AI Research Agent: Technical Design Document

1. Problem Framing & Assumptions

Problem Statement

Building an autonomous research agent that transforms vague user queries into comprehensive, well-sourced research reports—essentially automating the research workflow of a human analyst.

Repo Link

https://github.com/weshallchat/AI-Research-Agent.git

Quick Start:

Prerequisites:

- Python 3.9+
- OpenAI API key
- Serper API key

Installation:

```
# Clone the repository
git clone https://github.com/weshallchat/AI-Research-Agent.git
cd AI-Research-Agent

# Create virtual environment
python3 -m venv venv
source venv/bin/activate  # On Windows: venv\Scripts\activate

# Install dependencies
pip install -r requirements.txt

# Set up environment variables
echo "OPENAI_API_KEY=your_api_key_here" > .env
```

Run the Application:

Option 1: Gradio UI (recommended)

```
python app.py
```

Then open the URL shown in the terminal (Ex: https://2ede52a197d64e8c60.gradio.live).

Markdown reports can be found in AI-Research-Agent/output folder.

Option 2: Run Tests

```
python tests/run_all_tests.py
```

More test cases can be found under AI-Research-Agent/tests folder.

Key Challenges Addressed

- Query Ambiguity: Users often ask poorly-formed or vague questions ("AI bad for jobs?")
- Information Quality: Need to gather credible sources, not just any web results
- Relevancy Validation: Generated research plans may not align with user intent
- Synthesis Complexity: Raw information must be distilled into coherent, actionable reports

Assumptions

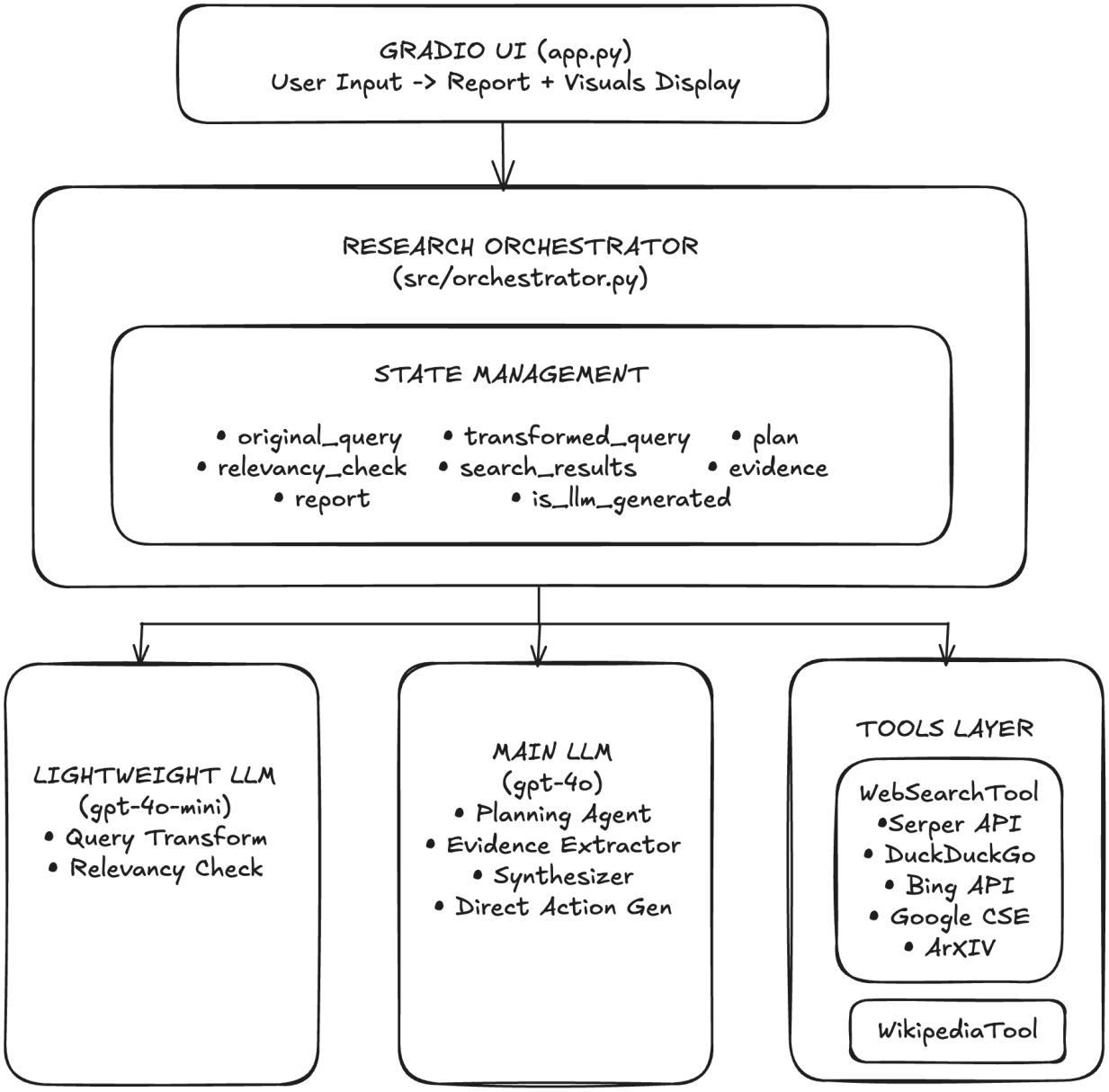
Assumption	Rationale
OpenAI API availability	GPT-4o/GPT-4o-mini provides sufficient reasoning capability for planning and synthesis
Web search reliability	DuckDuckGo (free) serves as the primary fallback; Serper/Bing/Google as premium options
English-only queries	Prompts and parsing logic optimized for English text
Research-focused queries	System designed for factual research, not creative writing or code generation
Internet connectivity	Real-time web search is core to the pipeline
Report length < 10K tokens	Bounded by LLM context windows and cost constraints

