Contents

1 Introduction

This document describes the requirements and modelling of *Wigvana*—an e-commerce platform for buying and selling human hair extensions.

2 Stakeholders

1. Primary stakeholders:

   (a) Buyers / Customers:

   - **Role**: Individuals or groups purchasing hair extension items.
   - **Interests**: Wide selection, fair prices, accurate product descriptions and images, easy navigation and search, secure payment options, reliable and timely shipping, hassle-free returns/exchanges, good customer service, trend information, personalization.

   (b) Sellers / Vendors / Brands:

   - **Role**: Individuals, small businesses, or established brands listing and selling their products on the platform.
   - **Interests**: Access to a large customer base, easy-to-use tools for listing products and managing inventory, fair commission/fee structure, timely payments, reliable platform performance, seller support, marketing opportunities, brand visibility, protection against fraud.

   (c) Platform Owner / Operator (Wigvana):

   - **Role**: The entity that builds, manages, and operates the e-commerce platform.
   - **Interests**: Profitability (through commissions, listing fees, advertising, etc.), user growth (both buyers and sellers), platform stability and performance, brand reputation, operational efficiency, legal compliance, competitive advantage, data insights.

   (d) Payment Gateway Providers:

   - **Role**: Third-party services facilitating secure online transactions (e.g., Stripe, PayPal, local mobile money operators).
   - **Interests**: Transaction volume, seamless integration, platform reliability, security compliance, timely settlement of funds.

2. Secondary stakeholders:

   (a) Payment Processors (e.g., Stripe, PayPal, Banks): Facilitate secure transactions.

   (b) Logistics & Shipping Partners (e.g., FedEx, DHL, local couriers): Handle order fulfillment and delivery.

   (c) Marketing & Advertising Partners: Agencies or platforms (Google Ads, Meta) helping attract buyers.

   (d) Third-Party Integrations (e.g., CRM, analytics tools): Services enhancing platform functionality.

3 Actors

3.1 Buyers (Customers)

The buying actor can behave in two ways:

- With their identity established, i.e., *authenticated*.
- Browsing without an established identity (Anonymous).

Goals:

1. Browse/search for fashion products by using:
   - Selecting a product functionality:
     (a) Category
     (b) Price
     (c) Search terms
   - Viewing details on a product:
     (a) Name
     (b) Description
     (c) Price
     (d) Images
     (e) Seller details
2. Compare prices, styles, and reviews.
3. Add items to cart/wishlist.
4. Make secure payments.
5. Track orders & request returns/refunds (Line of communication with vendors).
6. Leave reviews & ratings.

3.2 Sellers (Merchants/Vendors)

Goals:

1. Register & set up seller profiles.
2. List, update, and manage product inventory.
3. Set pricing, discounts, and promotions.
4. Process & fulfill orders.
5. Handle returns/refunds.
6. Analyze sales performance.

3.3 Platform Administrator

Goals:

1. Manage user accounts (buyers & sellers).
2. Moderate listings (fraud, counterfeit detection).
3. Handle disputes (refunds, scams).
4. Monitor platform performance & security.
5. Update platform features & policies.

4 Use cases

- Search for a product.
- Place an order.

5 Authentication

5.1 Authentication flows

1. Login flow

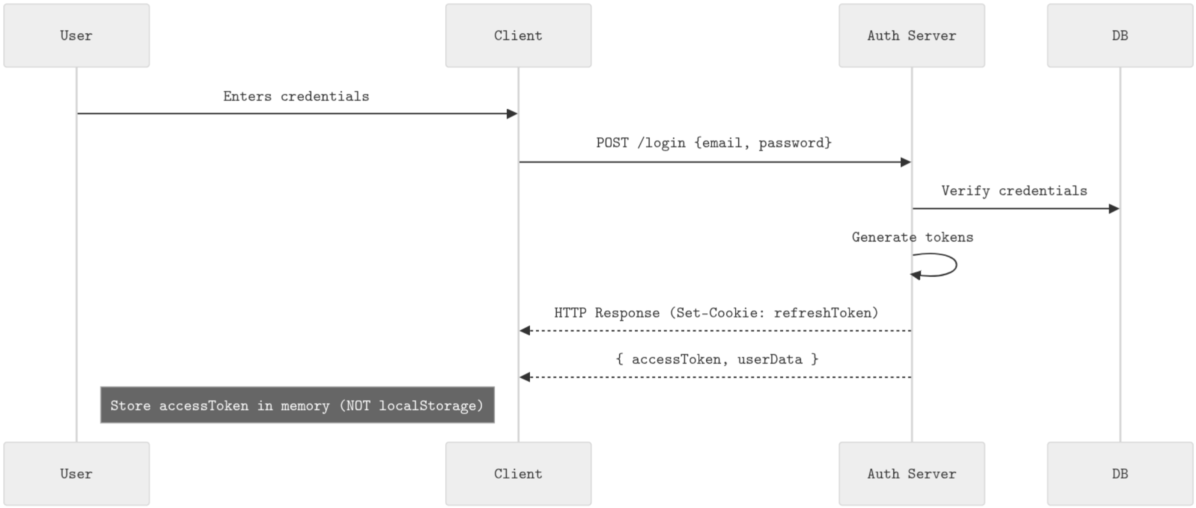
User
Client
Auth Server
DB
Enters credentials
POST /login {email, password}
Verify credentials
Generate tokens
HTTP Response (Set-Cookie: refreshToken)
{ accessToken, userData }
Store accessToken in memory (NOT localStorage)
User
Client
Auth Server
DB

