Phase 1: Backend Setup and Web Scraping Data Integration (4–6 weeks)			
Goal: Set up the backend infrastructure to receive, store, and test web scraping data.			
Milestone	**Description**	**Responsible**	**Timeline**
1. Deploy Environment	Set up the development environment, repository, and dev server.	Full Stack Dev	Week 1
2. Database Creation	Create the initial database schema (PHP Laravel) based on core data ne	Full Stack Dev	Week 1
3. Web Scraping Setup	Start scraping data from retailer sites using Scrapy/Selenium and store in	Web Scraping Team	Week 2
4. Test Data Flow	Verify that web scraping data is correctly stored in the database.	Full Stack Dev / Scraper	Week 2–3
5. API Development	Build APIs for data retrieval and any initial reporting needs.	Full Stack Dev	Week 3–4
6. Simple Frontend Setup	Create a basic frontend using Bootstrap to display stored data (for testing	Full Stack Dev	Week 4–5
7. Backend Testing	Test the system for proper integration and ensure data can flow smoothly	Full Stack Dev / Analyst	Week 5–6
Phase 1: Deliverables			
Backend (PHP Laravel) set up and connected to the database.			
Database schema implemented and tested.			
Web scraping data flowing into the system and stored correctly.			
API endpoints for testing data retrieval.			
Simple front-end interface to view and interact with data.			

Phase 2: Calculation Logic, Insights, and Advanced Features (4–6 weeks)			
Goal: Add advanced calculation logic, reporting, and insights based on the web scraping data.			
Milestone	**Description**	**Responsible**	**Timeline**
1. Data Engineer Onboarding	Bring the data engineer onboard to review the backend and begin data pr	Data Engineer	Week 1
2. Calculation Logic Setup	Implement logic for share of voice, SKU matching, compliance, etc.	Data Engineer	Week 1–2
3. Insights Development	Develop insights for key KPIs such as share of voice, market share, and	Data Engineer / Backend	Week 2–4
4. Backend & Database Updates	Ensure that the database and backend are updated for new insights and	Full Stack Dev / Data En	Week 3–5
5. Frontend Enhancements	Build frontend widgets and dashboards to display actionable insights.	Full Stack Dev / Data En	Week 4–6
6. Final Testing & Deployment	Test the full solution, including advanced insights and frontend functionali	Full Stack Dev / Data En	Week 5–6
Phase 2: Deliverables			
Advanced calculation logic (e.g., share of voice, SKU matching) implemented.			
Dynamic insights created based on key KPIs.			
Frontend enhancements displaying insights in an intuitive, user-friendly way.			
Comprehensive testing to ensure end-to-end functionality.			

Technical Brief		
Section	**Description**	**Action Required**
Objective	Define the technical requirements to manage SKU matching, multimodal processing, dynamic web scraping, and a scalable category-agnostic database schema.	Ensure that the schema and processing can handle multiple product categories and flexible scraping workflows.
SKU Matching (Multimodal)	Use multimodal techniques (e.g., image recognition, text matching) to compare product titles and images with manufacturer SKU codes (EAN/UPC).	API-driven SKU matching based on product titles, images, and manufacturer part numbers. Implement confidence score thresholds.
Multimodal Image and Text Compliance	Perform multimodal checks to ensure product title, descriptions, and images match the source of truth provided by the brand.	Develop an API to compare images and text data (title and bullet points) for compliance, returning a match score for each product.
Category-Agnostic Database Schema	Implement a category-agnostic schema using shared tables for common product data (e.g., name, price, stock status), with dynamic attributes for category-specific features.	Create tables such as products, categories, and product_attributes to support scaling to multiple product categories easily.
JSON-Based Attributes (Optional)	Use JSON fields to store category-specific attributes directly in the products table for flexibility.	Example: Store attributes like {"RAM": "8GB", "screen_size": "15 inches"} in the attributes JSON field.
Multiple Web Scrapers	Use multiple web scrapers (e.g., Scrapy, Browse AI) to maximize data delivery, all using the same structured format for the PHP Laravel backend.	Ensure that all scrapers deliver data in a consistent format (e.g., JSON), using the same schema to ensure data uniformity.
Web Scraping Integration	Ensure that data scraped by Scrapy, Browse AI, or other tools arrives structured (e.g., JSON) and is fed into the backend for storage and processing.	Use APIs to send structured data directly from scrapers to the backend for processing. Capture screenshots, prices, and descriptions.
Source of Truth Data	Ingest product images and information from brands (via PIM connection or CSV) as the source of truth for compliance checks (title, image, attributes).	Store source of truth data in separate tables for compliance checks. Use PIM connection ID for traceability.
Data Processing Pipeline	Develop an ETL pipeline to process the scraped data and compare it against the source of truth for SKU matching, content compliance, and rich content checks.	Build this as an internal pipeline that normalizes and transforms incoming scraped data and prepares it for API-based checks.
Product Table Structure	Central table to store product details including common attributes (e.g., price, stock, description), category reference, and JSON-based custom attributes.	Example columns: id, product_name, category_id, price, stock_status, description, attributes (JSON), created_at.
Category Table Structure	Table to store category information, enabling a category-agnostic setup.	Example columns: id, name (e.g., laptops, smartphones), created_at, updated_at.
Product Attributes Table (if not using JSON)	Store dynamic attributes for each product category (e.g., RAM, processor) in a separate table linked to the product.	Example columns: product_id, attribute_name (e.g., RAM), attribute_value (e.g., "8GB").
PIM Integration (Source of Truth)	Set up API or batch processes to ingest PIM system data (images, product descriptions) as the source of truth.	Example: Ingest PIM-provided product images, titles, and descriptions via API or CSV into a separate source of truth table.
Multimodal Processing Flow	Once data is stored, call the multimodal API for SKU matching, image compliance, and content validation, and store results in the database.	Ensure multimodal API results (e.g., compliance score, image match) are stored in the database for further analysis or reporting.
API and Backend Flow	API endpoints should ingest structured data from scrapers, send it to multimodal processing, and store results for reporting and dashboards.	Use API requests to handle data flow from scraping to multimodal content compliance, image matching, and SKU matching.

Key Points for the Backend Team:
Category-Agnostic Database Schema: Use a single schema with tables for shared product attributes and dynamic handling of category-specific fields (JSON or product_attributes).
Multiple Scrapers Integration: Implement consistent data formats across scrapers (Scrapy, Browse AI) to ensure uniformity. Scrapers should deliver structured data (e.g., JSON) into the PHP Laravel backend via API.
Source of Truth Compliance: The backend will need to ingest source of truth data from PIM systems or CSV files to validate against the scraped data (for SKU matching, content compliance, etc.).
Multimodal Processing: After data is stored, the backend should trigger multimodal processing (via API) for image matching, text compliance, and SKU matching. Results should be stored in the database for further analysis.