Scope Boundaries

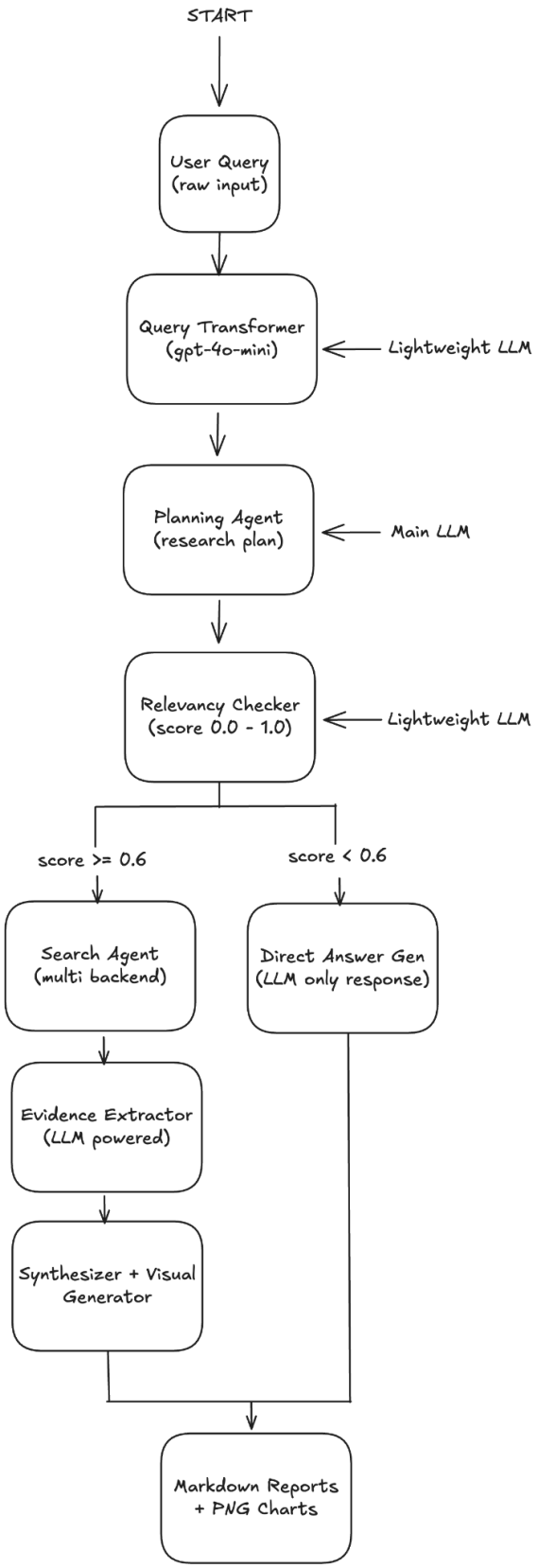
- In scope: Factual research, comparative analysis, trend exploration
- Out of scope: Real-time data (stock prices), personal advice, code generation