Figure 1: Login flow.

   What happens:

   - Server validate credentials.
   - Generates:

     (a) Short-lived access tokens (e.g., 15-30 mins)

     ```
     {
       "sub": "user123",
       "roles": ["user"],
       "iat": 1620000000,
       "exp": 1620001800
     }

     ```

     (b) Long lived refresh tokens (e.g., 7 days) stored in `HTTPOnly` cookie.

   - Client stores access token in memory (React/Vue state, Angular service)

2. Accessing protected resources.

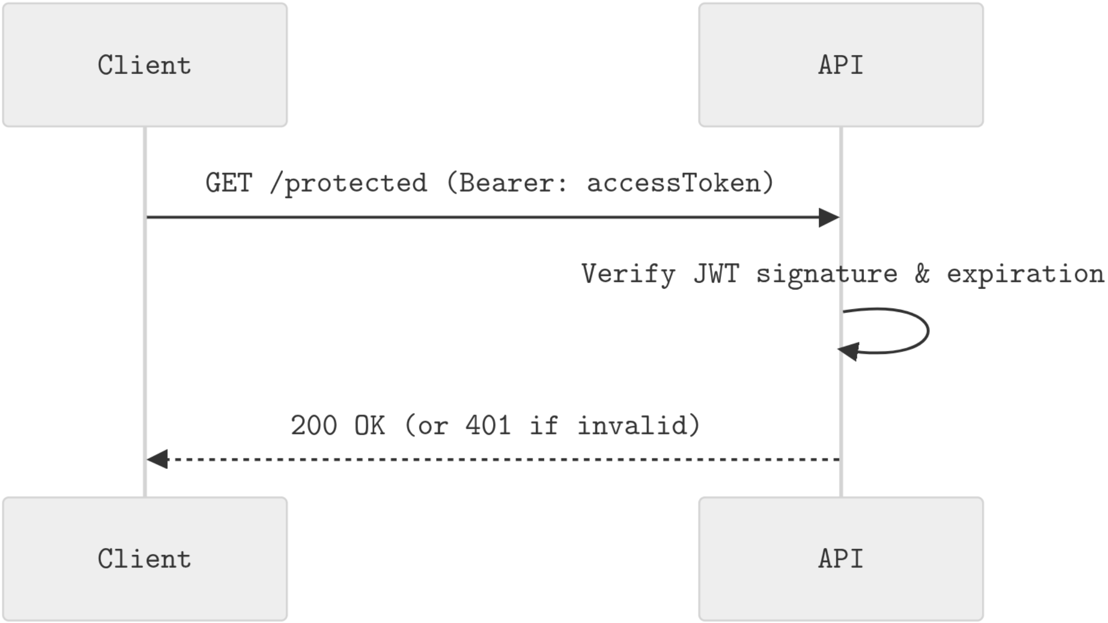
Client
API
GET /protected (Bearer: accessToken)
Verify JWT signature & expiration
200 OK (or 401 if invalid)
Client
API

Figure 2: Protected resource access.

