Building an auto repair shop management module with integrated parts and payments

The complete tech stack for a production auto repair shop management module requires five core integrations: PartsTech for parts procurement (gated partner API, free to integrate), NMI as the payment gateway backbone for both SignaPay and PCBancard, the NHTSA VIN decoder (free government API), QuickBooks Online for accounting sync, and a labor guide provider like MOTOR. This architecture fits cleanly into an existing React/TypeScript + Node.js/Express + PostgreSQL platform as a feature-flagged modular monolith. The dual pricing model (cash discount, not surcharging) is legal in all 50 states OrderPin +3 and avoids the regulatory minefield of credit card surcharging, which remains banned in Connecticut, Massachusetts, Maine, and California. Corepay

---

1. PartsTech is powerful but gated — partner registration required

PartsTech (acquired by OEC in February 2025 for undisclosed terms) Digital Commerce 360 aggregates **7+ million parts across 1,900+ manufacturer brands** Gem-car **from 225+ suppliers at 30,000+ locations**. OEC The platform connects to NAPA, AutoZone, Advance Auto Parts, Gem-car O'Reilly, WorldPac, Parts Authority, and hundreds of local distributors. It supports VIN lookup, license plate lookup, and Year/Make/Model vehicle identification PartsTech with real-time wholesale pricing and color-coded inventory availability. PartsTech

Two integration methods exist. The Punchout API (most popular among the 35+ integrated shop management systems) OEC embeds PartsTech's search UI inside your application via punchout.partstech.com. The SMS sends vehicle context, PartsTech handles the entire search/compare/select workflow, and selected parts data flows back to your repair order. The Full API at api-docs.partstech.com allows programmatic access where you build your own UI — but this documentation is **not publicly accessible** and requires partner registration.

Authentication uses a **username + API key** pair. Ratchet+Wrench Shop owners find credentials at My Account → Account → API section PartsTech (help article at partstech.my.site.com/help/s/article/How-do-I-find-my-API-Key). The API is free for SMS integration partners PartsTech — PartsTech's revenue model is supplier-funded. No public SDKs, Postman collections, or OpenAPI specs exist. The GitHub organization (github.com/partstech) has zero public repositories. github Rate limits and sandbox details are disclosed only during partner onboarding.

The recommended integration path:

1. Contact PartsTech to become an integration partner (support@partstech.com, 866-208-5193) PartsTech
2. Receive access to api-docs.partstech.com and test credentials

3. Implement the Punchout API first (fastest path — days, not weeks)
4. Optionally upgrade to Full API for deeper UX control

The data flow for the Punchout integration is straightforward: your app sends VIN/vehicle data → PartsTech UI opens (Counterman) in an iframe/modal → user searches, compares pricing across suppliers, and builds a cart (PartsTech) → on close, cart items (part number, description, price, supplier, quantity, delivery ETA) are returned to your app (Counterman) via postMessage/callback → you map these to repair order line items.

Key PartsTech URLs

Resource	URL
API documentation (partner-only)	`api-docs.partstech.com`
Punchout domain	`punchout.partstech.com`
Help center	`partstech.my.site.com/help/s/`
Integration partners page	`partstech.com/software/management-system-integrations/`
OEC parent company	`oeconnection.com/products/partstech/`

---

2. NMI is the gateway that powers both SignaPay and PCBancard

Neither SignaPay nor PCBancard publishes standalone public API documentation. However, both route through well-documented payment gateways — and **NMI (Network Merchants Inc.) is the critical common denominator**.

PCBancard uses NMI's white-label gateway directly, confirmed by their integration portal at `pcb.transactiongateway.com/merchants/resources/integration/integration_portal.php` (the `transactiongateway.com` domain is NMI infrastructure). This means NMI's complete API documentation applies to PCBancard merchants.

SignaPay offers three gateway options: PayHub+ (their proprietary gateway with RESTful APIs and tokenization), Authorize.net, and NMI. (SignaPay) Their patented PayLo dual pricing technology is firmware-level software on Dejavoo Z9/Z11 and Pax S80 terminals, compatible only with the PayHub+ gateway for online transactions.