2. Architecture

High-Level System Architecture



Data Flow Diagram



Short Overview

1. **Query Transformation**

```
Input:  "AI in healthcare"
Output: "What are the applications, benefits, and challenges of
         artificial intelligence in healthcare delivery and diagnosis?"
```

2. **Research Planning**

```
{
  "research_angles": [
    "Clinical applications of AI",
    "Benefits for patient outcomes",
    "Challenges and limitations"
  ],
  "search_queries": [
    "AI healthcare clinical applications 2024",
    "machine learning medical diagnosis benefits"
  ],
  "focus_areas": ["accuracy", "bias", "adoption"]
}
```

3. **Relevancy Check**

- Validates the plan actually addresses the query
- Passes the threshold: 0.6
- If below the threshold it falls back to LLM generator answer

4. **Evidence Extraction**

- Searches multiple sources
- Extracts key claims, data, quotes
- Scores relevance (0.0 - 1.0)

5. **Report Synthesis**

- Markdown format with headers
- Executive summary
- Key findings by angle
- Visual Charts (by PNG)
- Reference Section

3. Agent Loop Design

State Model

```
state = {
    'original_query': str,          # Raw user input
    'transformed_query': Dict,      # {original, transformed,
research_focus, was_transformed}
    'plan': Dict,                   # {research_angles, search_queries,
focus_areas}
    'relevancy_check': Dict,        # {is_relevant, relevancy_score,
reasoning}
    'is_llm_generated': bool,       # True if fallback to direct LLM answer
    'search_results': List[Dict],   # Raw search results
    'evidence': List[Dict],         # Extracted evidence with relevance
scores
    'report': str                   # Final markdown report
}
```

Agent Responsibilities

Agent	Input	Output	LLM Model
QueryTransformer	Raw query	Optimized query + research focus	gpt-4o-mini
PlanningAgent	Transformed query	Research plan (angles, queries, focus areas)	gpt-4o
RelevancyChecker	Original + transformed query + plan	Relevancy score + decision	gpt-4o-mini
SearchAgent	Search queries	Raw search results	None (tools only)
EvidenceExtractor	Search results + context	Structured evidence with scores	gpt-4o
SynthesisAgent	Evidence + plan	Markdown report	gpt-4o
DirectAnswerGenerator	Query + reasoning	LLM-only report with disclaimer	gpt-4o
VisualGenerator	Plan + evidence	PNG charts	None (matplotlib)

Key Prompts

Query Transformation:

```
Transform the user's query into a clear, research-oriented question.
- Clarify vague terms
- Add research framing (analyze, evaluate, compare)
- Ensure specificity for academic/industry research
```

Planning:

```
Create a comprehensive research plan:
1. Research Angles (3-5 perspectives)
2. Search Queries (5-8 specific queries)
3. Focus Areas (key aspects to cover)
```

Relevancy Check:

```
Analyze whether the research plan aligns with the user intent.
Score 0.0-1.0. If < 0.6, mark "Not Relevant".
Be strict - generic or off-topic plans should fail.
```

Tool Integration

Tool	Provider	Fallback Chain
WebSearchTool	Serper → DuckDuckGo → Bing → Google CSE	Yes
WikipediaTool	Wikipedia API	Used as search fallback
ArXiv Search	ArXiv API	Academic search fallback

4. Data & Trace Model

Evidence Schema

```python
evidence = {
    'source': str,      # URL
    'title': str,       # Page/article title
    'evidence': str,    # LLM-extracted key points
    'relevance': float, # 0.0-1.0 score
    'content': str      # Original snippet
}
```

Search Result Schema

```
search_result = {
    'title': str,
    'snippet': str,
    'url': str,
    'source': str,      # Backend used (duckduckgo, serper, etc.)
    'timestamp': str
}
```

Execution Trace (Terminal Output)

```
============================================================
Step 0: Transforming query...
============================================================
Query transformed: [optimized query preview]...

============================================================
Step 1: Creating research plan...
============================================================
Plan created with 4 angles

============================================================
Step 2: Checking plan relevancy...
============================================================
Relevancy Score: 0.85/1.0
Reasoning: [explanation]...
Plan is relevant (score: 0.85)
Proceeding with research pipeline...

============================================================
Step 3: Executing searches...
============================================================
✓ Found 5 results for: [query 1]...
✓ Search successful with duckduckgo
...

============================================================
Step 4: Extracting evidence...
============================================================
[evidence extraction progress]

============================================================
```

```
Step 5: Synthesizing report...
============================================================
[synthesis completion]
```