PHP Lavarel: Technical Brief for Scalable Category Agnostic Database Schema		
Section	**Description**	**Action Required**
Objective	Ensure the database schema is flexible and scalable to support multiple product categories (e.g., laptops, tablets).	Develop a category-agnostic schema that allows for the addition of new categories without major schema changes.
General Schema Approach	Use a single database schema for all categories rather than duplicating schema per category.	Create shared tables for common attributes (e.g., products, categories, pricing) that apply across all categories.
Dynamic Attributes Handling	Implement a flexible system to manage category-specific attributes using dynamic fields (e.g., attribute-value model or JSON).	Set up a product_attributes table to handle unique product attributes dynamically, or use JSON fields for flexibility.
Products Table	Centralized table storing core product details such as product name, price, stock, category reference, etc.	Example columns: id, product_name, category_id, price, stock_status, description, created_at, updated_at.
Categories Table	Table to store category information (e.g., laptops, smartphones, tablets)	Example columns: id, name (e.g., laptops, smartphones), created_at, updated_at.
Product Attributes Table	Dynamic table to store category-specific attributes (e.g., screen size, RAM for laptops, battery for smartphones).	Example columns: product_id, attribute_name (e.g., screen_size, RAM), attribute_value (e.g., "15 inches", "8GB").
JSON-based Attributes (Optional)	Use JSON fields to store product-specific attributes directly in the products table for flexibility.	Example column: attributes (JSON object), e.g., {"screen_size": "15 inches", "RAM": "8GB"}.
Attribute Templates (Optional)	Set up attribute templates for predefined sets of attributes by category, which can help manage validation.	Create a product_types table that stores default attributes for each category (e.g., laptops, smartphones).
Expansion to New Categories	When new categories are introduced (e.g., tablets, wearables), the existing schema will support these with minor adjustments.	No new schema required. Add new category entries in the categories table and define their specific attributes dynamically.
ETL Pipelines (Optional)	Consider implementing an ETL pipeline to standardize data from various sources (e.g., web scraping, PIM systems).	Integrate ETL tools to normalize category-specific data before it's inserted into the database.
API Integration for Multimodal	Ensure the API endpoints that process multimodal data (e.g., SKU matching, image compliance) are category-agnostic.	Design the API to accept dynamic attributes from different categories, ensuring flexibility when scaling new categories.
Future-Proofing	The category-agnostic design should accommodate product-specific changes (e.g., new product types, attribute changes).	Ensure flexibility in data types and table structures to avoid the need for frequent database schema updates.

PHP Lavarel DB Schema: Retail Data Scraping Alignment				
Field Name	**Description**	**Field Type**	**Required**	**Example**
id	Primary key for the table	bigIncrements	Yes	1
retailer_id	Unique ID for the retailer	string	Yes	RET123
retailer_name	Name of the retailer	string	Yes	Walmart, Currys
retailer_country	Country of the retailer	string	Yes	USA, UK
retailer_website	Website address for the retailer	string	Yes	walmart.com, currys.co.uk
product_page_url	URL of the product page	string	Yes	https://walmart.com/product/12345
product_id	Retailer-specific product ID	string	Yes	12345-ABCD
manufacturer_ean_upc	Product's EAN or UPC code for SKU matching	string	Yes	1234567890123
manufacturer_name	Name of the manufacturer (e.g., Dell, HP)	string	Yes	Dell, HP
brand_name	Brand name of the product	string	Yes	Lenovo Yoga Slim, Acer Swift Go
category_name	Category name of the product	string	Yes	Laptops, Desktops
manufacturer_processor_name	Processor type (Intel, AMD, Snapdragon)	string	Yes	Intel, AMD
manufacturer_processor_brand_name	Processor brand name (e.g., Intel® Core™ i5)	string	Yes	Intel® Core™ i5
operating_platform_name	Operating system	string	Yes	Microsoft Windows 11, Google Chrome
software_included	Software included with the product	string	Yes	Windows 360 Trial, McAfee Anti-Virus
currency	Currency of the product price	string	Yes	USD, EUR
out_of_stock	Flag for out-of-stock status	boolean	Yes	Y/N
price	Price of the product	decimal	Yes	599.99
promotion_details	Details of any promotion applicable	text	Yes	"20% off" or "Buy 1 get 1 free"
category_rank	Rank position of the product in the category	Integer	Yes	1, 2, 3, etc.
result_type	Marks whether the result is paid (sponsored) or organic	String	Yes	"paid", "organic"
product_title	Title of the product	string	Yes	Lenovo Yoga Slim 7
product_description	Full description of the product	text	Yes	"Lightweight design, 16GB RAM, etc."
bullet_points	Key product features in bullet points	json	Yes	["Lightweight design", "Intel Core Processor"]
badges	Badges displayed for the product (e.g., Amazon Choice, Best Seller, Clubcard Deal)	Array	Yes	["Amazon Choice", "Best Seller"]
bullet_point_brand_logos	Logos displayed next to bullet points (e.g., Intel, AMD logos)	json	Yes	["Intel", "AMD"]
primary_image_url	URL of the primary image	string	Yes	https://image.url/primary.jpg
secondary_images	URLs of secondary images	json	Yes	["https://image.url/secondary1.jpg", "https://image.url/secondary2.jpg"]
image_sequence	Sequence order of secondary images	json	Yes	[1, 2, 3]
complementary_products	Complementary products (cross-sells and upsells)	json	Yes	["Lenovo Laptop Case", "USB Hub"]
product_recommendations	Recommended products	json	Yes	["Acer Swift 5", "HP Spectre x360"]
stock_availability_click_and_collect	Stock availability for click-and-collect option	boolean	Yes	Y/N
product_page_screenshot	Screenshot of the full product page	string	Yes	https://screenshot.url/page.jpg
technical_specs	Technical specifications of the product	text	Yes	"Processor: Intel, Memory: 16GB, etc."
video_url	URL of any video present on the product page	string	Yes	https://video.url/video.mp4
PHP Lavarel DB Schema: Retail Data Scraping Alignment (Grocery Specifics)				
ingredients	List of ingredients in the grocery product	Text	Yes	"Water, Sugar, Salt"
nutritional_info_energy_per_100g	Energy per 100g (kcal)	Integer	Yes	511
nutritional_info_energy_per_serving	Energy per serving size (kcal)	Integer	Yes	128
nutritional_info_energy_percentage		String	Yes	"6%"
nutritional_info_fat_per_100g	Energy percentage of daily intake per serving	Decimal	Yes	29
nutritional_info_fat_per_serving	Total fat per serving (grams)	Decimal	Yes	7.3
nutritional_info_fat_percentage	Fat percentage of daily intake per serving	String	Yes	"10%"
nutritional_info_saturates_per_100g	Saturates per 100g (grams)	Decimal	Yes	2.4
nutritional_info_saturates_per_serving	Saturates per serving (grams)	Decimal	Yes	0.6
nutritional_info_saturates_percentage	Saturates percentage of daily intake per serving	String	Yes	"3%"
nutritional_info_carbohydrate_per_100g	Carbohydrate per 100g (grams)	Decimal	Yes	53
nutritional_info_carbohydrate_per_serving	Carbohydrate per serving (grams)	Decimal	Yes	13
nutritional_info_sugars_per_100g	Sugars per 100g (grams)	Decimal	Yes	1.4
nutritional_info_sugars_per_serving	Sugars per serving (grams)	Decimal	Yes	0.4
nutritional_info_sugars_percentage	Sugars percentage of daily intake per serving	String	Yes	"<1%"
nutritional_info_fiber_per_100g	Fiber per 100g (grams)	Decimal	Yes	4
nutritional_info_fiber_per_serving	Fiber per serving (grams)	Decimal	Yes	1
nutritional_info_protein_per_100g	Protein per 100g (grams)	Decimal	Yes	6.7
nutritional_info_protein_per_serving	Protein per serving (grams)	Decimal	Yes	1.7
nutritional_info_salt_per_100g	Salt per 100g (grams)	Decimal	Yes	1.2
nutritional_info_salt_per_serving	Salt per serving (grams)	Decimal	Yes	0.31
nutritional_info_salt_percentage	Salt percentage of daily intake per serving	String	Yes	"5%"
nutritional_info_type	Specifies if the data is for adult or child	String	Yes	"Adult" / "Child"
country_of_origin	Country where the product is produced	String	Yes	"United Kingdom"
allergens	List of allergens in the product	Text	Yes	"Milk, Soy, Nuts"
dietary_claims	Dietary labels associated with product	Text	Yes	"Vegan, Gluten-Free"

