API Mapping Guide - Which API to Call Where

This guide shows exactly which backend API endpoint to call from each frontend page and user action.

□ login.html

When: User clicks "Login" button

```
// Action: User login
POST /api/auth/login

// Request:
{
  "username": "john_doe",
  "password": "password123"
}

// Response:
{
  "access_token": "eyJ0eXAiOiJKV1Q...",
  "refresh_token": "eyJ0eXAiOiJKV1Q...",
  "user": {
    "id": 1,
    "username": "john_doe",
    "email": "john@example.com"
  }
}

// What to do:
// 1. Store access_token in localStorage
// 2. Store refresh_token in localStorage
// 3. Redirect to dashboard.html
```

When: Token expires

```
// Action: Refresh access token
POST /api/auth/refresh

// Request:
{
  "refresh_token": "stored_refresh_token"
}

// Response:
{
  "access_token": "new_access_token"
}
```

□ signUp.html

When: User clicks "Sign Up" button

```
// Action: Register new user
POST /api/auth/register

// Request:
{
  "username": "new_user",
  "email": "newuser@example.com",
  "password": "securepass123"
}

// Response:
{
  "message": "User created successfully",
  "user_id": 2
}

// What to do:
// 1. Show success message
// 2. Redirect to login.html (or auto-login)
```

□ dashboard.html

When: Page loads (get current user info)

```
// Action: Get logged-in user details
GET /api/auth/me

// Headers:
Authorization: Bearer {access_token}

// Response:
{
  "id": 1,
  "username": "john_doe",
  "email": "john@example.com",
  "created_at": "2026-01-15T10:30:00Z"
}
```

When: Fetching user's saved searches

```
// Action: Get saved searches
GET /api/users/saved-searches

// Headers:
Authorization: Bearer {access_token}

// Response:
[
  {
    "id": 1,
    "search_query": "iPhone 15",
    "filters": {},
    "created_at": "2026-02-01T14:20:00Z"
  }
]
```

When: Fetching user's price alerts

```
// Action: Get price alerts
GET /api/users/alerts

// Headers:
Authorization: Bearer {access_token}

// Response:
[
  {
    "id": 1,
    "product_listing_id": 45,
    "target_price": 95000,
    "is_active": true,
    "created_at": "2026-02-05T09:15:00Z"
  }
]
```

□ search.html

When: User enters search query and clicks "Search"

```
// Action: Search for products
GET /api/products/search?q={query}

// Example:
GET /api/products/search?q=iPhone+15+Pro

// Headers:
Authorization: Bearer {access_token}

// Response:
[
  {
    "id": 1,
    "name": "iPhone 15 Pro 256GB",
    "brand": "Apple",
    "category": "smartphones",
    "image_url": "https://...",
    "listings": [
      {
        "id": 1,
        "platform": "Amazon",
        "price": 99999,
        "original_price": 109999,
        "discount_percentage": 9.1,
        "rating": 4.5,
        "availability_status": "in_stock"
      },
      {
        "id": 2,
        "platform": "Flipkart",
        "price": 98999,
        "original_price": 109999,
        "discount_percentage": 10.0,
        "rating": 4.4,
        "availability_status": "in_stock"
      }
    ]
  }
]

// What to do:
// 1. Display product cards
// 2. Show best price among all listings
// 3. Add "Compare Prices" button
// 4. Add "View Details" button
```

When: User clicks "Save Search" button

```
// Action: Save current search
POST /api/users/saved-searches

// Headers:
Authorization: Bearer {access_token}

// Request:
{
  "search_query": "iPhone 15 Pro",
  "filters": {
    "category": "smartphones",
    "brand": "Apple"
  }
}

// Response:
{
  "id": 2,
  "message": "Search saved successfully"
}
```