NMI API integration (applies to both SignaPay NMI accounts and PCBancard)

NMI provides the most flexible integration path with full developer documentation (NMI) at docs.nmi.com. The key endpoints and methods:

- **Payment API (Direct Post):** POST https://secure.nmi.com/api/transact.php (production) or POST https://sandbox.nmi.com/api/transact.php (sandbox). Vercel Authentication via security key. NMI Parameters sent as key-value pairs: security_key, type=sale, amount, ccnumber, ccexp, cvv.
- **Collect.js:** JavaScript tokenization library that creates hosted payment fields within your UI. Card data never touches your server NMI — **SAQ A-EP eligible**, dramatically reducing PCI scope. This is the recommended approach for a SaaS platform.
- **Customer Vault:** PCI-compliant card-on-file storage. Nmi Store a vault ID in your database; charge against it for future transactions without handling card data. nmi
- **Webhooks:** Near-real-time notifications for transactions, settlements, chargebacks, and automatic card updates. NMI
- **Surcharge configuration:** Set via merchant portal under Settings → Customer Fee Configuration. The gateway auto-calculates surcharge amounts when processing. NMI Fields include amount (base) and surcharge (fee). Max **3%** for credit; 3dmerchant auto-hidden on debit cards.

For dual pricing specifically, NMI supports cash/ACH discounting under Settings → Transaction Options. The gateway can apply a service fee to card transactions and waive it for electronic check/ACH payments. NMI Partners can configure fees on behalf of merchants and lock merchant-level access. NMI

Authorize.net as alternative

Authorize.net's API uses XML with JSON translation at api.authorize.net/xml/v1/request.api (production) Authorize.net and apitest.authorize.net/xml/v1/request.api (sandbox). Key integration methods include Accept.js (client-side tokenization), Accept Hosted (iFrame checkout), and Customer Information Manager for card-on-file. Authorize.net Node.js SDK available via github.com/AuthorizeNet. GitHub Gateway-only pricing: **$0 setup, $25/month, $0.10/transaction**. Kurv

Embedded payments architecture recommendation

For a SaaS platform, use **Collect.js (NMI) or Accept.js (Authorize.net)** for client-side tokenization. Cardconnect This creates hosted input fields styled to match your UI while keeping card data off your servers entirely. Lember The flow: customer enters card details in hosted fields → JavaScript library tokenizes and returns a token → your backend sends the token + amount to the gateway API → gateway processes and returns confirmation. Nmi Store the vault/token ID for card-on-file recurring charges. This achieves **SAQ A-EP PCI compliance** — the lightest burden for a software platform handling payments.

---

3. The database schema and workflow engine that powers a shop

Auto repair software revolves around a single central document: the **Repair Order (RO)**. Every competitor — Shop-Ware, Tekmetric, AutoLeap, Mitchell 1 Manager SE, Shop Boss — structures workflow around the RO lifecycle. The estimate, work order, and invoice are not separate entities but **status transitions of the same RO**.

Core PostgreSQL schema

```
sql
```

```
-- Unified work order with status-driven lifecycle
CREATE TYPE ro_status AS ENUM (
  'estimate','sent','approved','in_progress',
  'waiting_parts','completed','invoiced','paid','cancelled'
);

CREATE TABLE work_orders (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  tenant_id UUID NOT NULL REFERENCES tenants(id),
  ro_number SERIAL,
  customer_id UUID NOT NULL REFERENCES customers(id),
  vehicle_id UUID NOT NULL REFERENCES vehicles(id),
  assigned_tech_id UUID REFERENCES employees(id),
  bay_id UUID REFERENCES bays(id),
  status ro_status DEFAULT 'estimate',
  mileage_in INTEGER,
  promised_date TIMESTAMPTZ,
  customer_concern TEXT,
  subtotal_parts DECIMAL(10,2),
  subtotal_labor DECIMAL(10,2),
  tax_amount DECIMAL(10,2),
  total_card_price DECIMAL(10,2),
  total_cash_price DECIMAL(10,2),
  cash_discount_pct DECIMAL(4,2) DEFAULT 4.00,
  qbo_invoice_id VARCHAR(50),
  created_at TIMESTAMPTZ DEFAULT NOW()
);

CREATE TABLE service_lines (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  work_order_id UUID NOT NULL REFERENCES work_orders(id),
  line_type VARCHAR(20) CHECK (line_type IN ('labor','part','fee','sublet','discount')),
  description TEXT NOT NULL,
  quantity DECIMAL(10,2) DEFAULT 1,
  unit_cost DECIMAL(10,2) DEFAULT 0,    -- wholesale/cost
  unit_price DECIMAL(10,2) NOT NULL,    -- card price (posted price)
  labor_hours DECIMAL(5,2),
  part_number VARCHAR(100),
  supplier VARCHAR(100),
  partstech_order_id VARCHAR(100),
  approval_status VARCHAR(20) DEFAULT 'pending',  -- pending/approved/declined/deferred
  sort_order INTEGER DEFAULT 0
);
```

Additional required tables include `customers` (contact info, QBO sync ID), `vehicles` (VIN, YMM, license plate, mileage history), `employees` (role, labor rate, certifications), `bays` (number, status, current RO assignment), `payments` (amount, method, gateway transaction ID, QBO payment ID), `inspections` (DVI with color-coded condition points and photo URLs), and `appointments` (scheduling with bay/tech assignment).