serving_size	Standard serving size	Text	Yes	"25g"	
storage_instructions	Instructions for storage	Text	Yes	"Refrigerate after opening"	
preparation_instructions	Steps for preparation	Text	Yes	"Cook for 10 minutes"	
package_type	Material type of the product packaging	String	Yes	"Recycled Plastic"	
recyclable	Indicates if the packaging is recyclable (Yes/No)	Boolean	Yes	TRUE	
promotion_type	Type of promotion (e.g., loyalty card, multi-buy, price match, subscription)	String	Yes	"Clubcard Price, Aldi Price Match, Subscribe and Save"	
promotion_description	Detailed description of the promotion	Text	No	"Buy 2 Get 1 Free"	
promotion_discount	Discount amount or percentage (if applicable)	Decimal	No	20.0 (for 20% off)	
promotion_price	Price after the promotion is applied	Decimal	No	1.5	
promotion_conditions	Conditions for the promotion	Text	No	"Valid with Clubcard only"	
promotion_expiry	Expiration date for the promotion	Date	No	"2024-11-30"	
Rich Content					
rich_content	Rich content (e.g., brochures, enhanced product info)	text	Yes	Rich content description	
rich_content_displayed	Flag indicating if rich content is displayed (Y/N)	boolean	Yes	Y/N	
rich_content_images	URLs of any rich content images	json	Yes	["https://image.url/rich1.jpg", "https://image.url/rich2.jpg"]	
brand_mentions	List of brands mentioned on the product page	json	Yes	["Intel", "AMD", "Microsoft"]	
rich_content_brand_logos	Logos displayed in rich content (e.g., Intel, AMD logos)	json	Yes	["Intel", "AMD"]	
complementary_accessories	Complementary accessories in rich content	json	Yes	["Lenovo Laptop Stand", "USB-C Adapter"]	
Ratings & Reviews					
customer_reviews	Full text of customer reviews	json	Yes	["Great performance", "Battery life is short"]	
customer_rating	Star rating or score given by customers	float	Yes	4.5	
customer_review_date	Date of customer reviews	timestamp	Yes	2024-10-15	
customer_review_helpful_count	Number of times a review was marked helpful	integer	Yes	12	
sentiment_keywords	Keywords for sentiment analysis	json	Yes	["Fast", "Reliable", "Battery life"]	
reviews_timestamp	Timestamp for when the data was scraped	timestamp	Yes	2024-10-14 12:34:56	
Banners					
banner_brand_detection	Brand(s) detected in banner ads	string	Yes	Intel, AMD	
banner_type	Type of banner (e.g., static or dynamic)	string	Yes	Static, Dynamic	
banner_page_screenshot	Screenshot path for banners	string	Yes	https://screenshot.url/banner.jpg	
banner_destination_url	Destination URL for the banner ad	string	Yes	https://destination.url/	
banner_link_url	URL where the banner links to	string	Yes	https://link.url/	
banner_brands_in_carousel	Brands featured in carousel banners	json	Yes	["Intel", "AMD", "Apple"]	
Basket Conversion					
add_to_basket_recommendations	Products recommended when adding items to basket	json	Yes	["Laptop Case", "Wireless Mouse"]	
Brand Shops					
brand_shop_id	Unique identifier for the brand shop	String	Yes	intel_01	
brand_shop_name	Name of the brand shop	String	Yes	Dell Brand Shop	
brand_shop_url	URL of the brand shop homepage	String	Yes	https://amazon.com/dell	
brand_shop_retailer_id	Unique identifier for the marketplace	String	Yes	amazon_us	
is_in_brand_shop	Flag indicating if product is in the brand shop	Boolean	Yes	TRUE	
brand_shop_product_id	Retailer-specific product ID from brand shop	String	Yes	B08KFZLX7R	
brand_shop_product_title	Title of the product from brand shop	String	Yes	Dell XPS 13	
brand_shop_product_price	Price of the product from brand shop	String	Yes	$999.99	
brand_shop_product_rating	Rating score of the product from brand shop	Float	Yes	4.5	
brand_shop_product_review_count	Number of reviews for the product from brand shop	Integer	Yes	1500	
brand_shop_primary_image_url	URL of the primary image for the product	String	Yes	https://example.com/image1.jpg	
brand_shop_secondary_images	Array of secondary images for the product	Array	Yes	["https://example.com/image2.jpg", "https://example.com/image3.jpg"]	
brand_shop_out_of_stock	Flag indicating if the product is out of stock	Boolean	Yes	FALSE	
brand_shop_categories	Array of product categories within the brand shop	Array	Yes	["Laptops", "Gaming Laptops", "Desktops"]	
Best Seller Ranking					
best_seller_rank	Product's rank in the best sellers list	Integer	Yes	1, 2, 3, etc.	
best_seller_rank	Product's rank in the best sellers list	Integer	Yes	1, 2, 3, etc.	
best_seller_ranking_share_of_voice_in_top_100	Percentage of brand's products in the Top 100 Best Sellers	Percentage	Yes	10%	
best_seller_ranking_marketplace	Marketplace for the best seller ranking data	String	Yes	Amazon_uk	
best_seller_rank_brand_name	Brand name as it appears in the Best Seller Ranking	String	Yes	Dell	
best_seller_rank_node_url	URL of the best seller ranking node on Amazon	String	No	https://amazon.com/...	
best_seller_ranking_timestamp	Date and time of the best seller ranking scraping	Timestamp	Yes	2024-10-18 10:00:00	

PHP Lavarel DB Schema: Source Of Truth (PIM (Salsify, ICECAT, CSV)					
Field Name	**Description**	**Data Type**	**Required**	**Source of Truth**	**Example**
ean_upc	**Product's EAN or UPC code for SKU matching**	**string**	**Yes**	**ICECAT, Salsify, CSV**	**1234567890123**
mpn	Manufacturer Part Number (MPN)	string	Yes	ICECAT, Salsify, CSV	XYZ123
manufacturer	Manufacturer name	string	Yes	ICECAT, Salsify, CSV	Dell, HP
brand_name	Brand name for the product	string	Yes	ICECAT, Salsify, CSV	Lenovo Yoga Slim, Acer Swift Go

