">

{['General','Extra Questions','Unavailable Notices','Privacy'].map(t=>

<button key={t} onClick={()=>setTab(t)} className={`px-3 py-2 rounded-lg ${tab===t?'text-white':'border'}`} style={tab===t?{background:'#3c0b43'}:{}}>

{t}

</button>

)}

</div>

{tab==='General' && <General/>}

{tab==='Extra Questions' && <ExtraQuestions/>}

{tab==='Unavailable Notices' && <Notices/>}

{tab==='Privacy' && <Privacy/>}

</div>

)

}

function General(){

const [g,setG]=useState(null)

useEffect(()=>{ axios.get('/api/booking-engine/general').then(r=>setG(r.data||{})) },[])

async function save(p){ await axios.post('/api/booking-engine/general', p); const {data}=await axios.get('/api/booking-engine/general'); setG(data) }

if(!g) return null

return (

<>

<Section title="General">

<input className="input mb-2" placeholder="Header Text" defaultValue={g.header\_text} onBlur={e=>save({header\_text:e.target.value})}/>

<input className="input mb-2" placeholder="Subheader Text" defaultValue={g.subheader\_text} onBlur={e=>save({subheader\_text:e.target.value})}/>

<input className="input" placeholder="Call-to-Action Text" defaultValue={g.cta\_text || 'Choose Now'} onBlur={e=>save({cta\_text:e.target.value})}/>

</Section>

<Section title="Multiple Packages">

<label className="flex items-center gap-2">

<input type="radio" name="mp" defaultChecked={!g.allow\_multiple\_packages} onChange={()=>save({allow\_multiple\_packages:false})}/> Only pick a single package

</label>

<label className="flex items-center gap-2 mt-2">

<input type="radio" name="mp" defaultChecked={!!g.allow\_multiple\_packages} onChange={()=>save({allow\_multiple\_packages:true})}/> Add additional packages with multiple dates & times

</label>

</Section>

<Section title="Additional Options">

<label className="flex items-center gap-2">

<input type="checkbox" defaultChecked={g.enable\_captcha} onChange={e=>save({enable\_captcha:e.target.checked})}/> Enable Captcha

</label>

</Section>

</>

)

}

function ExtraQuestions(){

const [rows,setRows]=useState([])

useEffect(()=>{ axios.get('/api/booking-engine/questions').then(r=>setRows(r.data)) },[])

async function add(seed){

const {data}=await axios.post('/api/booking-engine/questions', seed)

setRows([...rows, data])

}

return (

<>

<Section title="Questions">

<button className="px-3 py-2 rounded-lg text-white mb-3" style={{background:'#3c0b43'}}

onClick={()=>add({label:'Customer Address', type:'group', required:true, help\_text:'Street / City / State / Postal Code'})}>+ New Question</button>

<div className="space-y-2">

{rows.map(q=>(

<div key={q.id} className="border rounded-xl p-3">

<div className="font-semibold">{q.label}</div>

<div className="text-xs text-neutral-500">{q.required?'Required':'Optional'} • {q.collect\_during\_booking?'Collect during booking':'Internal'}</div>

</div>

))}

</div>

</Section>

<Section title="Suggested Presets">

<div className="grid grid-cols-2 gap-2">

<Preset label="Stair Setup" onClick={()=>add({label:'Stair Setup', type:'select', options:[{v:'No stairs'},{v:'1 flight'},{v:'2+ flights'}], required:true})}/>

<Preset label="Estimated Guest Count" onClick={()=>add({label:'Estimated Guest Count', type:'text', placeholder:'e.g., 150', internal\_only:true})}/>

<Preset label="Insurance Requirements" onClick={()=>add({label:'Insurance Requirements', type:'textarea', help\_text:'Describe required insurance limits', internal\_only:true})}/>

<Preset label="Text Opt In" onClick={()=>add({label:'Text Opt In', type:'checkbox', options:[{v:'Yes'},{v:'No'}], help\_text:'Consent for SMS messaging', required:true})}/>

</div>

</Section>

</>

)

}

function Preset({label,onClick}){ return <button onClick={onClick} className="border rounded-xl p-3 text-left hover:bg-neutral-50">{label}</button> }

function Notices(){

const [rows,setRows]=useState([])

const [draft,setDraft]=useState('Sorry, we are fully booked for this date. Please choose another date or contact us.')