Multi-tenancy pattern: Shared database with `tenant_id` on every table `Clerk` `Acropolium` plus PostgreSQL Row Level Security policies. Feature flag table (`tenant_features`) gates the auto repair module per tenant.

The 16-step estimate-to-payment workflow

Every modern shop management system follows this sequence, derived from analyzing Shop-Ware, Tekmetric, AutoLeap, and Mitchell 1:

1. **Customer arrives** (appointment or walk-in) → look up or create customer record
2. **Vehicle identification** → scan VIN, enter plate (→ plate-to-VIN API → NHTSA decode), or select YMM manually
3. **Create RO** (status: `estimate`) → document customer concerns, record mileage
4. **Digital Vehicle Inspection** → technician uses tablet to inspect 30-50+ points, marking each green/yellow/red with photos
5. **Build estimate** → add labor from MOTOR guide (hours × rate), add parts via PartsTech (real-time pricing), add fees/shop supplies
6. **Calculate dual pricing** → card price = posted price; cash price = card price minus discount percentage
7. **Send to customer** → text/email with DVI report + estimate showing both card and cash totals
8. **Customer approves** → individual line items approved/declined/deferred digitally; timestamp captured
9. **Convert to work order** (status: `approved` → `in_progress`) → approved items become active
10. **Order parts** → submit via PartsTech to suppliers; track order status
11. **Assign tech and bay** → dispatch via job board/tech board
12. **Technician completes work** → updates line item status; logs time
13. **Quality check** → service advisor reviews completed work
14. **Generate invoice** (status: `invoiced`) → finalize all line items
15. **Collect payment** → process via NMI/Authorize.net; apply card price or cash discount

16. **Sync to QuickBooks** → create invoice + payment in QBO; follow up with customer

Vehicle identification: free NHTSA API plus commercial plate lookup

The NHTSA vPIC API is free, requires no authentication, and returns **130+ attributes per VIN**:

```
GET https://vpic.nhtsa.dot.gov/api/vehicles/decodevinvalues/{vin}?format=json
```

Response includes Make, Model, ModelYear, BodyClass, EngineCylinders, EngineHP, DisplacementL, DriveType, TransmissionStyle, FuelTypePrimary, Trim, and more. PyPI Batch decoding is available via POST to `/vehicles/DecodeVINValuesBatch/`. NHTSA For license plate → VIN conversion, commercial APIs like **Plate2VIN** or **CarsXE** ($99/month + per-call) are required — no free government plate lookup exists.

Labor guide integration

MOTOR (motor.com, part of Hearst) provides estimated work times via a RESTful DaaS (Data as a Service) API covering 450+ procedures. MOTOR Times are indexed by Year/Make/Model/Engine using ACES vehicle standards. MOTOR Time types include standard flat rate (for non-factory-trained techs), warranty time, and optional overlap labor. ALLDATA **Mitchell 1 ProDemand** offers similar data plus repair information, diagrams, and OEM parts pricing, Mitchell 1® integrated via a TAPE (Transfer Application Public Extension) token workflow. Mitchell 1®

Standard industry practice uses **flat rate billing**: customers pay the labor guide's listed hours regardless of actual time spent. autoGMS Blog A brake job listed at 1.5 hours × $150/hr = $225 labor, even if the technician completes it in 45 minutes. autoGMS Blog Shops typically set rates between **$100-$200/hour** depending on market, with matrix pricing varying by job category (diagnostic, mechanical, electrical).

---

4. Dual pricing compliance: cash discount is the safe nationwide path

Cash discounts are legal in all 50 states Durangomerchantservices SignaPay under the Durbin Amendment (Dodd-Frank Act, 2010). TrueMerchant +2 Surcharging is banned in Connecticut, Massachusetts, Maine, and California, Getvms and restricted with caps or special requirements in Colorado Corepay (2% cap), New York (must show total card price per item), New Jersey, Nevada, Corepay Minnesota, and Oklahoma. The practical implication is clear: **structure the program as a cash discount, never as a surcharge**.

