NinexGroup Backend Documentation

Overview

- Stack: `Node.js` + `Express` + `MongoDB (Mongoose)`
- Entrypoint: `index.js`
- Config: `config/db.js`
- Auth: JWT (`x-auth-token` or `Authorization: Bearer`) and API Key (`x-api-key`)
- Background Jobs: Hourly settlement job (`jobs/settlementJob.js`)

Setup

1. Install dependencies: `npm install`
2. Configure environment: create `.env` with keys below
3. Run dev: `npm run dev`
4. Run prod: `npm start`

Environment Variables

Do not commit secrets. Required keys (values redacted):

- `MONGO_URI` � MongoDB connection string
- `JWT_SECRET` � JWT signing secret
- `FRONTEND_URL` � Frontend base URL for redirects
- `BACKEND_URL` � API base URL
- `CASHFREE_BASE_URL` , `CASHFREE_APP_ID` , `CASHFREE_SECRET_KEY` , `CASHFREE_PAYOUT_URL`
- `RAZORPAY_KEY_ID` , `RAZORPAY_KEY_SECRET` , `RAZORPAY_WEBHOOK_SECRET`

Authentication

- JWT (Merchants/Admin):
  - Header: `x-auth-token: <jwt>` OR `Authorization: Bearer <jwt>`
  - Middleware: `middleware/auth.js`
- API Key (Merchant API):
  - Header: `x-api-key: <merchant_api_key>`
  - Middleware: `middleware/apiKeyAuth.js`
- SuperAdmin:
  - Header: `x-auth-token: <jwt>` with role `superAdmin`
  - Middleware: `middleware/superAdminAuth.js`

Routes Summary

Base path: `BACKEND_URL`

Auth (`/api/auth`)

- POST `/signup` � create user (admin or superAdmin)
  - Body: `name, email, password, role?, businessName?, businessLogo?, businessDetails?`
  - Returns: `token` , `user`
- POST `/login` � login with email/password

    - Body: `email, password`
    - Returns: `token`, `user`
- GET `/profile` � get current user (JWT)
- PUT `/profile` � update profile (JWT)

Files: `routes/authRoutes.js`, `controllers/authController.js`

API Keys (`/api`)

All require JWT (`auth` middleware).

- POST `/create` � create API key
- GET `/get` � get API key
- DELETE `/delete` � delete API key
- POST `/regenerate` � regenerate API key

Files: `routes/apiRoutes.js`, `controllers/apiController.js`

Payments (Merchant) (`/api/payments`)

API Key auth unless noted.

- GET `/status/:orderId` � get payment status (x-api-key)
- GET `/transactions` � list transactions with filters (x-api-key)
    - Query: `page, limit, status, payment_gateway, payment_method, start_date, end_date, search, sort_by, sort_order`
    - Returns paginated list + summary

Merchant webhook configuration (JWT auth):

- POST `/merchant/webhook/configure` � set webhook URL and events
- GET `/merchant/webhook/config` � get current webhook config
- POST `/merchant/webhook/test` � send test webhook to merchant URL
- DELETE `/merchant/webhook` � delete webhook config

Files: `routes/paymentRoutes.js`, `controllers/paymentController.js`, `controllers/merchantWebhookController.js`

Razorpay (`/api/razorpay`)

- POST `/create-payment-link` � create Razorpay payment link (x-api-key)
    - Body: `amount, customer_name, customer_email, customer_phone, description?, callback_url?, success_url?, failure_url?`
    - Returns: `payment_link_id, payment_url, transaction_id`
- POST `/verify-payment` � verify a payment by `payment_link_id`
    - Body: `payment_link_id`
- POST `/webhook` � Razorpay webhook
    - Header: `x-razorpay-signature`
    - Uses `RAZORPAY_WEBHOOK_SECRET`

Files: `routes/razorpayRoutes.js`, `controllers/razorpayController.js`

Merchant Dashboard (JWT) (`/api/payments`)

- GET `/merchant/balance` � current balances and settlement breakdown
- GET `/merchant/payouts` � list my payout requests (filters supported)
- POST `/merchant/payout/request` � request payout for a settlement date
  - Body: `payoutDate (YYYY-MM-DD), transferMode, beneficiaryDetails, notes?`
- POST `/merchant/payout/:payoutId/cancel` � cancel payout request
- GET `/merchant/transactions/:transactionId` � get transaction details
- GET `/merchant/payout/:payoutId/status` � get payout status

Files: `routes/paymentRoutes.js`, `controllers/adminController.js`

SuperAdmin (JWT, role=superAdmin) (`/api/payments` and `/api/superadmin`)

- GET `/admin/transactions` � all transactions (filters supported)
- GET `/admin/payouts/all` � all payout requests
- POST `/admin/payout/:payoutId/approve` � approve payout
- POST `/admin/payout/:payoutId/reject` � reject payout
- POST `/admin/payout/:payoutId/process` � mark payout completed (set UTR)
- GET `/api/superadmin/dashboard/stats` � high-level stats

Files: `routes/paymentRoutes.js`, `routes/superAdminRoutes.js`, `controllers/superAdminController.js`

Models

- `models/User.js` � user/merchant with API key, business details, webhook config, payout settings.
- `models/Transaction.js` � transaction details incl. gateway fields, settlement fields, payout flags, indexes for queries.
- `models/Payout.js` � payout request lifecycle: requested ? approved/rejected ? processing ? completed/failed.
- `models/Payin.js` � simple payin record (not central in flows).

Background Jobs

- `jobs/settlementJob.js` runs hourly:
  - Applies 4PM cutoff and weekday rules via `utils/settlementCalculator.js`
  - Marks eligible transactions `settled` and `availableForPayout`

Utilities

- `utils/settlementCalculator.js` � expected settlement date, readiness checks, status messages.
- `utils/commissionCalculator.js` � payin and payout commission logic.

Common Response Shape

- Success: `{ success: true, ...data }`
- Error: `{ success: false, error: <message> }`

Headers Quick Reference

- JWT: `x-auth-token: <jwt>` or `Authorization: Bearer <jwt>`
- API Key: `x-api-key: <key>`
- Razorpay Webhook: `x-razorpay-signature: <sig>`
- Internal Merchant Webhooks (sent by this backend):

    - `x-webhook-signature`, `x-webhook-timestamp`, `x-merchant-id`, `x-event-type`

Example: Create Razorpay Payment Link

POST `${BACKEND_URL}/api/razorpay/create-payment-link` Headers: `x-api-key` Body:

```
{
  "amount": 499.00,
  "customer_name": "Jane Doe",
  "customer_email": "jane@example.com",
  "customer_phone": "9999999999",
  "description": "Order #1234",
  "callback_url": "https://merchant.example.com/payment/callback"
}
```

Response: `{ success, payment_link_id, payment_url, transaction_id, ... }`

Notes

- Some console logs contain stray characters from a previous copy; harmless but can be cleaned.
- Never expose `.env` values in client applications.