CUSTOMER JOURNEY OVERVIEW

Authentication & Basic Onboarding

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

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

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

- **DB writes**: `OTP` created, `UserProfile` created/updated with phone, role, and `basic_onboarding_done=True`.
- **Screens**: Login (enter phone), OTP verification, Basic info form (name, city, society, role selection).

Consumer Specific Onboarding

```
POST /api/v1/consumer/onboarding
```

- **DB writes**: `ConsumerProfile` inserted with user_phone foreign key. `UserProfile.role_onboarding_done=True`.
- **Screens**: Profile screen to capture flat_number, profile photo, gender, birth date, preferred language.

1. Browsing Shops

```
GET /api/v1/consumer/shops
GET /api/v1/consumer/shops/search?q=...
```

- **DB reads**: `Shop` filtered by user's city/society and query params.
- **Screens**: Shop list with filters (status open/closed, type, tags), search bar.

1. Viewing Items in a Shop

```
GET /api/v1/shop/<shop_id>/items
```

- **DB reads**: `Item` table where `shop_id` matches and `is_available=True`.
- **Screens**: Shop details with item grid/list, item images and price.

1. Cart Management

```
POST /api/v1/consumer/cart/add
```

```
GET /api/v1/consumer/cart/view
```

```
POST /api/v1/consumer/cart/update
```

```
POST /api/v1/consumer/cart/remove
```

```
POST /api/v1/consumer/cart/clear
```

- **DB writes**: `CartItem` add/update/delete per action.
- **Screens**: Cart icon on top, cart page with quantity selectors, remove buttons, clear cart action.

1. Wallet Management

```
GET /api/v1/consumer/wallet
```

```
POST /api/v1/consumer/wallet/load
```

```
POST /api/v1/consumer/wallet/debit
```

```
POST /api/v1/consumer/wallet/refund
```

```
GET /api/v1/consumer/wallet/history
```

- **DB writes**: `ConsumerWallet` created if missing. `WalletTransaction` inserted for each load/debit/refund.
- **Screens**: Wallet balance page, add money form, history list.

1. Placing an Order

```
POST /api/v1/consumer/order/confirm
```

- Body fields: `payment_mode` (cash or wallet), `delivery_notes`.
- **DB writes**: new `Order`, `OrderItem` rows for items, `OrderStatusLog` entry "pending". Cart cleared. Wallet debited if payment_mode=wallet.
- **Screens**: Order confirmation page summarizing totals, choose payment mode, success confirmation.

1. Viewing Order History & Actions

```
GET /api/v1/consumer/order/history
```

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

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

```
POST /api/v1/consumer/orders/<id>/confirm-modified
```

```
POST /api/v1/consumer/orders/<id>/cancel
```

```
POST /api/v1/consumer/orders/<id>/rate
```

```
POST /api/v1/consumer/orders/<id>/issue
```

```
POST /api/v1/consumer/orders/<id>/return/raise
```

- **DB interactions**: `OrderActionLog`, `OrderMessage`, `OrderRating`, `OrderIssue`, `OrderReturn` as appropriate. Wallet adjustments on cancel or return.
- **Screens**: Order list, order detail page with actions (chat, cancel, confirm, etc.), rating dialog, issue report form, return request form.

1. Optional AI Assistant

```
POST /api/v1/agent/query
```

- **DB reads**: none, but user authentication required.
- **Screens**: Chat or help screen where user enters questions.

1. Summary of Data Flow

- User data persists mainly in `UserProfile` and `ConsumerProfile`.
- Cart operations read/write `CartItem` with shop and item relations.
- Orders generate `Order`, `OrderItem`, logs and messages, with wallet debits/credits recorded in `WalletTransaction`.
- Throughout the flow, the frontend should display status messages from API responses.