The critical legal test: if a customer pays **more** than the advertised/sticker price, it's a surcharge regardless of labeling. If the customer pays the sticker price **or less**, it's a cash discount. Card Fellow +2 Visa caps surcharges at **3%** Corepay (reduced from 4% in April 2023), 3dmerchant requires 30-day advance registration, 3dmerchant Zendesk and prohibits surcharging debit/prepaid cards. Corepay Cash discounts

require **no Visa registration, have no percentage cap, and face no card-type restrictions**.
Durangomerchantservices Zendesk

State configuration for software compliance

```sql
CREATE TABLE dual_pricing_state_rules (
  state_code CHAR(2) PRIMARY KEY,
  surcharge_allowed BOOLEAN NOT NULL,
  surcharge_cap DECIMAL(4,2),          -- NULL if no cap or banned
  cash_discount_allowed BOOLEAN DEFAULT TRUE,
  cash_discount_cap DECIMAL(4,2),      -- Only Wyoming: 5%
  required_disclosure TEXT,
  special_rules JSONB DEFAULT '{}'
);
```

Key state entries: Connecticut (surcharge banned, Getvms cash discount allowed OrderPin with conspicuous posting), Massachusetts (surcharge banned, OrderPin Getvms cash discount allowed if offered to all buyers), nfib California (SB 478 Getvms "Honest Pricing Law" bans add-on fees above advertised price, cash discounts explicitly allowed), nfib RateRemover Colorado (surcharge capped at 2% 3dmerchant with specific statutory notice language required), Colorado Attorneys New York (total card price must be posted per item, surcharge cannot appear as separate line item), LawPay nfib and Wyoming (only state with a cash discount cap at 5%). PaymentsJournal Merchant Cost Consulting

Compliant estimate/invoice format

The posted price must always be the **card price**. POS Nation +2 A compliant dual-price estimate shows:

```
ESTIMATE #1234 | 2021 Ford Mustang GT | VIN: 1FA6P8CF...

Service/Part                  List Price    Cash Price
─────────────────────────────────────────────────────────────
Oil Change                     $52.00        $50.00
Front Brake Pads (Centric)     $156.00       $150.00
Labor - Brake Replacement (1.5h) $208.00     $200.00
─────────────────────────────────────────────────────────────
Subtotal                      $416.00       $400.00
Tax (8%)                       $33.28        $32.00
CARD TOTAL                         $449.28
CASH TOTAL                                  $432.00

We offer a ~4% cash discount off the list price for cash payments.
```

Software must generate compliant signage Aurora Payments for store entrance and point of sale, PayWavez +2 calculate tax on the post-discount amount for cash payments (confirmed in 16+ states), handle split payments (partial cash + partial card), and exclude tips from the non-cash adjustment. Shiftprocessing **Never use the terms "service fee," "processing fee," or "surcharge"** — always frame as "cash discount" or "cash price." Durangomerchantservices Aurora Payments

---

5. Module architecture fits a modular monolith on Replit

The optimal architecture for adding auto repair shop management as a vertical module inside an existing SaaS platform is a **modular monolith with feature flags** — not microservices. This avoids the operational overhead of distributed systems while maintaining clean boundaries between the auto repair module and the parent platform.

Architecture pattern

The auto repair module shares authentication, billing, user management, and core infrastructure with the parent platform. A feature flag (`auto_repair_module`) gates access per tenant. The module owns its own database tables (prefixed or in a logical schema), API routes (`/api/auto-repair/*`), and frontend routes (`/shop/*`). All auto repair tables include `tenant_id` with PostgreSQL Row Level Security for isolation.

```typescript
// Feature flag middleware (Express)
const requireFeature = (featureKey: string) => async (req, res, next) => {
  const feature = await db.query(
    'SELECT enabled FROM tenant_features WHERE tenant_id = $1 AND feature_key = $2',
    [req.tenantId, featureKey]
  );
  if (!feature?.enabled) return res.status(403).json({ error: 'Feature not available' });
  next();
};
app.use('/api/auto-repair/*', requireFeature('auto_repair_module'));
```

Recommended tech stack for Replit deployment

- **Frontend:** React 19 + Vite + TypeScript + TailwindCSS + Shadcn/UI. Use React Hook Form + Zod for the estimate builder, Zustand for complex multi-step workflow state, and Socket.IO client for real-time bay/work order updates.
- **Backend:** Node.js 20 + Express 4 + TypeScript. Drizzle ORM (first-class Replit support with built-in Drizzle Studio). Circuit breaker via `opossum` for all third-party API calls.