product_title	Title of the product	string	Yes	ICECAT, Salsify, CSV	Lenovo Yoga Slim 7
description	Full description of the product	text	Yes	ICECAT, Salsify, CSV	Full product description from ICECAT
technical_specifications	Product's technical specifications (e.g., processor, memory)	text	Yes	ICECAT, Salsify, CSV	Processor: Intel, Memory: 16GB
processor_brand	Processor brand (Intel, AMD, Snapdragon)	string	Yes	ICECAT, Salsify, CSV	Intel
operating_platform	Operating system	string	Yes	ICECAT, Salsify, CSV	Microsoft Windows 11, Google Chrome
software_included	Software included with the product	string	Yes	ICECAT, Salsify, CSV	Windows 360 Trial, McAfee Anti-Virus
image_url_primary	URL of the primary image for product	string	Yes	ICECAT, Salsify, CSV	https://image.url/primary.jpg
image_url_secondary_1	URL of the first secondary image	string	Yes	ICECAT, Salsify, CSV	https://image.url/secondary1.jpg
image_url_secondary_2	URL of the second secondary image	string	Yes	ICECAT, Salsify, CSV	https://image.url/secondary2.jpg
image_url_secondary_3	URL of the third secondary image	string	Yes	ICECAT, Salsify, CSV	https://image.url/secondary3.jpg
image_order	Sequence order for secondary images, if applicable	integer	Yes	1 (1st secondary image), 2 (2nd secondary image), etc.	
image_checksum	Checksum for image validation and integrity check	string	Yes	a1b2c3d4e5f6g7h8	
rich_content	Rich content elements (videos, brochures, etc.)	text	Yes	Brochure.pdf, Video.mp4	
pim_connection_id	Identifier for PIM connection	string	Yes	PIM-XYZ123	
source_of_truth_comparison	JSON field comparing source of truth across platforms	json	Yes	{"title_match": true, "image_match": false}	
updated_at	Timestamp for when the product data was last updated	timestamp	Yes	ICECAT, Salsify, CSV	2024-10-14 12:34:56
PHP Lavarel DB Schema: Search Scraping					
search_keyword	The keyword used in the search (generic or brand-specific)	String	Yes	Laptops, "Dell XPS 13"	
search_type	Type of search (generic or brand-specific)	String	Yes	generic, "brand"	
search_page_number	The page number of the search results	Integer	Yes	1, 2, etc.	
search_rank	Rank position of the product in the search results	Integer	Yes	1, 2, 3, etc.	
search_results_type	Marks whether the result is paid (sponsored) or organic	String	Yes	paid, "organic"	
search_timestamp	Date and time of the search scraping	Timestamp	Yes	2024-10-18 10:00:00	
retailer_id	Unique ID for the retailer	String	Yes	amazon_uk	
retailer_name	Name of the retailer	String	Yes	Amazon UK	
retailer_country	Country where the retailer is based	String	Yes	UK	
retailer_website	Website address for the retailer	String	Yes	https://amazon.co.uk	
product_id	Retailer-specific product ID	String	Yes	B08KFZLX7R	
product_link_url	URL linking to the product page	String	Yes	https://amazon.co.uk/product/B08KFZLX7R	

How to Integrate this with Existing Schema:
Add this as a separate table in the database schema to store search result data.
Link this table with the product table (if needed) via product_id or retailer_id to ensure data consistency across product listings and search results.

Retail Data Scraping Scope - Consumer Electronics							
Data Element	**Description**	**Required Format**	**Main Product Detail Page**	**Rich Content Section**	**Homepage | Category Page**	**Brand Shop**	**Selenium Required**
Timestamp	Time and date when the data was scraped.	Timestamp	✔				No
Currency	Currency in which the price is listed (e.g., USD, GBP).	String	✔				No
Retailer Country	Country where the retailer operates (e.g., US, UK).	String	✔				No
Retailer Website Address	The website URL of the retailer (e.g., walmart.com).	String	✔				No
Product Page							
Retailer Product ID	Unique identifier for the product on the retailer's site.	String	✔				No
product_page_url	URL of the product page	string	✔				
Manufacturer EAN/UPC	Manufacturer's unique product code (EAN/UPC) for global identification.	String	✔				No
Manufacturer Name	Name of the manufacturer (e.g., Lenovo, HP).	String	✔				No
Brand Name	Specific product brand name that includes the manufacturer and product line (e.g., Acer Swift Go, Apple iPhone).	String	✔				No
Category Name	The category under which the product is listed (e.g., laptops, desktops, tablets, smartphones).	String	✔				No
Category_Rank	Rank position of the product in the category	Integer (1, 2, 3, etc.)			✔		
Manufacturer Processor Name	Name of the processor manufacturer (e.g., Intel, AMD, SnapDragon).	String	✔				No
Manufacturer Processor Brand Name	The specific processor brand and model (e.g., Intel® Core™ i5, AMD Ryzen™ 7).	String	✔				No
Operating Platform Name	Name of the operating platform (e.g., Google Chrome, Microsoft Windows 11).	String	✔				No
Software Included	Any software included with the product (e.g., Windows 360 Trial, McAfee Anti-Virus Trial).	Text	✔				No
Out of Stock	Track the availability status (in-stock or out-of-stock) for each SKU on the retailer's site.	Boolean (In Stock/Out of Stock)	✔				No
Price	Capture both the list price and promotion price for the product.	Numeric (with currency symbol, e.g., $)	✔				No
Promotion Details	Extract any promotion details such as percentage off, money off, bundles, loyalty points, etc.	Text (e.g., "20% off", "Buy 1 Get 1 Free")	✔				No
Product Title	Scrape the full product title, ensuring it includes proper branding, such as Intel® Core™ i5. If the title is "Intel i5," it is incorrect.	Text	✔				No
Product Description	Capture the complete product description, ensuring all required keywords and brand mentions are present.	Text	✔				No
Badges	Badges displayed for the product (e.g., Amazon Choice, Best Seller, Clubcard Deal)	Text	✔		✔		
Bullet Points	Extract bullet points, ensuring they align with brand compliance and mention key features.	Text (list or paragraph format)	✔				No
Logos in Bullet Points	Scrape any brand logos (e.g., Intel, AMD, Windows) in the bullet points section.	Image URL	✔				No
Primary Image	Extract the URL for the primary product image used in the product listing.	Image URL	✔				No
Secondary Images	Extract URLs for all secondary images present in the product gallery.	Image URLs (list of URLs)	✔				No
Image Sequence	Ensure the correct sequence of images (primary followed by secondary) is captured.	Text/Ordinal (e.g., 1st, 2nd, 3rd)	✔				No
Complementary Products (Cross-Sells and Upsells)	Track any cross-sell or upsell products recommended on the product page.	Text/List of products	✔				No
Product Recommendations	Track any recommended products or bundles suggested alongside the main product.	Text/List of products	✔				No
Stock Availability (Click-and-Collect)	Capture any mention of click-and-collect availability.	Boolean (Available/Not Available)	✔				No
Product Page Screenshot	Capture a screenshot of the full product page, including rich content.	Image (PNG or JPEG)	✔	✔			No
Technical Specifications (e.g., Processor)	Extract specific technical details like processor type, RAM, storage from the product page.	Text	✔				No
video_url	URL of any video present on the product page	string	Yes	https://video.url/video.mp4			
Product Page (Grocery Specific)							
ingredients	List of ingredients in the grocery product	Text	✔				
nutritional_info_energy_per_100g	Energy per 100g (kcal)	Integer	✔				
nutritional_info_energy_per_serving	Energy per serving size (kcal)	Integer	✔				
nutritional_info_energy_percentage	Energy percentage of daily intake per serving	String	✔				
nutritional_info_fat_per_100g	Total fat per 100g (grams)	Decimal	✔				
nutritional_info_fat_per_serving	Total fat per serving (grams)	Decimal	✔				
nutritional_info_fat_percentage	Fat percentage of daily intake per serving	String	✔				
nutritional_info_saturates_per_100g	Saturates per 100g (grams)	Decimal	✔				
nutritional_info_saturates_per_serving	Saturates per serving (grams)	Decimal	✔				
nutritional_info_saturates_percentage	Saturates percentage of daily intake per serving	String	✔				
nutritional_info_carbohydrate_per_100g	Carbohydrate per 100g (grams)	Decimal	✔				
nutritional_info_carbohydrate_per_serving	Carbohydrate per serving (grams)	Decimal	✔				
nutritional_info_sugars_per_100g	Sugars per 100g (grams)	Decimal	✔				
nutritional_info_sugars_per_serving	Sugars per serving (grams)	Decimal	✔				
nutritional_info_sugars_percentage	Sugars percentage of daily intake per serving	String	✔				
nutritional_info_fiber_per_100g	Fiber per 100g (grams)	Decimal	✔				
nutritional_info_fiber_per_serving	Fiber per serving (grams)	Decimal	✔				
nutritional_info_protein_per_100g	Protein per 100g (grams)	Decimal	✔				
nutritional_info_protein_per_serving	Protein per serving (grams)	Decimal	✔				
nutritional_info_salt_per_100g	Salt per 100g (grams)	Decimal	✔				
nutritional_info_salt_per_serving	Salt per serving (grams)	Decimal	✔				
nutritional_info_salt_percentage	Salt percentage of daily intake per serving	String	✔				
nutritional_info_type	Specifies if the data is for adult or child	String	✔				
country_of_origin	Country where the product is produced	String	✔				
allergens	List of allergens in the product	Text	✔				
dietary_claims	Dietary labels associated with product	Text	✔				
serving_size	Standard serving size	Text	✔				
storage_instructions	Instructions for storage	Text	✔				
preparation_instructions	Steps for preparation	Text	✔				
package_type	Material type of the product packaging	String	✔				
recyclable	Indicates if the packaging is recyclable (Yes/No)	Boolean	✔				
promotion_type	Type of promotion (e.g., loyalty card, multi-buy, price match, subscription)	String	✔				
promotion_description	Detailed description of the promotion	Text	✔				
promotion_discount	Discount amount or percentage (if applicable)	Decimal	✔				
promotion_price	Price after the promotion is applied	Decimal	✔				
promotion_conditions	Conditions for the promotion	Text	✔				
promotion_expiry	Expiration date for the promotion	Date	✔				
Brand Shops							
brand_shop_id	Unique identifier for the brand shop	String	Yes	intel_01		✔	
brand_shop_name	Name of the brand shop	String	Yes	Dell Brand Shop		✔	

