VENDOR JOURNEY OVERVIEW

This document outlines the workflow for a vendor using the Habrio backend. Each step lists the API endpoint, expected database interactions, and suggested frontend screens.

1. Authentication & Basic Onboarding

```
POST /api/v1/send-otp
```

```
POST /api/v1/verify-otp
```

```
POST /api/v1/onboarding/basic
```

- **DB writes**: `OTP` table insert, `UserProfile` row created/updated with role set to vendor and `basic_onboarding_done=True`.
- **Screens**: Login screen, OTP entry screen, onboarding form capturing name/city/society and role selection.

1. Vendor Profile Creation

```
POST /api/v1/vendor/profile
```

- Body requires `business_type`, `business_name`, and `address`.
- **DB writes**: `VendorProfile` inserted and linked to `UserProfile`. `role_onboarding_done` set true.
- **Screens**: Form for business details, progress indicator.

1. Document Upload & Payout Setup

```
POST /api/v1/vendor/upload-document
```

```
POST /api/v1/vendor/payout/setup
```

- **DB writes**: `VendorDocument` rows for each file. `VendorPayoutBank` row for bank details.
- **Screens**: Document upload section, bank account form.

1. Shop Management

```
POST /api/v1/vendor/shop
```

```
POST /api/v1/vendor/shop/edit
```

```
GET /api/v1/vendor/shop/my
```

```
POST /api/v1/vendor/shop/hours
```

```
POST /api/v1/vendor/shop/toggle-status
```

- **DB writes**: `Shop` entry for creation/edit, `ShopHours` entries, `ShopActionLog` when toggling status.
- **Screens**: New shop form, shop detail editor, hours editor (weekday/time pickers), switch to open/close shop.

1. Item Catalog Management

```
POST /api/v1/vendor/item/add
```

```
POST /api/v1/vendor/item/<id>/toggle
```

```
POST /api/v1/vendor/item/update/<id>
```

```
GET /api/v1/vendor/item/my
```

```
POST /api/v1/vendor/item/bulk-upload
```

- **DB writes**: `Item` table insert/update per operation.
- **Screens**: Item form, list with toggles, bulk upload page accepting CSV/XLS, edit item modal.

1. Wallet Management

```
GET /api/v1/vendor/wallet
```

```
POST /api/v1/vendor/wallet/credit
```

```
POST /api/v1/vendor/wallet/debit
```

```
POST /api/v1/vendor/wallet/withdraw
```

```
GET /api/v1/vendor/wallet/history
```

- **DB writes**: `VendorWallet` created if needed. `VendorWalletTransaction` row for each operation. Balance updated accordingly.
- **Screens**: Wallet overview, credit/debit/withdraw forms, transaction history table.

1. Handling Orders

```
GET /api/v1/vendor/orders
```

```
POST /api/v1/vendor/orders/<id>/status
```

POST /api/v1/vendor/orders/<id>/modify

POST /api/v1/vendor/orders/<id>/cancel

POST /api/v1/vendor/orders/<id>/message

GET /api/v1/vendor/orders/<id>/messages

GET /api/v1/vendor/orders/issues?order_id=<id>

POST /api/v1/vendor/orders/<id>/return/accept

POST /api/v1/vendor/orders/<id>/return/complete

POST /api/v1/vendor/orders/<id>/return/initiate

- **DB writes**: `OrderStatusLog`, `OrderActionLog`, `OrderMessage`, `OrderReturn` changes, wallet adjustments for refunds/credits.
- **Screens**: Order list with filters, order detail view with actions (accept/reject/deliver, modify items, cancel, chat), returns management screen.

1. Post-Delivery Earnings

[Order marked delivered] → system credits vendor wallet if payment was prep aid

- **DB writes**: `VendorWalletTransaction` credit, update `VendorWallet.balance`.
- **Screens**: Notification for earnings, wallet updated amount.

1. Summary of Data Flow

- Vendor accounts rely on `UserProfile` + `VendorProfile` for identity.
- Shop and items persist in `Shop` and `Item` tables.
- Orders contain `Order` and `OrderItem` records; vendor actions recorded in logs and messages.
- Wallet operations tracked in `VendorWallet` and `VendorWalletTransaction` along with payouts to bank.