- **Database:** Replit's built-in PostgreSQL 16 (powered by Neon). Serverless, sleeps after 5 minutes of inactivity, 10GB storage. Connection string via `DATABASE_URL` env var.
- **Real-time:** Socket.IO with tenant-scoped rooms (`tenant:${tenantId}`) for bay status and work order updates.
- **Background jobs:** In-process event-driven queue (no Redis available on Replit by default). For production scale, connect to Upstash Redis free tier.
- **File storage:** Cloudflare R2 or S3 for DVI inspection photos (no persistent local filesystem on Replit deployments).

QuickBooks Online integration

QBO uses OAuth 2.0 [Apideck] [GetKnit] with the `intuit-oauth` npm package. Authorization URL: `appcenter.intuit.com/connect/oauth2`. [Celigo] Token exchange: `oauth.platform.intuit.com/oauth2/v1/tokens/bearer`. [Celigo] Access tokens expire [Coefficient] hourly (auto-refresh required); refresh tokens last 100 days. [DEV Community] Sandbox at `sandbox-quickbooks.api.intuit.com/v3/company/{realmId}/`.

The sync pattern: create/match customers → map service line items to QBO Items (pre-create "Mechanical Labor," "Parts," "Shop Supplies" items) → create Invoice when RO reaches `invoiced` status → record Payment when collected → handle webhooks for bidirectional updates. Store `qbo_customer_id`, `qbo_invoice_id`, and `qbo_payment_id` as foreign keys in your tables.

Third-party integration resilience layer

Every external API call (PartsTech, NMI, NHTSA, QBO, MOTOR) should pass through a shared integration gateway with circuit breaking, exponential backoff retry, and TTL-based caching. Cache NHTSA VIN decodes for 24 hours (data rarely changes). Cache parts catalog searches for 5 minutes (prices fluctuate). Use idempotency keys for payment processing retries.

```typescript
// Circuit breaker for all third-party calls
const partsBreaker = new CircuitBreaker(fetchParts, {
  timeout: 10000,
  errorThresholdPercentage: 50,
  resetTimeout: 30000,
});
```

API route structure

```
/api/auto-repair/
  /customers        CRUD + search + QBO sync
  /vehicles          CRUD + VIN decode + plate lookup
  /work-orders       Full lifecycle (estimate → invoice → paid)
  /work-orders/:id/lines    Service line items (parts + labor)
  /work-orders/:id/dvi      Digital vehicle inspection
  /work-orders/:id/approve  Customer approval endpoint
  /work-orders/:id/payment  Process payment via gateway
  /bays              Status management + real-time updates
  /parts/search      PartsTech integration launcher
  /parts/import      Import cart items from PartsTech
  /reports           Revenue, car count, ARO, tech productivity
/api/integrations/
  /qbo/connect       OAuth2 initiation
  /qbo/callback      Token exchange
  /qbo/webhook       Webhook receiver
  /partstech/config  Credential management
```

Conclusion: a clear build path with manageable complexity

The technical research reveals a well-trodden integration landscape. **PartsTech's Punchout API is the fastest path to parts ordering** — expect days of development, not weeks — though Full API access requires becoming a registered partner. **NMI is the single gateway API to learn** since it powers both PCBancard directly and SignaPay's NMI-routed accounts, with Collect.js providing the cleanest embedded payments path NMI at SAQ A-EP compliance. The **cash discount model eliminates 90% of dual pricing compliance headaches** by being legal nationwide without card network registration. SignaPay +3

Three decisions will most impact development velocity. First, start with the PartsTech Punchout embed rather than building a custom parts search UI — this is what 35+ competing platforms do. PartsTech +2 Second, use NMI's Collect.js for payment tokenization to avoid PCI scope entirely on your servers. NMI Third, model the repair order as a single entity with status transitions rather than separate estimate/work order/invoice tables — this matches how every successful competitor structures their data and dramatically simplifies the workflow engine. The entire module can ship as a feature-flagged addition to the existing platform without architectural disruption, sharing auth, billing, and infrastructure while owning its domain-specific tables and routes.