brand_shop_url	URL of the brand shop homepage	String	Yes	https://amazon.com/dell		✔	
brand_shop_retailer_id	Unique identifier for the marketplace	String	Yes	amazon_us		✔	
is_in_brand_shop	Flag indicating if product is in the brand shop	Boolean	Yes	TRUE		✔	
brand_shop_product_id	Retailer-specific product ID from brand shop	String	Yes	B08KFZLX7R		✔	
brand_shop_product_title	Title of the product from brand shop	String	Yes	Dell XPS 13		✔	
brand_shop_product_price	Price of the product from brand shop	String	Yes	$999.99		✔	
brand_shop_product_rating	Rating score of the product from brand shop	Float	Yes	4.5		✔	
brand_shop_product_review_count	Number of reviews for the product from brand shop	Integer	Yes	1500		✔	
brand_shop_primary_image_url	URL of the primary image for the product	String	Yes	https://example.com/image1.jpg		✔	
brand_shop_secondary_images	Array of secondary images for the product	Array	Yes	["https://example.com/image2.jpg", "https://example.com/image3.jpg"]		✔	
brand_shop_out_of_stock	Flag indicating if the product is out of stock	Boolean	Yes	FALSE		✔	
brand_shop_categories	Array of product categories within the brand shop	Array	Yes	["Laptops", "Gaming Laptops", "Desktops"]		✔	
Best Seller Ranking							
best_seller_rank	Product's rank in the best sellers list	Integer	Yes	1, 2, 3, etc.			
best_seller_ranking_share_of_voice_in_top_100	Percentage of brand's products in the Top 100 Best Sellers	Percentage	Yes	10%			
best_seller_ranking_marketplace	Marketplace for the best seller ranking data	String	Yes	Amazon_uk			
best_seller_rank_brand_name	Brand name as it appears in the Best Seller Ranking	String	Yes	Dell			
best_seller_rank_node_url	URL of the best seller ranking node on Amazon	String	No	https://amazon.com/...			
best_seller_ranking_timestamp	Date and time of the best seller ranking scraping	Timestamp	Yes	2024-10-18 10:00:00			
Rich Content							
Rich Content Displayed (Y/N)	Identify whether rich content is displayed on the product page.	Boolean (Yes/No)	✔	✔			No
Videos in Rich Content	Identify and extract any videos embedded in the rich content section of the product page.	Video URL		✔			No
Rich Content Images	Extract the number of images and their URLs present in rich content.	Image URLs (list of URLs)		✔			No
Mentions of Intel/AMD/SnapDragon	Count how many times brands like Intel, AMD, and SnapDragon are mentioned.	Numeric (Count)		✔			No
Intel/AMD/SnapDragon Logos in Rich Content	Track Intel, AMD, and SnapDragon logos in rich content.	Image URL		✔			No
Other Brand Logos in Rich Content	Track logos of other brands that appear in rich content (e.g., Microsoft Windows, NVIDIA).	Image URL		✔			No
Complementary Accessories in Rich Content	Capture any complementary accessories featured in rich content.	Text (list of accessory names)		✔			No
Ratings & Reviews							
Customer Reviews (Full Text)	Capture the entire customer review text for sentiment analysis.	Text (full review)	✔				No
Customer Ratings	Scrape the rating score (e.g., out of 5 stars).	Numeric (out of 5)	✔			✔	No
Review Date	Extract the date when the review was posted.	Date (e.g., YYYY-MM-DD)	✔				No
Number of Helpful Votes	Capture how many users found the review helpful.	Numeric (count of helpful votes)	✔				No
Sentiment Analysis Keywords	Extract key terms from customer reviews, such as performance, reliability, battery life, for sentiment analysis.	Text (list of keywords)	✔				No
Banners							
Banner Tracking	Capture banner ads displayed on the homepage, category pages, and keyword-based searches, including banner URLs and text.	Image URL, Text			✔	✔	No
Banner Brand Detection	Detect the brands displayed on banner ads (e.g., Intel Evo, AMD, etc.).	Text			✔	✔	No
Banner Type	Identify the banner type, whether it is static or dynamic.	Text			✔	✔	No
Banner Page Screenshot	Capture a screenshot of the page with banners displayed.	Image (PNG or JPEG)			✔	✔	No
Banner Destination URL	Track the destination URL when a banner ad is clicked.	URL			✔	✔	No
Banner Link URL	Track the link URL associated with the banner ad.	URL			✔	✔	No
Banner Brands in Carousel	Capture and identify brands in each banner of a carousel (e.g., banner 1: Intel, banner 2: AMD, banner 3: Apple, banner 4: Non-PC products, banner 5: Mixed products like a bundle of different items).	Text/List of brands by banner	✔			✔	No
Basket Conversion							
Add to Basket Recommendations	Identify and capture recommended products or complementary items suggested after adding a product to the basket.	Text, Image URLs	✔				Yes (Selenium)
Search Ranking							
search_keyword	The keyword used in the search (generic or brand-specific)	String	Yes	Laptops, "Dell XPS 13"			
search_type	Type of search (generic or brand-specific)	String	Yes	generic, "brand"			
search_page_number	The page number of the search results	Integer	Yes	1, 2, etc.			
search_rank	Rank position of the product in the search results	Integer	Yes	1, 2, 3, etc.			
search_results_type	Marks whether the result is paid (sponsored) or organic	String	Yes	paid, "organic"			
search_timestamp	Date and time of the search scraping	Timestamp	Yes	2024-10-18 10:00:00			
retailer_id	Unique ID for the retailer	String	Yes	amazon_uk			
retailer_name	Name of the retailer	String	Yes	Amazon UK			
retailer_country	Country where the retailer is based	String	Yes	UK			
retailer_website	Website address for the retailer	String	Yes	https://amazon.co.uk			
product_id	Retailer-specific product ID	String	Yes	B08KFZLX7R			
product_link_url	URL linking to the product page	String	Yes	https://amazon.co.uk/product/B08KFZLX7R			

Key Points:

Search Keyword: The term used in the search (generic or brand).

Search Type: Whether the search was for a general category (e.g., "laptop") or a specific brand (e.g., "Intel processor").

Search Page Number: The page number of search results collected (either page 1 or page 2).

Search Rank: The product's rank in the search results for the given keyword.

Product Information: The basic details of the product, including ID, product url

Retailer Information: Details of the retailer that displayed the product in the search results.

