# AI-Powered Python Application for Natural Language E-Commerce and Flight Data Extraction

## Overview

This application processes natural-language user queries (e.g., product searches, flight price requests, cross-site price comparisons) and produces comprehensive Excel reports with relevant data. It leverages state-of-the-art AI for query understanding and the Model Context Protocol (MCP) for robust browser-driven multi-site automation.

## Key Features

* Natural Language Query Processing: Handles diverse query types—from product searches to flight price lookups and cross-platform comparisons.
* MCP Browser Automation: Uses Playwright or Selenium with MCP for precise, scriptable browser control (navigation, form-filling, dynamic content handling).
* Multi-Site Scraping: Extracts structured data from Amazon, Flipkart, major travel booking sites, and more.
* Excel Report Generation: Produces formatted Excel files with dynamic sheets, data tables, analysis, and charts customized per query type.

## Architecture

## 1. AI Query Interpretation

* Utilizes OpenAI, Anthropic, or local LLMs to parse user queries and:
  + Identify intent (product search, flight search, price comparison)
  + Determine relevant sites (Amazon, Flipkart, travel portals)
  + Extract search criteria (item, price, city, dates, etc.)

## 2. Automated Browser Workflows

* MCP-driven commands: Automates the browser with human-like precision:
  + Navigation (landing page, category, results)
  + Search form interaction (type, select, submit)
  + Filter application (price, brand, rating)
  + Dynamic loading (scroll, AJAX, pagination)
  + Error handling (popups, CAPTCHA, fallbacks)

## 3. Data Extraction & Aggregation

* Structured scraping: Extracts relevant data into Python structures, including:
  + Product: Title, Price, Image, Rating, Reviews, Features, Links
  + Flights: Airline, Departure/Arrival, Dates, Prices, Booking Links
  + Multi-site: Normalizes and aggregates fields across platforms

## 4. Excel Report Generation

* Uses pandas/OpenPyXL/xlsxwriter to:
  + Create dynamic sheets per query/website
  + Populate filterable tables with itemized data
  + Generate summary analytics and charts (price distribution, comparison plots)
  + Provide organization (site, product, flight, date)

## Workflow Examples

## A. Product Search (e.g., "Find me laptops under ₹50,000")

* Step 1: User enters query.
* Step 2: LLM identifies intent: product search, "laptop," price ≤ ₹50,000, targets: Amazon, Flipkart.
* Step 3: Browser automation:
  + Navigate to Amazon → Fill search → Apply price filter → Extract results
  + Repeat for Flipkart.
* Step 4: Aggregate data, normalize fields.
* Step 5: Create Excel with:
  + Sheets: "Amazon", "Flipkart"
  + Columns: Title, Price, Rating, Reviews, Link, Features
  + Summary charts: Price ranges, top brands.

## B. Flight Price Lookup (e.g., "What's the price of ticket from Bangalore to SF")

* Step 1: LLM identifies: Route search, "Bangalore" to "SF", target: travel sites.
* Step 2: Browser automation:
  + Visit MakeMyTrip → Fill flight search → Pick dates → Extract flight grid
* Step 3: Data extraction: Airlines, timings, layovers, prices, booking links.
* Step 4: Excel sheet for flights with sorting/filtering and chart of price distribution.

## C. Multi-Site Price Comparison ("Compare AirPods prices on different sites")

* Step 1: LLM parses: Product = AirPods, sites = Amazon, Flipkart, Croma.
* Step 2: Sequential/parallel site data extraction via MCP.
* Step 3: Normalization and feature matching.
* Step 4: Excel with product rows and price columns per site + comparison chart.

## Technical Components

## Core Python Modules

* LLMs: openai, anthropic, or local via transformers
* MCP Automation: mcp-playwright, selenium-mcp, or REST MCP interface
* Data Extraction: BeautifulSoup, lxml, pandas
* Excel Generation: openpyxl, xlsxwriter, pandas
* Visualization: matplotlib, seaborn

## MCP Command Examples

* browser\_navigate('https://www.amazon.in')
* browser\_type('#searchBox', 'laptop')
* browser\_click('.price-filter')
* browser\_scroll('.results-list')
* browser\_snapshot()

## Error Handling

* Automatic dialog/alert closing
* CAPTCHA and access block detection
* Retry/fallback logic for partial failures

## Example Directory Structure

text

project-root/  
├── app/  
│ ├── llm\_interface.py  
│ ├── mcp\_client.py  
│ ├── extractors/  
│ │ ├── amazon.py  
│ │ └── flipkart.py  
│ ├── excel\_report.py  
│ ├── main.py  
│ └── utils.py  
├── requirements.txt  
├── README.md  
├── MCP\_AUTOMATION.md  
├── WORKFLOW\_EXAMPLES.md  
└── samples/  
 ├── laptops\_under\_50K.xlsx  
 ├── bangalore\_to\_sf\_flights.xlsx  
 └── airpods\_prices\_comparison.xlsx

## Sample Output (Excel)

* Product Sheet Columns: Name, Price, Rating, Review Count, Features, Shop, Product Link
* Flight Sheet Columns: Airline, Departure, Arrival, Duration, Stops, Price, Booking Link
* Charts/Analytics: Price distribution, best-rated options, site-by-site comparison

## Setup Requirements

* Python 3.8+
* Dependencies: AI SDKs, MCP library, Excel writers (see requirements.txt)
* LLM access: OpenAI / Anthropic API key or local LLM configuration
* Documentation: Detailed in README.md, browser automation in MCP\_AUTOMATION.md, hands-on workflows in WORKFLOW\_EXAMPLES.md.

## Performance and Extendability

* Modular, extensible codebase for new sites and query types
* Advanced prompt/project error handling and verification
* Dynamic Excel output for full user value (filter, sort, analyze right away)

This system template enables rapid, intelligent extraction and reporting on e-commerce and travel data, providing users with actionable insights in a familiar Excel format.