1. Inventory Tracking & Stock Alerts

- **Screen**: Stock list with quantity indicators.
- Vendors poll `GET /api/v1/vendor/item/my` sorted by low quantity.
- Adjust stock via `POST /api/v1/vendor/item/update/<id>` sending new quantity.
- **DB writes**: updates on `Item.stock_qty` and timestamp; low-stock triggers background notifications.

```
GET /api/v1/vendor/item/my
```

```
POST /api/v1/vendor/item/update/<id>
```

1. Sales Analytics

- **Screen**: Charts summarizing daily orders and revenue.
- Data fetched using `GET /api/v1/vendor/analytics/sales?range=30d`.
- Backend aggregates `Order` data grouped by day.
- **DB reads**: computed from orders and order items; cached for performance.

```
GET /api/v1/vendor/analytics/sales?range=30d
```

1. Customer Messaging

- **Screen**: Unified chat inbox for all active orders.
- List uses `GET /api/v1/vendor/orders/messages/summary` to show unread counts.
- Opening a chat loads messages via `GET /api/v1/vendor/orders/<id>/messages` and new messages are sent with `POST /api/v1/vendor/orders/<id>/message`.
- **DB writes**: `OrderMessage` with sender role; read receipts update `read_at`.

```
GET /api/v1/vendor/orders/messages/summary
```

```
GET /api/v1/vendor/orders/<id>/messages
```

```
POST /api/v1/vendor/orders/<id>/message
```

1. Handling Returns

- **Screen**: Returns dashboard listing `GET /api/v1/vendor/orders/returns`.
- Vendor approves or rejects using `POST /api/v1/vendor/orders/<id>/return/accept` or `.../reject`.
- After receiving goods, vendor finalizes using `POST /api/v1/vendor/orders/<id>/return/complete`.
- **DB writes**: `OrderReturn` status transitions and wallet debits/credits when completed.

```
GET /api/v1/vendor/orders/returns
```

```
POST /api/v1/vendor/orders/<id>/return/accept
```

```
POST /api/v1/vendor/orders/<id>/return/reject
```

```
POST /api/v1/vendor/orders/<id>/return/complete
```

1. Managing Ratings & Feedback

- **Screen**: Table showing star ratings and comments from consumers.
- Data from `GET /api/v1/vendor/ratings` with pagination.
- Respond to feedback via `POST /api/v1/vendor/ratings/<id>/reply`.
- **DB writes**: `RatingReply` referencing the original rating.

```
GET /api/v1/vendor/ratings
```

```
POST /api/v1/vendor/ratings/<id>/reply
```

1. Payout and Settlement History

- **Screen**: History of wallet withdrawals and bank transfers.
- Fetched from `GET /api/v1/vendor/wallet/history` filtered by `type=payout`.
- Includes status of each payout (pending, complete, failed).
- **DB reads**: `VendorWalletTransaction` joined with payout bank info.

```
GET /api/v1/vendor/wallet/history
```

1. Staff Accounts

- **Screen**: Manage employee logins under main vendor account.
- Create staff via `POST /api/v1/vendor/staff/create` with permissions.
- List and revoke using `GET /api/v1/vendor/staff` and `POST /api/v1/vendor/staff/<id>/revoke`.
- **DB writes**: `VendorStaff` table with role flags and active status.

```
POST /api/v1/vendor/staff/create
```

```
GET /api/v1/vendor/staff
```

```
POST /api/v1/vendor/staff/<id>/revoke
```

1. Marketing & Promotion Tools

- **Screen**: Create discount campaigns and share links.
- Endpoint `POST /api/v1/vendor/campaign/create` with percentage and expiry.
- View running campaigns via `GET /api/v1/vendor/campaign/list`.
- **DB writes**: `PromotionCampaign` rows; redemption stats tracked per order.

```
POST /api/v1/vendor/campaign/create
```

```
GET /api/v1/vendor/campaign/list
```

1. Support & Dispute Resolution

- **Screen**: Submit support requests to platform admins.
- `POST /api/v1/vendor/support/ticket` opens a case, listing with `GET /api/v1/vendor/support/tickets`.
- Chat with support through `GET` and `POST` on `/ticket/<id>/message`.
- **DB writes**: `VendorSupportTicket` and `VendorSupportMessage` tables.

```
POST /api/v1/vendor/support/ticket
```

```
GET /api/v1/vendor/support/tickets
```

```
GET /api/v1/vendor/support/ticket/<id>
```

1. Account Settings & Security

- **Screen**: Change password, update contact phone and enable 2FA.
- Retrieve current settings via `GET /api/v1/vendor/settings`.

- Submit updates with `POST /api/v1/vendor/settings/update`.
- **DB writes**: `UserProfile` modifications and new `TwoFactorSecret` when enabled.

```
GET /api/v1/vendor/settings
```

```
POST /api/v1/vendor/settings/update
```

1. Logout & Session Management

- **Screen**: Option to logout this device or all staff devices.
- Current device logout uses `POST /api/v1/logout`.
- All-device logout uses `POST /api/v1/logout-all` to revoke tokens.
- User returned to phone login screen after completion.

```
POST /api/v1/logout
```

```
POST /api/v1/logout-all
```