Search Timestamp: The time and date when the search data was scraped.

Role	Responsibility	Key Actions	Tools/Technologies
Web Scraping Team	Responsible for setting up scraping tasks to collect data from both category sections and brand shops. They will also need to implement cloud storage logic to back up brand shop data.	- Category Section Scraping: Continue scraping product data from the category sections of retailer sites (e.g., Laptops on Amazon). - Brand Shop Scraping: Parallel scraping for products in brand shops (e.g., Dell Brand Shop) and storing data in a separate cloud storage. - Cloud Storage Setup: Store brand shop data in a designated cloud (e.g., AWS S3, Google Cloud Storage) to be used as backup. - Deduplication Logic: Ensure the use of retailer_product_id or manufacturer_ean_upc to avoid duplication when filling gaps in category data. - Schedule Backups: Regularly backup scraped brand shop data to the cloud. - Integration with Main DB: Ensure backup data can be pulled into the main database if category data fails or is incomplete.	- Scrapy, Selenium (for web scraping) - Cloud storage (AWS S3, Google Cloud Storage, Azure) - API for data ingestion
Data Engineer	In charge of managing the ETL pipeline for both category and brand shop data and ensuring seamless integration between the cloud-stored backup data and the main database.	- Data Ingestion: Set up a process for ingesting scraped category data into the main database. - Backup Retrieval: Create an API or process for retrieving brand shop data from cloud storage when needed. - Deduplication Logic: Implement logic that ensures there is no duplication between category data and backup brand shop data. - Database Integration: Work with the full stack team to integrate backup data into the PHP Laravel backend. - Data Quality Checks: Ensure that product information retrieved from the brand shop is accurate and that the correct version (either category or brand shop) is used when pulling into the database. - Error Handling: Set up error handling in case of missing category data, triggering the brand shop backup pull. - Tracking/Monitoring: Track the success of data retrieval processes and monitor for data consistency.	- Python (Pandas, NumPy) for data processing - SQLAlchemy or PyMySQL for database integration - API for cloud data retrieval - Database Management (MySQL, PostgreSQL)
Full Stack Developer	Responsible for setting up and managing the PHP Laravel backend database and ensuring it can handle both category and brand shop backup data ingestion.	- Database Schema Implementation: Ensure the updated database schema accommodates both category and brand shop data. - Database Integration: Work closely with the data engineer to ensure smooth ingestion of category and backup data into the PHP Laravel database. - API Setup: If needed, set up APIs to facilitate the backup data ingestion. - Backup Handling: Create mechanisms in the backend to retrieve brand shop data when category data is missing or incomplete. - Data Deduplication: Ensure that database entries are only created for products that are not already stored, and update entries if they appear in both sources (category and brand shop). - Data Tracking: Track where the data comes from (category or brand shop) using fields like is_in_brand_shop.	- PHP Laravel (Backend framework) - SQL Database (MySQL/PostgreSQL) - API Setup for data handling
DevOps/Cloud Engineer	Responsible for setting up and managing cloud infrastructure to store and manage brand shop backup data. They will also ensure cloud storage is secure, scalable, and easy to access for data retrieval.	- Cloud Infrastructure Setup: Set up scalable and secure cloud storage (e.g., AWS S3, Google Cloud Storage) to store brand shop data. - Data Backup Scheduling: Automate the regular backup of brand shop data to cloud storage. - Access Control: Ensure proper access controls for cloud storage, granting access to the scraping team, data engineer, and backend team. - API Setup: Work with the data engineer to set up APIs to retrieve data from cloud storage when needed. - Monitoring and Alerts: Set up monitoring for cloud data storage to ensure data is backed up and accessible when needed.	- AWS S3 / Google Cloud Storage / Azure Blob Storage for backup - Access Management (IAM) - API Integration with cloud services
Project Manager	Oversees the entire operation to ensure proper coordination among the web scraping team, data engineer, full stack developer, and DevOps engineer. Ensures the timeline is adhered to and the goals are met.	- Task Coordination: Ensure that each team is clear on their responsibilities and timeline. - Monitor Progress: Track progress on web scraping, data integration, and cloud backup setups. - Communication: Facilitate communication between the scraping team, data engineer, and full stack developer. - Risk Management: Identify risks such as data scraping failures and ensure backup systems are in place and functioning.	- Project Management Software (Asana, Jira) - Slack/Teams for communication - Google Docs/GitHub for documentation

Proposed Workflow:	Next Steps:
Primary Scraping: Scrape products from category sections as the primary data source.	Cloud Setup: Set up the cloud storage for brand shop data backups (e.g., AWS S3 buckets).
Parallel Brand Shop Scraping: Scrape products from brand shops in parallel and store this data in separate cloud storage as backup.	Data Ingestion Logic: Add logic to pull from brand shop backups when category scraping fails or data is incomplete.
Gap-Filling: If category scraping fails or misses data, trigger a process to pull products from brand shop data stored in the cloud.	Monitor Data Integrity: Ensure that data deduplication and merging processes are set up properly, so there's no overlap between the primary and backup data sources.
De-Duplication and Merge: Ensure no duplication by using unique identifiers (e.g., retailer_product_id or manufacturer_ean_upc). If a product exists in both sources, update the main database entry with the most complete data from either source.	
Automated Monitoring: Set up automated monitoring to track the success of category scraping and trigger backup retrieval when necessary.	

