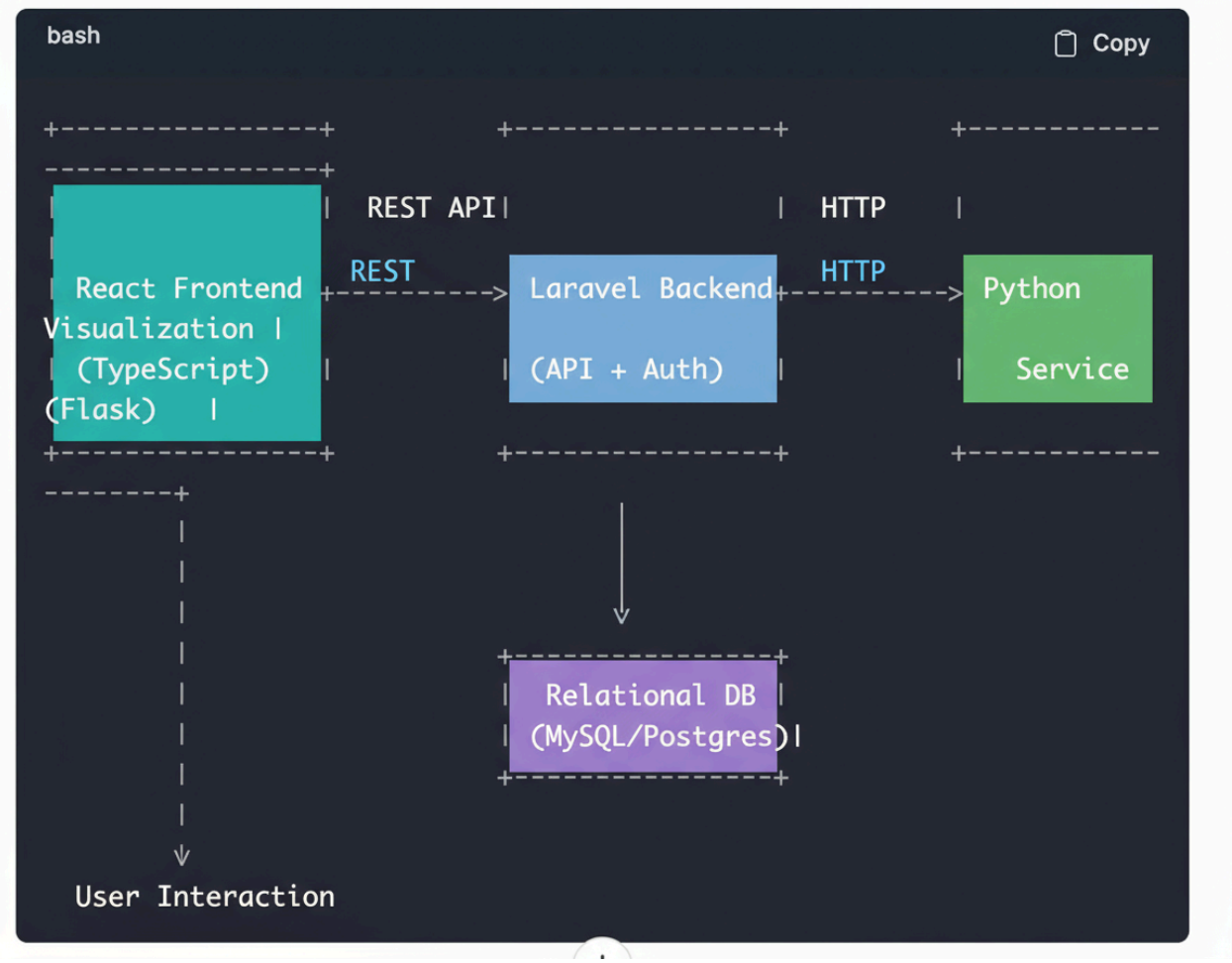
Personal Finance Application — Updated Technical Architecture Proposal

1. Overview

The Personal Finance Application is a multi-user platform designed to provide detailed tracking of financial transactions, support multiple accounts and currencies, allow CSV imports, present advanced financial visualizations, and facilitate collaborative financial management with role-based access control.

The software is designed to support the sale of licenses, accepting payments via credit card and PayPal, enabling subscription management with secure and compliant handling of license lifecycles.