What happens:

- Validate JWT signature using server's secret/public key.
- Check `exp` claim.
- Verify token wasn't revoked (optional denylist for critical systems)

3. Access token expiration (silent refresh)

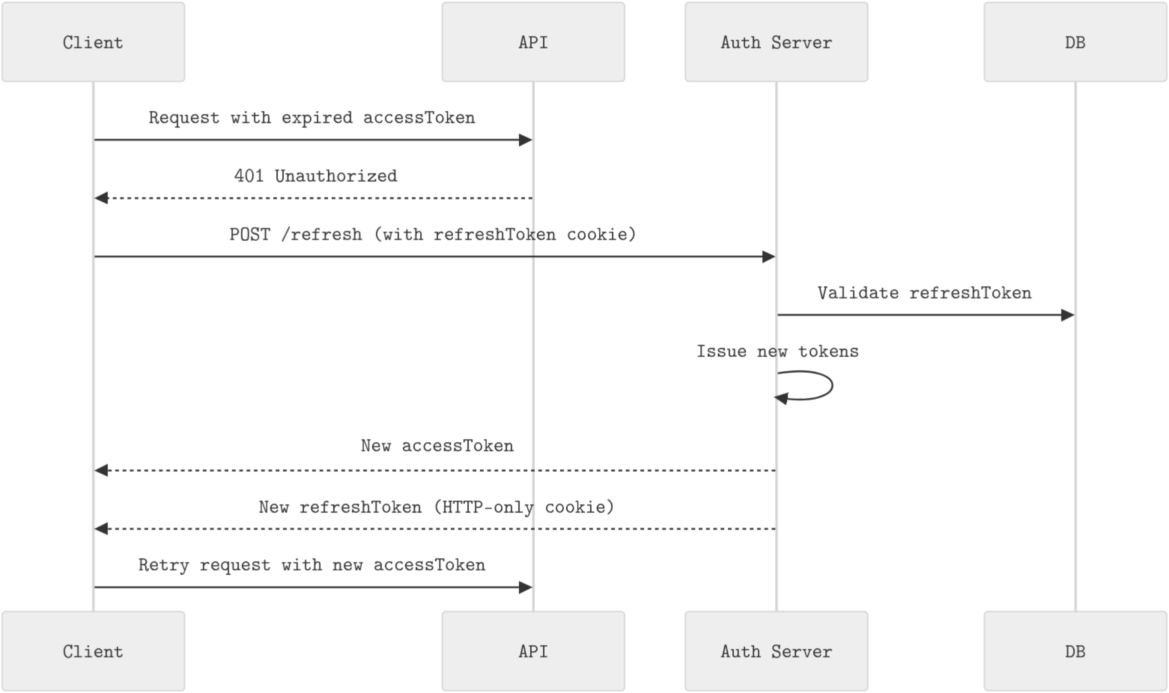
Client
API
Auth Server
DB
Request with expired accessToken
401 Unauthorized
POST /refresh (with refreshToken cookie)
Validate refreshToken
Issue new tokens
New accessToken
New refreshToken (HTTP-only cookie)
Retry request with new accessToken
Client
API
Auth Server
DB

Figure 3: Silent refresh flow.

Considerations

- Refresh token is never exposed to JS (`HttpOnly` cookie)
- Server rotates refresh tokens (invalidates old one)
- If refresh token is invalid/expired, force logout

4. Logout

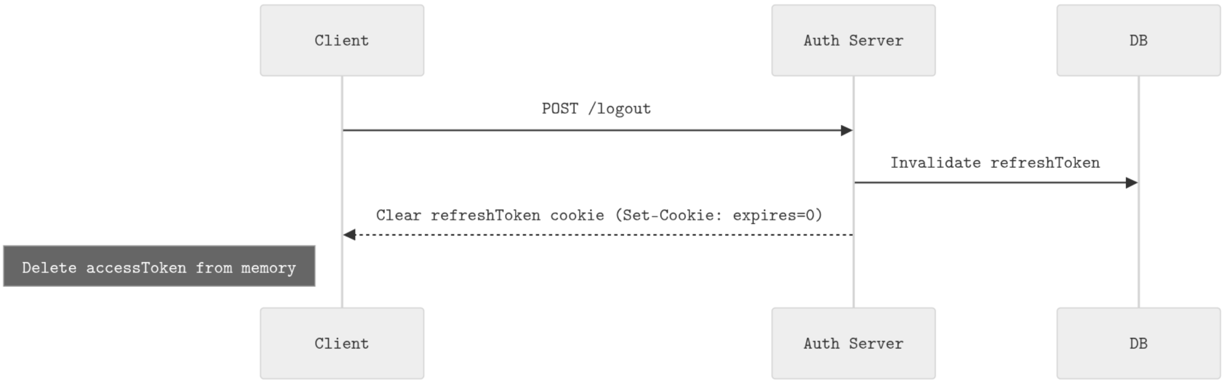
Client
Auth Server
DB
POST /logout
Invalidate refreshToken
Clear refreshToken cookie (Set-Cookie: expires=0)
Delete accessToken from memory
Client
Auth Server
DB

Figure 4: Logout flow.

Critical actions:

- Server adds refresh token to denylist (or deletes from DB)
- Client removes access token from memory
- Cookie is cleared via `Set-Cookie` header