Stage	Task	Details/Requirements	Example Payload/Response	Notes
1. Web Scraping (Scrapy | Browse AI)	Structured Data Delivery via API	Ensure all scraped data is sent to the backend via API in a structured format (JSON or CSV preferred).	JSON example: { "retailer_id": "101", "product_id": "XYZ123", "price": 499.99, "primary_image_url": "..." }	**JSON preferred for flexibility**: CSV can be used for flat data but may not handle rich content and images well.
	Segment Data by Sections	Separate the scraped data into Main Product Page, Rich Content Section, and Homepage/Category Pages.	Example Segments: { "main_product_data": { ... }, "rich_content_data": { ... }, "homepage_banner_data": { ... } }	Helps organize data for different areas of the page. This ensures backend and multimodal tasks can be run more efficiently.
	Screenshots	Capture screenshots of banners, product pages, and rich content sections.	"product_page_screenshot_url": "https://example.com/screenshot.jpg"	Include the screenshot file paths/URLs within the structured data for easy retrieval by multimodal models.
	Error Handling and Logging	Implement retry mechanisms for failed API requests and log successes and failures.	Logs should track failed deliveries and retry attempts for failed API requests.	Ensures reliable delivery of data from Scrapy to backend.
	Frequency and Delivery Method	Decide between real-time API requests or batch delivery of scraped data based on requirements.	Real-time example: API requests after each scrape. Batch example: API delivery every hour or daily.	Real-time for high-frequency updates, batch mode for periodic data collection.
2. Multimodal Processing (API)	Title Compliance Check	Send the scraped product title to the multimodal API to check for compliance with brand standards (e.g., Intel® Core™ i5).	JSON Request: { "product_title": "Intel® Core™ i5 Laptop", "expected_title_format": "Intel® Core™ i5" }	Ensures compliance of product titles. The API returns a compliance score or status for the backend to store.
	Image Matching (Primary and Secondary)	Send image URLs to the multimodal API for comparison with the brand's source of truth images.	JSON Request: { "primary_image_url": "https://example.com/image1.jpg", "source_image_url": "..." }	The API compares images using visual recognition models and returns a match score or status.
	SKU Matching (EAN/UPC)	Send the EAN/UPC codes to the multimodal API to match across retailers and ensure product consistency.	JSON Request: { "ean_upc": "1234567890123", "retailer_sku": "XYZ123" }	The API checks SKU consistency between different retailers using the EAN/UPC code, returning a match result.
	Banner Tracking	Send banner data (e.g., brand detection, type, and URLs) to the multimodal API for analysis of SOV and brand exposure.	JSON Request: { "banner_type": "static", "banner_brand": "Intel Evo", "banner_destination_url": "..." }	The API will return results on banner brand compliance, visibility, and correct links for further processing.
	Multimodal Result Storage	Store results of content compliance, image matching, and SKU matching in the backend database.	JSON Response: { "title_compliance": true, "image_match": 95, "sku_match": "matched" }	Results can be stored in the backend for dashboard presentation or further analysis.
3. Backend Integration and Flow	Receive and Structure Scraped Data	API should ingest structured data from Scrapy and store it in the backend for multimodal processing.	API Request: JSON { "retailer_id": "101", "product_id": "XYZ123", "rich_content_images": ["..."] }	The backend API should handle structured JSON payloads from the scraper to prepare for multimodal processing.
	Send Data to Multimodal APIs	After receiving scraped data, the backend can call the multimodal API for content and image processing.	Backend sends structured data via API requests to multimodal systems for compliance and matching checks.	This step ensures that multimodal elements are processed in real-time or batch based on the data received from Scrapy.
	Store Multimodal Results	Once the multimodal API returns compliance/matching scores, store these in the database.	API Response: JSON { "compliance_score": 98, "image_match": "perfect" }	The backend should capture and store the results for further analysis, visualization, or alerts for the brand.
4. Source of Truth Content (API or CSV)	Source of Truth Data Ingestion	Brands provide product images and information via PIM connection or CSV file.	PIM Connection: JSON { "product_id": "XYZ123", "title": "Acer Swift Go", "primary_image_url": "..." }	The backend should store the source of truth data in separate tables for images and product information for compliance checks.
	API or Batch CSV Ingestion	If PIM integration is available, set up an API to regularly ingest source of truth data. If CSV, build a parsing pipeline.	API Example: { "product_id": "XYZ123", "images": [{ "image_url": "..." }], "product_title": "Acer Swift Go" }	Source of truth content is ingested as a batch file (CSV) or real-time via API and stored in the database for later compliance checks.
	Compliance and Matching	Compare scraped data (from retailers) to the source of truth for content compliance, image matching, and SKU matching.	API/Batch Ingestion: Compare product titles, descriptions, and images to the brand's source of truth data.	The results of the comparison should be stored for further analysis in the backend and displayed in dashboards or reports.
	PIM Connection ID for Traceability	For PIM integration, store the pim_connection_id to ensure traceability back to the original source.	Example PIM ID: "PIM-12345"	PIM Connection ID helps link the product content to its original source in the PIM system for auditing or verification purposes.

Key Points:

- **API Flow:** The scraping team delivers data via API in structured JSON format, making it easier for the backend to process.
- **Multimodal Analysis:** The multimodal models process content compliance, image matching, and SKU matching, and return structured results via API.
- **Error Handling:** Implement retries and logging for API requests to ensure robust delivery.
- **Real-Time vs. Batch:** Depending on requirements, data can be processed in real-time or as batch jobs.

Data Engineer Role & Responsibilities		
Responsibility	**Description**	**Tools/Technologies**
Leveraging Scraped Data	Utilize data collected by the specialist scraping team using Scrapy, Selenium, and source of truth data (ICECAT) for further processing, transformation, and analysis.	Scrapy, Selenium, ICECAT, Pandas, NumPy
Data Cleaning and Preprocessing	Clean and normalize the scraped data to ensure consistency and accuracy (e.g., removing duplicates, handling missing data).	Pandas, NumPy
Database Connectivity	Connect Python scripts to the PHP Laravel back-end database using PyMySQL or SQLAlchemy.	PyMySQL, SQLAlchemy, MySQL/PostgreSQL
Data Insertion (ETL)	Implement efficient ETL processes to insert and update data in the Laravel database, ensuring schema adherence.	SQLAlchemy, MySQL/PostgreSQL
Collaboration with Full Stack Team	Coordinate with the full stack team on database access, APIs, and data requirements, ensuring compatibility with the simple Bootstrap interface.	GitHub, Email, Asana/Jira
Automation of Data Pipelines	Schedule data processing jobs using tools like cron or Airflow, ensuring timely data integration into the back-end database.	cron, Airflow
Error Handling and Monitoring	Implement error handling and logging for pipeline monitoring and quick issue resolution.	Python logging, custom scripts
SKU Matching	Perform multimodal SKU matching by comparing product titles and images to retailer product codes, manufacturer part numbers, and ICECAT data.	Python (custom scripts), NLP, ICECAT
Data Quality Assurance	Perform validation checks on data accuracy, such as SKU verification and image matching against the source of truth (e.g., ICECAT).	Python, Pandas, custom QA scripts
Collaboration with Multimodal Freelancer	Provide clean data (images, text) for multimodal analysis; incorporate feedback into the pipeline.	Email, Shared Python Libraries
Data Pipeline Scalability	Design scalable data pipelines that can accommodate new retailers and increased data volume.	Python, Prefect, SQLAlchemy
Documentation and Reporting	Maintain clear documentation of all scripts, processes, and methodologies, and report progress to stakeholders.	GitHub, Markdown, Asana/Jira
Data Privacy and Compliance	Ensure all data collection and processing comply with relevant data protection regulations (e.g., GDPR).	Python, Compliance Checklists
Version Control and Code Management	Manage and track changes to scripts using Git for collaboration and transparency.	Git, GitHub
Progress Updates	Provide regular updates on data collection progress, issues, and resolutions to the team and stakeholders.	Email, Asana/Jira, GitHub Issues
API Development Support	Collaborate with the full stack team to develop or consume APIs for data insertion or retrieval.	REST APIs, SQLAlchemy
Skill	**Description**	**Tools/Technologies**
Python Programming	Proficiency in Python for data processing, analysis, and automation tasks.	Python
Leveraging Scraped Data	Ability to work with data provided by Scrapy, Selenium, and ICECAT scraping tools for transformation and analysis.	Scrapy, Selenium, ICECAT, Pandas, NumPy
Database Integration	Knowledge of connecting Python to databases used by PHP Laravel for data insertion and updates.	PyMySQL, SQLAlchemy, MySQL/PostgreSQL
Data Processing and Cleaning	Ability to clean, normalize, and process data for ETL pipelines.	Pandas, NumPy
Error Handling and Logging	Ability to implement logging for monitoring pipeline performance and identifying issues.	Python logging
Version Control	Proficiency in Git for code versioning and repository management.	Git, GitHub
Automation and Scheduling	Experience with scheduling tools to automate data processing jobs.	cron, Airflow
Collaboration Tools	Familiarity with project management and communication tools for coordinating with the development team using a Bootstrap interface.	Asana, Jira, Email, GitHub
Compliance Awareness	Understanding of legal and ethical data handling and data protection regulations (e.g., GDPR).	Compliance Guidelines
Deliverable	**Description**	
Data Pipelines	Fully functional and automated data processing scripts using Python and connected to the PHP Laravel back-end, aligned with the front-end Bootstrap interface.	
Clean Data Sets	Structured data inserted into the back-end database according to the defined schema.	
Documentation	Clear and detailed documentation of code, processes, and methodologies.	
Progress Reports	Regular updates on data collection, including any challenges and resolutions.	
Test Phase Priorities		