2. Technology Stack



Backend: Laravel 12 RESTful API

Frontend: React with TypeScript

Data Visualization: Python with Seaborn (served via API or static images)

Database: MySQL or PostgreSQL

Authentication: JWT-based token authentication with role-based access control

Payment Gateway Integration: TBD (recommended gateways supporting credit card and PayPal like NOWPayments and PayPal)

3. Architecture Overview

3.1 Backend

Core Application (Existing)

- User Authentication & Authorization using JWT and middleware enforcing roles (owner, editor, viewer)
- Transaction CRUD with filters and categories
- CSV import handling with data validation
- Financial summary APIs (balances, trends, category breakdowns)
- Participant management (invite users, assign roles)

Payment Module

Backend Design for License and Subscription Management

Entities:

- **License**: Stores unique license keys, expiration dates, status, and user associations.
- **Subscription:** Tracks subscription periods with fields `user_id`, `start_date`, `end_date`, `status` (active, canceled, expired).
- **Payment Records:** Logs payment transactions (`transaction_id`, `user_id`, `amount`, `currency`, `payment_method`, `status`, timestamps).

Functionality:

- Update license and subscription statuses based on payment verification.
- APIs to retrieve, renew, revoke licenses.
- Webhook endpoints to asynchronously process payment notifications from payment gateways safely.
- Support for trials and promotional codes for flexible subscription offerings.

Payment Gateway Integration

Implement payment flow using gateway SDKs or APIs to:

- Create payment intents/orders or fixed price, with currency and method specification.
- Support redirect or inline checkout flows for credit card and PayPal.
- Handle webhook callbacks for payment success or failure with idempotency and retry logic.
- Optionally implement refund and cancellation processes.

License Issuance Workflow

- Automatically generate license keys (encrypted or token-wrapped) on confirmed payment.
- Store license linked to user and subscription with a 1-year expiration.
- Deliver license information through email and make it available on user dashboard.
- Support license management operations such as revocation and renewal.

Security & Compliance

- Continue enforcing JWT authentication and role-based access control for all payment-related endpoints.
- Follow PCI DSS best practices; no raw credit card data storage; use tokenization and vaulting from gateways.
- Secure webhook endpoints with secrets and HTTPS enforcement.
- Implement rate limiting, logging, and monitoring on payment APIs.

3.2 Frontend

Core Application

- React components for transaction input forms, lists with filters, CSV upload, dashboards with charts, user authentication UI.
- Axios/Fetch API integrated with JWT for secure communication.

Payment Module

Payment UI:

- “Buy License” interface on the user dashboard showing subscription price.
- Payment method selection between credit card and PayPal.
- Integrate hosted payment pages or embedded SDK-based checkouts to handle payment data securely (minimize PCI scope).
- Display real-time status of payment process: progress, success, failure messages.
- Post-purchase display of license key and transaction receipt within dashboard.
- Frontend validation and error handling for payment input forms.
- Use existing React Context or Redux for global payment and license state management.

4. API Endpoints (Summary)

Endpoint Category	Key APIs	Description
User Authentication	`/auth/register`, `/auth/login`, `/auth/logout`	Standard JWT-based auth flows
Transactions	`/transactions` (CRUD)	Create, read, update, delete transactions
CSV Import	`/transactions/import`	Upload and process bulk transaction data
License Management	`/licenses` (GET, POST, PUT)	Retrieve license info, renew, revoke
Subscription Management	`/subscriptions`	Manage subscription lifecycle
Payment Processing	`/payments/initiate`	Create payment intents/orders
	`/payments/webhook`	Secure webhook to receive payment status notifications
Participant Management	`/participants`, `/invitations`	Invite and manage collaborators

5. Security Considerations

- Secure JWT tokens stored as HttpOnly cookies or secure local storage.
- Input validation and sanitization across backend forms and endpoints.
- Protect webhook endpoints with secret signatures and HTTPS.
- Payment data handled via tokenized gateways, avoiding sensitive card data storage.
- Role-based access control enforced consistently on new endpoints.

6. Deployment and Operational Notes

Payment gateway credentials and webhook URLs managed securely in environment variables.

Use sandbox environments of payment providers during development.

Monitor payment transactions and subscription states for anomalies.

Backup license and subscription data regularly.

7. Summary

This integrated architecture proposal extends the robust Personal Finance Application platform with a secure, PCI-compliant payment module for software license sales. It integrates seamlessly within the Laravel REST backend and React frontend stack with JWT authentication, extending data models, API endpoints, and UI components while preserving the existing system’s collaborative and visualization capabilities.