useEffect(()=>{ axios.get('/api/booking-engine/notices').then(r=>setRows(r.data)) },[])

async function add(){ const {data}=await axios.post('/api/booking-engine/notices',{category:'default', body:draft}); setRows([...rows, data]) }

async function save(id, body){ await axios.post(`/api/booking-engine/notices/${id}`, {body}); const {data}=await axios.get('/api/booking-engine/notices'); setRows(data) }

return (

<>

<Section title="Default Notice">

<textarea className="w-full border rounded-xl p-3 min-h-[120px]" value={draft} onChange={e=>setDraft(e.target.value)}/>

<button onClick={add} className="mt-2 px-3 py-2 rounded-lg text-white" style={{background:'#3c0b43'}}>Save</button>

</Section>

<Section title="Existing Notices">

{rows.map(n=>(

<div key={n.id} className="border rounded-xl p-3 mb-2">

<div className="text-xs uppercase text-neutral-500">{n.category}</div>

<InlineEdit value={n.body} onSave={(v)=>save(n.id,v)}/>

</div>

))}

</Section>

</>

)

}

function InlineEdit({value,onSave}){

const [v,setV]=useState(value)

return (

<div>

<textarea className="w-full border rounded-xl p-2" rows="4" value={v} onChange={e=>setV(e.target.value)}/>

<button className="mt-2 px-3 py-1 rounded-lg border" onClick={()=>onSave(v)}>Update</button>

</div>

)

}

function Privacy(){

const [p,setP]=useState(null)

useEffect(()=>{ axios.get('/api/booking-engine/privacy').then(r=>setP(r.data||{})) },[])

async function save(u){ await axios.post('/api/booking-engine/privacy', u); const {data}=await axios.get('/api/booking-engine/privacy'); setP(data) }

if(!p) return null

return (

<>

<Section title="About Your Privacy Settings">

<label className="flex items-center gap-2 mb-3">

<input type="radio" name="consent" defaultChecked={!!p.request\_consent} onChange={()=>save({request\_consent:true})}/> Yes, request consent on first step

</label>

<label className="flex items-center gap-2 mb-3">

<input type="radio" name="consent" defaultChecked={!p.request\_consent} onChange={()=>save({request\_consent:false})}/> No

</label>

<div className="text-sm text-neutral-500 mb-2">Consent message</div>

<textarea className="w-full border rounded-xl p-2 mb-3" rows="4" defaultValue={p.consent\_message} onBlur={e=>save({consent\_message:e.target.value})}/>

<div className="text-sm text-neutral-500 mb-1">URL of your Privacy Policy</div>

<input className="input" defaultValue={p.privacy\_policy\_url} onBlur={e=>save({privacy\_policy\_url:e.target.value})}/>

</Section>

</>

)

}

Route: <Route path="/settings/booking-engine" element={<BookingEngine/>} />

# **3) Seed helpers (optional)**

After migrating, you can quickly seed payment settings + defaults:

// /server/scripts/seedPayments.js (one-off)

import { pool } from '../index.js'

const ownerId = process.env.DEMO\_OWNER\_ID

await pool.query(`insert into payment\_settings (owner\_id,deposit\_term,deposit\_due\_days,final\_due\_days,tips\_enabled)

values ($1,'retainer',0,30,true) on conflict (owner\_id) do nothing`, [ownerId])

console.log('Seeded payment settings'); process.exit(0)

## **What you just gained**

* **Payments**: processors (Square/PayPal placeholders), customer payment options (credit card + manual methods), **due dates**, **payment plans**, **tips toggle**—mirroring your screenshots.
* **Booking Engine**: **General copy/CTA**, **multiple packages** toggle, **captcha** toggle, **extra questions** (with presets you showed: address, stairs, guest count, insurance, text opt-in), **unavailable notices**, and **privacy consent** (message + policy URL).

If you want the next block, send screens for: **Packages/Items**, **Proposal builder**, **Invoices**, and **Booking calendar rules**. I’ll translate them the same way into schema + routes + React so your MVP hits Check Cherry parity with Monday-like UX. 🌈👑