1. Profile Management

- **Screen**: "Edit Profile" accessible via the menu. Displays fields prefilled using `GET /api/v1/consumer/profile/me`.
- User edits name, preferred language, flat number and uploads a new avatar.
- "Save" triggers `POST /api/v1/consumer/profile/edit`.
- **DB writes**: Updates to `ConsumerProfile` and `UserProfile` timestamp.
- Success message shown and local profile cache refreshed via the same GET endpoint.

```
GET /api/v1/consumer/profile/me
```

```
POST /api/v1/consumer/profile/edit
```

1. Address Book

- **Screen**: Manage delivery addresses listing existing rows from `GET /api/v1/consumer/addresses`.
- Add button opens form for house number, street and landmark.
- Submission hits `POST /api/v1/consumer/address/add` and on success displays in list.
- Each address can be edited `POST /api/v1/consumer/address/<id>/edit` or deleted `POST /api/v1/consumer/address/<id>/delete`.
- **DB writes**: new `ConsumerAddress` rows or updates; soft delete flag when removed.

```
GET /api/v1/consumer/addresses
```

```
POST /api/v1/consumer/address/add
```

```
POST /api/v1/consumer/address/<id>/edit
POST /api/v1/consumer/address/<id>/delete
```

1. Wishlist

- **Screen**: Items the consumer saved for later. Data from `GET /api/v1/consumer/wishlist`.
- Tap heart on item detail triggers `POST /api/v1/consumer/wishlist/add` with `item_id`.
- Remove via `POST /api/v1/consumer/wishlist/remove`.
- **DB writes**: `WishlistItem` table referencing user and item.
- Items can be moved to cart via `POST /api/v1/consumer/cart/add` from the wishlist view.

```
GET /api/v1/consumer/wishlist
```

```
POST /api/v1/consumer/wishlist/add
```

```
POST /api/v1/consumer/wishlist/remove
```

```
POST /api/v1/consumer/cart/add
```

1. Notifications

- **Screen**: Notification center listing updates from `GET /api/v1/consumer/notifications`.
- Includes order status changes, wallet events and promotional messages.
- Tapping a notification deep links to the relevant order or wallet screen.
- Mark read action uses `POST /api/v1/consumer/notifications/mark-read`.
- **DB reads**: `Notification` table filtered by user; `read_at` timestamp set when marked.

```
GET /api/v1/consumer/notifications
```

```
POST /api/v1/consumer/notifications/mark-read
```

1. Advanced Search & Filters

- **Screen**: Search bar plus filters for price range, shop rating and item availability.
- `GET /api/v1/consumer/shops/search` with additional query params like `min_price`, `max_price` or `rating`.
- Results list open shops first, then closed ones greyed out.
- **DB reads**: same `Shop` table with dynamic filtering and sort by rating.
- Clear filter button resets query and refreshes results.

```
GET /api/v1/consumer/shops/search
```

1. Promotional Offers

- **Screen**: Dedicated page for coupons and vendor promotions.
- Consumer fetches active offers via `GET /api/v1/consumer/offers`.
- Apply coupon on checkout using `POST /api/v1/consumer/cart/apply-coupon` returning adjusted total.
- **DB interactions**: `CouponRedemption` insert, order total recalculated on server.

```
GET /api/v1/consumer/offers
```

```
POST /api/v1/consumer/cart/apply-coupon
```

1. Loyalty & Rewards

- **Screen**: Shows points earned from past purchases. Data via `GET /api/v1/consumer/rewards`.
- Redeem points at checkout using `POST /api/v1/consumer/rewards/redeem` with amount.

- **DB writes**: `RewardLedger` credit on order completion and debit on redemption.
- Visual meter displays progress towards next reward tier.

```
GET /api/v1/consumer/rewards
```

```
POST /api/v1/consumer/rewards/redeem
```

1. Support & Help Center

- **Screen**: FAQ list plus option to open a support ticket.
- Tickets posted through `POST /api/v1/consumer/support/ticket`.
- View previous tickets with `GET /api/v1/consumer/support/tickets` and chat replies with `GET /api/v1/consumer/support/ticket/<id>`.
- **DB writes**: `SupportTicket` and `SupportMessage` tables; `status` updated by staff.

```
POST /api/v1/consumer/support/ticket
```

```
GET /api/v1/consumer/support/tickets
```

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

1. Data & Privacy Settings

- **Screen**: Toggle location access, set data sharing preferences.
- Retrieve current settings via `GET /api/v1/consumer/settings`.
- Changes submitted by `POST /api/v1/consumer/settings/update`.
- **DB writes**: flags stored on `UserProfile` such as marketing_opt_in or location_allowed.

```
GET /api/v1/consumer/settings
```

```
POST /api/v1/consumer/settings/update
```

1. Logout & Multi-Device

- **Screen**: Option to sign out on current device or all devices.
- Logout calls `POST /api/v1/logout` and removes tokens locally.
- For global signout there is `POST /api/v1/logout-all` revoking refresh tokens in database.
- After logout user returns to phone entry screen completing the cycle.

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

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