Multomodal Engineer Role & Responsibilitie					
Digital Shelf Data Element	**Multimodal Use Case**	**Multimodal Techniques**	**Main Product Page**	**Rich Content Section**	**Homepage/Category Page**
Retailer Product ID	Used to link scraped product data across retailers, matching multimodal elements like images and titles to unique product identifiers.	Image recognition, text matching algorithms	✔		
Manufacturer EAN/UPC	Essential for matching global SKUs across retailers to verify product consistency.	Text matching, SKU comparison	✔		
Product Page					
Product Title	Ensures product title compliance by checking correct brand and processor names are used (e.g., Intel® Core™ i5 vs. Intel i5).	NLP for title compliance, sequence matching algorithms	✔		
Primary Image	Image compliance checks to ensure that the primary product image matches the brand's source of truth.	Image matching (multimodal models), visual similarity analysis	✔		
Secondary Images	Ensure all secondary images are present and in the correct sequence, matching against the source of truth.	Image comparison, sequence validation using image hashing	✔		
Bullet Points	Extract bullet points ensuring compliance with brand messaging and key features.	Text Compliance: Verify bullet points match the source of truth, check for key feature mentions (e.g., "Intel® Evo™ platform").		✔	
Mentions of Intel/AMD/SnapDragon	Count and analyze the mentions of Intel, AMD, SnapDragon, etc., within product pages and banners to evaluate brand prominence.	NLP for brand mentions, keyword extraction algorithms	✔	✔	✔
Brand Logos in Bullet Points	Check for the presence and correct display of brand logos (e.g., Intel, AMD, Windows) in product bullet points and compare against source of truth.	Image detection, OCR to verify text alongside logos	✔		
Complementary Products (Cross-Sells)	Analyze cross-sell and upsell recommendations presented on the product page and determine which brands are frequently suggested together.	NLP to detect product associations, image recognition for complementary product visual identification	✔		
Technical Specifications	Ensure key technical specifications (e.g., processor, RAM) are consistent with the brand guidelines and comply with accuracy standards.	Text comparison, specification matching algorithms	✔		
Product Page Screenshot	Capture full-page screenshots to check for the presence of all required content, including images, videos, and rich content elements.	Image capture, comparison of full page layout	✔	✔	
Ratings & Reviews					
Customer Reviews (Full Text)	Sentiment analysis on customer reviews to understand product feedback, highlighting key pain points or positive aspects like performance.	NLP, sentiment analysis	✔		
Customer Ratings	Track and compare product ratings versus competitor products to identify consumer sentiment towards the product.	Statistical comparison, sentiment analysis	✔		
Banners					
Banner Brand Detection	Detect the presence of brand logos in banners to measure share of voice in banner ads (e.g., Intel Evo, AMD).	Image recognition, OCR for detecting logos and text			✔
Banner Type	Identify whether banners are static or dynamic to evaluate the type of promotion and branding strategy used.	Video/image analysis for static vs. dynamic content differentiation			✔
Banner Page Screenshot	Analyze the banner placement on the page to ensure proper positioning of ads in relation to competing brands.	Image comparison, page layout analysis			✔
Banner Destination URL	Check whether banner ads are linking to the correct destination as part of compliance analysis.	URL comparison, link verification algorithms			✔
Banner Link URL	Ensure the banner links to relevant pages, ensuring correct ad-to-content destination tracking.	Text matching, URL validation			✔
Banner Brands in Carousel	Evaluate the sequence of banners in carousels and check the visibility of each brand (e.g., banner 1: Intel, banner 2: AMD, etc.).	Image recognition, sequence validation (for carousels), brand logo detection			✔
Rich Content					
Rich Content Images	Ensure compliance by verifying that all rich content images are present and match the brand's content guidelines.	Image comparison, visual content matching		✔	
Videos in Rich Content	Identify any videos embedded in the rich content section.	Video Presence Check: Detect and verify if the video is present, ensure it's the correct video as per the source of truth.	✔		
Mentions of Intel/AMD/SnapDragon	Count how many times processor brands are mentioned in the rich content.	Text Counting: Use natural language processing (NLP) to count mentions of key brands (Intel, AMD, SnapDragon) and verify compliance.	✔		
Intel/AMD/SnapDragon Logos in Rich Content	Track logos of processor brands in rich content (Intel, AMD, SnapDragon).	Logo Detection: Ensure the presence of correct processor logos (e.g., Intel, AMD) and validate their positioning in the content.	✔		
Other Brand Logos in Rich Content	Identify logos of other brands appearing in the rich content (e.g., Microsoft Windows, NVIDIA).	Co-Branding Analysis: Detect and analyze other brand logos (e.g., Microsoft Windows) and their positioning in co-brand strategies.	✔		
Complementary Accessories in Rich Content	Capture complementary accessories featured in the rich content (e.g., stylus, mouse).	Accessory Detection: Identify and check if all relevant accessories are showcased in the rich content in compliance with brand strategy.	✔		
Complementary Products (Cross-Sells and Upsells)	Track cross-sell or upsell products suggested in the rich content section.	Cross-Sell/Upsell Analysis: Analyze and verify suggested cross-sells/upsells are aligned with brand or retailer strategy.	✔	✔	
Database Enrichment					
SKU Matching	Check for consistency between product titles, images, and retailer product codes against manufacturer part numbers (including ICECAT validation).	SKU Matching: Leverage multimodal techniques to ensure that product titles, images, and codes match the manufacturer part number, ensuring data normalization for SKU-level analysis.	✔	✔	
Test Phase Priorities					

Its essential to clarify how data will flow between the "Web Scraping with Browse AI", "Multimodal Content Compliance", "Image Matching", and "SKU Matching"
Data Flow Clarification:
1. Web Scraping (Browse AI or Scrapy):
- Browse AI or Scrapy will scrape the data from retailer websites.
- The scraped data will be delivered in a "structured format" (likely JSON) to the back end via API.
- The data will be structured to include key digital shelf metrics (e.g., product titles, images, pricing, and other details).
2. Storing Structured Data in the Backend:
- Yes, the data will arrive structured from the scraping process, and the back-end team's responsibility will be to store this structured data in the database (using PHP Laravel).
- The structured data will include:
- Product information: Titles, descriptions, technical specs, images, prices.
- Image URLs: Both primary and secondary images, along with banners or rich content.
- Other elements: Stock status, reviews, etc.
3. Multimodal Processing (Content Compliance, Image Matching, SKU Matching):
- After the structured data is stored in the database, "Multimodal APIs" will be called from the backend to process certain data for compliance checks and matching:
- Content Compliance: Checking if product titles, descriptions, and bullet points match the brand's source of truth.
- Image Matching: Sending image URLs to an API that checks if the images scraped match the brand's source images.
- SKU Matching: Ensuring that the scraped product SKUs (or EAN/UPC) match the global brand's product codes.
4. Interaction with the Backend:
- API-based Interaction: The structured data scraped will "arrive via an API", and the backend will handle "storage" and "additional API requests" to the multimodal content compliance and image matching systems.
- For example:
- Once the product data (e.g., title, images) is scraped and stored, the back end will make a "POST request" to a multimodal API endpoint for "content compliance" or "image matching"
- The results (e.g., compliance score, image match score) are returned from the multimodal API and stored back in the **PHP Laravel database**.
Simplified Workflow:
1. Web Scraping (Browse AI or Scrapy):
- Scraped data delivered in JSON → Sent to the backend via API.
2. Backend Storage:
- Backend stores scraped data in the **PHP Laravel** database.
3. Multimodal Processing:
- Backend calls "Multimodal APIs" for content compliance, image matching, and SKU matching.
- The multimodal results are stored back in the database for further analysis or reporting.
Key Points:
- The data will arrive structured from the scraping service, and the backend team will primarily need to store it and manage subsequent API calls for multimodal analysis.
- If "data structuring" is needed (e.g., parsing CSV files or restructuring JSON), that would be part of the back-end logic. However, the multimodal tasks are API-driven.