□ compare.html

When: Page loads (user clicked "Compare Prices" from search)

```
// Action: Get all price listings for comparison
POST /api/comparison

// Headers:
Authorization: Bearer {access_token}

// Request:
{
  "product_id": 1,
  "sort_by": "price"  // or "rating", "discount"
}

// Response:
[
  {
    "id": 2,
    "platform": "Flipkart",
    "price": 98999,
    "original_price": 109999,
    "discount_percentage": 10.0,
    "rating": 4.4,
    "delivery_time": "3 days",
    "delivery_charges": 0,
    "availability_status": "in_stock"
  },
  {
    "id": 1,
    "platform": "Amazon",
    "price": 99999,
    "original_price": 109999,
    "discount_percentage": 9.1,
    "rating": 4.5,
    "delivery_time": "2 days",
    "delivery_charges": 0,
    "availability_status": "in_stock"
  }
]

// What to do:
// 1. Display side-by-side comparison cards
// 2. Highlight best deal (lowest score)
// 3. Show "Buy Now" buttons linking to platforms
```

When: Need to find the single best deal

```
// Action: Get only the best deal
GET /api/comparison/best-deal?product_id=1

// Headers:
Authorization: Bearer {access_token}

// Response:
{
  "id": 2,
  "platform": "Flipkart",
  "price": 98999,
  "original_price": 109999,
  "discount_percentage": 10.0,
  "rating": 4.4,
  "delivery_time": "3 days",
  "is_best_deal": true
}
```

□ productDetails.html

When: Page loads (user clicked on a product)

```
// Action: Get single product details
GET /api/products/{product_id}

// Example:
GET /api/products/1

// Headers:
Authorization: Bearer {access_token}

// Response:
{
  "id": 1,
  "name": "iPhone 15 Pro 256GB",
  "brand": "Apple",
  "category": "smartphones",
  "description": "Latest iPhone with A17 Pro chip...",
  "image_url": "https://...",
  "listings": [
    {
      "id": 1,
      "platform": "Amazon",
      "price": 99999,
      "product_url": "https://amazon.in/..."
    },
    {
      "id": 2,
      "platform": "Flipkart",
      "price": 98999,
      "product_url": "https://flipkart.com/..."
    }
  ]
}
```

When: User wants to see price history chart

```
// Action: Get price history for chart
GET /api/products/{product_id}/price-history

// Example:
GET /api/products/1/price-history

// Headers:
Authorization: Bearer {access_token}

// Response:
[
  {
    "id": 1,
    "price": 99999,
    "recorded_at": "2026-02-01T08:00:00Z",
    "platform": "Amazon"
  },
  {
    "id": 2,
    "price": 99499,
    "recorded_at": "2026-02-03T08:00:00Z",
    "platform": "Amazon"
  },
  {
    "id": 3,
    "price": 98999,
    "recorded_at": "2026-02-05T08:00:00Z",
    "platform": "Amazon"
  }
]

// What to do:
// 1. Use Chart.js or similar
// 2. X-axis: recorded_at (dates)
// 3. Y-axis: price
// 4. Show trend line
```

When: User clicks "Set Price Alert"

```
// Action: Create price alert
POST /api/users/alerts

// Headers:
Authorization: Bearer {access_token}

// Request:
{
  "product_listing_id": 1,  // Specific platform listing
  "target_price": 95000,
  "alert_type": "below_price"  // or "price_drop"
}

// Response:
{
  "id": 3,
  "message": "Price alert created successfully"
}
```

□ profile.html

When: Page loads

```
// 1. Get user info
GET /api/auth/me

// 2. Get saved searches
GET /api/users/saved-searches

// 3. Get price alerts
GET /api/users/alerts
```

When: User deletes a saved search

```
// Action: Delete saved search
DELETE /api/users/saved-searches/{id}

// Example:
DELETE /api/users/saved-searches/1

// Headers:
Authorization: Bearer {access_token}

// Response:
{
  "message": "Saved search deleted successfully"
}
```

