**Core pages (required)**

1. **Landing / Home**
   * Route: /
   * Purpose: Short intro + links to login and demo.
   * Key parts: Hero, quick links (Transactions, Login), public info.
   * Auth: Public
2. **Login (Auth)**
   * Route: /login
   * Purpose: User sign-in (returns JWT).
   * Key parts: Email, password, submit, error messages.
   * Calls: POST /auth/login (or your auth endpoint)
   * Auth: Public
3. **Transactions Overview (Dashboard)**
   * Route: /transactions or /dashboard
   * Purpose: Main table listing all transactions, with filters, sort, pagination.
   * Key parts:
     + Table columns: collect\_id, school\_id, gateway, order\_amount, transaction\_amount, status, custom\_order\_id
     + Filters: status (multi), school (multi), date range, search box
     + Sorting headers (payment\_time, transaction\_amount, status)
     + Pagination controls (page, limit)
     + Persist filters/sort/pagination in URL (query params)
     + Row hover effect and quick actions (view details)
   * Calls: GET /transactions?limit=&page=&sort=&order=&status=&schoolId=&from=&to=
   * Auth: Protected
4. **Transaction Details**
   * Route: /transactions/:collectId or /transaction/:custom\_order\_id
   * Purpose: Show full order + order\_status timeline, webhook updates, metadata.
   * Key parts: order info, payment details, history, raw webhook logs, retry button (admin).
   * Calls: GET /transactions (single via query or dedicated endpoint like GET /transaction-status/:custom\_order\_id)
   * Auth: Protected
5. **Create Payment (Payment Form)**
   * Route: /create-payment
   * Purpose: Accept payment details, call backend to create collect request, and redirect user to payment page returned by API.
   * Key parts: form fields (school\_id, student info, amount, custom\_order\_id), validation, submit → show loader → redirect to payment link.
   * Calls: POST /create-payment
   * Auth: Protected (or public depending on flow)
6. **School Transactions (By School)**
   * Route: /schools (list) and /schools/:schoolId/transactions
   * Purpose: Select a school and view that school's transactions only.
   * Key parts: school selector/search, table of transactions (same columns), filters specific to school.
   * Calls: GET /transactions/school/:schoolId
   * Auth: Protected
7. **Transaction Status Check (Single)**
   * Route: /check-status
   * Purpose: Input custom\_order\_id to check current status.
   * Key parts: input, submit, display status/result.
   * Calls: GET /transaction-status/:custom\_order\_id
   * Auth: Protected (or public if allowed)

**Admin / Support pages (strongly recommended)**

1. **Webhook Logs / Webhook Tester**
   * Route: /webhooks (logs) and /webhooks/test
   * Purpose: View stored webhook events, inspect payloads, replay or mark processed. Tester page to simulate webhook payloads from Postman style UI.
   * Key parts: list of webhooks (timestamp, status), raw JSON viewer, filter/search, resubmit button.
   * Calls: GET /webhook-logs, POST /webhook/test (optional)
   * Auth: Admin only
2. **Users / Team Management**
   * Route: /users
   * Purpose: View/add/edit users (trustees, admins).
   * Key parts: user list, roles, add/edit forms.
   * Calls: backend user CRUD endpoints
   * Auth: Admin only
3. **Settings / Config**
   * Route: /settings
   * Purpose: App settings (PG credentials for demo, theme toggle, pagination defaults).
   * Key parts: theme (dark/light), page size default, webhook endpoint config (admin).
   * Calls: settings endpoints, or local state
   * Auth: Admin only
4. **Reports & Analytics**
   * Route: /reports
   * Purpose: Charts and KPIs (total volumes, success rate, daily totals).
   * Key parts: charts (bar/line/pie), date range, export CSV.
   * Calls: GET /transactions with aggregation params or dedicated report endpoints
   * Auth: Protected

**Utility & UX pages**

1. **Payment Callback / Redirect Handler**
   * Route: /payment/callback
   * Purpose: Landing page when payment gateway redirects user back. Read query/body params, show success/fail, then optionally call backend to confirm.
   * Key parts: status UI, spinner while verifying, link to transaction detail.
   * Calls: optionally GET /transaction-status/:custom\_order\_id or backend verify endpoint
   * Auth: Usually public (gateway will hit it)
2. **404 / Error pages**
   * Route: \* / /404
   * Purpose: Friendly not-found and error handling pages.
   * Auth: Public

**Optional / Extra-credit pages**

1. **Real-time Dashboard**
   * Route: /realtime
   * Purpose: Live charts using websockets / polling for new transactions and status changes.
   * Key parts: live counters, streaming table, refresh toggle.
   * Calls: websocket or GET /transactions polling
   * Auth: Protected
2. **API Docs / Help**
   * Route: /docs
   * Purpose: Show API examples, Postman collection, environment setup instructions for testers.
   * Key parts: curl examples, sample payloads (webhook), test links.
   * Auth: Protected or public (internal docs)

**Shared components (needed across pages)**

* Header (with user menu / logout)
* Sidebar / Nav (school selector, links)
* Transactions Table (reusable, supports sorting/paging/row actions)
* Filters Panel (status multi-select, date-range, school multi-select)
* Pagination component
* Modal (for details / confirm)
* Toast / Notifications
* Loader / Empty State component
* ProtectedRoute wrapper (enforces JWT)
* Color theme toggle (if implementing dark mode)
* CSV export / Download component

**Mapping pages → backend endpoints (quick)**

* Transactions page → GET /transactions (with page, limit, sort, filters)
* School transactions → GET /transactions/school/:schoolId
* Create payment → POST /create-payment
* Webhook receive/update → POST /webhook (backend receives)
* Transaction status check → GET /transaction-status/:custom\_order\_id
* Webhook logs → GET /webhook-logs

**Extra implementation notes (short)**

* Persist filters in URL so users can share views.
* Always protect routes with JWT; use Authorization: Bearer <token>.
* Implement client-side validation for forms (e.g., create payment).
* Make UI responsive for mobile and desktop.
* Use query-string or React Router hooks to sync filters + pagination with URL.
* Add indexes in DB for collect\_id, school\_id, custom\_order\_id to make the lists fast.