When: User deletes a price alert

```
// Action: Delete price alert
DELETE /api/users/alerts/{id}

// Example:
DELETE /api/users/alerts/1

// Headers:
Authorization: Bearer {access_token}

// Response:
{
  "message": "Price alert deleted successfully"
}
```

□ Authentication Headers

All protected endpoints require this header:

```
Headers: {
  'Authorization': 'Bearer ' + localStorage.getItem('access_token'),
  'Content-Type': 'application/json'
}
```

Example fetch call:

```
const token = localStorage.getItem('access_token');

const response = await fetch('http://localhost:8000/api/products/search?q=iPhone', {
  method: 'GET',
  headers: {
    'Authorization': `Bearer ${token}`,
    'Content-Type': 'application/json'
  }
});

const data = await response.json();
```

□ Summary Table

Frontend Page	User Action	API Endpoint	Method
login.html	Login	`/api/auth/login`	POST
login.html	Refresh token	`/api/auth/refresh`	POST
signUp.html	Register	`/api/auth/register`	POST
dashboard.html	Load user info	`/api/auth/me`	GET
dashboard.html	Load saved searches	`/api/users/saved-searches`	GET
dashboard.html	Load price alerts	`/api/users/alerts`	GET
search.html	Search products	`/api/products/search?q={query}`	GET
search.html	Save search	`/api/users/saved-searches`	POST
compare.html	Compare prices	`/api/comparison`	POST
compare.html	Get best deal only	`/api/comparison/best-deal?product_id={id}`	GET
productDetails.html	Load product	`/api/products/{id}`	GET
productDetails.html	Price history	`/api/products/{id}/price-history`	GET
productDetails.html	Set price alert	`/api/users/alerts`	POST
profile.html	Delete saved search	`/api/users/saved-searches/{id}`	DELETE
profile.html	Delete price alert	`/api/users/alerts/{id}`	DELETE

□ Complete Example: Search Flow

```
// search.html - Complete implementation

document.getElementById('searchForm').addEventListener('submit', async (e) => {
  e.preventDefault();

  const query = document.getElementById('searchInput').value;
  const token = localStorage.getItem('access_token');

  // Show loading
  showLoader();

  try {
    // Call search API
    const response = await fetch(
      `http://localhost:8000/api/products/search?q=${encodeURIComponent(query)}`,
      {
        headers: {
          'Authorization': `Bearer ${token}`,
          'Content-Type': 'application/json'
        }
      }
    );

    if (!response.ok) {
      if (response.status === 401) {
        // Token expired - redirect to login
        window.location.href = 'login.html';
        return;
      }
      throw new Error('Search failed');
    }

    const products = await response.json();

    // Display results
    displayProducts(products);

  } catch (error) {
    console.error('Error:', error);
    showError('Failed to search products');
  } finally {
    hideLoader();
  }
});

function displayProducts(products) {
  const container = document.getElementById('resultsContainer');
  container.innerHTML = '';

  products.forEach(product => {
    // Find cheapest listing
    const cheapest = product.listings.reduce((min, listing) =>
      listing.price < min.price ? listing : min
    );

    const card = `
      <div class="product-card">
        <img src="${product.image_url}" alt="${product.name}">
        <h3>${product.name}</h3>
        <p class="brand">${product.brand}</p>
        <p class="price">From ₹${cheapest.price.toLocaleString('en-IN')}</p>
        <p class="platform">on ${cheapest.platform}</p>
        <div class="actions">
          <button onclick="viewProduct(${product.id})">View Details</button>
          <button onclick="compareProduct(${product.id})">Compare Prices</button>
        </div>
      </div>
    `;

    container.innerHTML += card;
  });
}

function compareProduct(productId) {
  // Redirect to compare page
  window.location.href = `compare.html?product_id=${productId}`;
}

function viewProduct(productId) {
  // Redirect to product details page
  window.location.href = `productDetails.html?id=${productId}`;
}
```

That's it! Every frontend action now has a clear API endpoint to call. □