Report Artifacts

Artifact	Location	Format
Research Report	`outputs/research_report_YYYYMMDD_HHMMSS.md`	Markdown
Relevance Chart	`outputs/evidence_relevance_*.png`	PNG
Evidence Distribution	`outputs/evidence_distribution_*.png`	PNG
Source Distribution	`outputs/source_distribution_*.png`	PNG

5. Reliability & Safety Considerations

Fallback Strategies

Failure Point	Fallback Strategy
Query transformation fails	Return cleaned original query
Planning fails	Use default plan template
Search backend fails	Chain to next backend (Serper → DDG → Bing → Google)
All searches fail	Return empty results, continue synthesis
Evidence extraction fails	Return raw snippet as evidence
Synthesis fails	Generate basic report template
Relevancy check fails	Assume relevant (score 0.7), proceed with caution

Rate Limiting

```python
def _enforce_rate_limit(self):
    """20 requests/minute limit to avoid API bans"""
    if len(self.request_times) >= self.rate_limit_per_minute:
        sleep_time = 60 - (now - self.request_times[0])
        time.sleep(sleep_time)
```

Safety Mechanisms

Mechanism	Purpose
LLM-generated disclaimer	Clear warning when report isn't based on external sources
Relevancy gating	Prevents generating misleading research on irrelevant plans
Source attribution	All evidence linked to original URLs
Score transparency	Relevancy scores displayed to user
Token limits	`max_tokens=2000` (main), `max_tokens=500` (lightweight)

Known Limitations

- No fact-checking beyond source diversity
- No detection of deliberately misleading sources
- No content moderation for generated reports
- Rate limits may slow down large research tasks

6. Observability & Metrics

Current Observability (Terminal Output)

```
Step-by-step progress logging:
- Query transformation status
- Plan creation summary (number of angles)
- Relevancy score and reasoning
- Search success/failure per query
- Evidence extraction progress
- Synthesis completion
```

Recommended Metrics to Track

Metric	Type	Purpose
`query_transformation_time`	Latency	Performance of lightweight LLM
`planning_time`	Latency	Main LLM planning performance
`relevancy_score`	Quality	Distribution of query-plan alignment
`search_success_rate`	Reliability	Per-backend search reliability
`evidence_count`	Volume	Sources gathered per research

`avg_relevance_score`	Quality	Evidence quality distribution
`llm_fallback_rate`	Reliability	How often direct LLM answers are used
`total_tokens_used`	Cost	API cost tracking
`end_to_end_latency`	Performance	Total research time

Proposed Logging Enhancement

```
# Structured logging (future implementation)
{
    "timestamp": "2026-01-18T12:00:00Z",
    "session_id": "uuid",
    "stage": "relevancy_check",
    "input": {"original_query": "...", "plan_summary": "..."},
    "output": {"score": 0.85, "is_relevant": true},
    "latency_ms": 450,
    "model": "gpt-4o-mini",
    "tokens_used": 320
}
```

7. Tradeoffs

Quality vs. Cost

Decision	Quality Impact	Cost Impact
Two-tier LLM (gpt-4o + gpt-4o-mini)	Slightly lower for simple tasks	~60% reduction for transformation/relevancy
Token limits (2000 main, 500 light)	May truncate complex responses	Predictable cost ceiling
Top 10 evidence in synthesis	May miss edge cases	Bounded synthesis cost
DuckDuckGo as primary search	Less comprehensive than Google	Free (no API cost)

Simplicity vs. Scale

Current Design (Simple)	Scaled Design (Complex)
Single orchestrator	Distributed task queue (Celery/RQ)
Sequential execution	Parallel search execution
In-memory state	Redis/database state persistence
Single-user Gradio	Multi-tenant API with auth
Synchronous processing	Async with WebSocket updates

Latency vs. Thoroughness

Approach	Latency	Thoroughness
5-8 search queries	~30-60s total	Good coverage
Comprehensive search (all backends)	~2-3 min	Maximum coverage
Skip relevancy check	-3-5s	Risk of irrelevant reports

Current Cost Estimate (Per Research)

Component	Estimated Tokens	Cost (GPT-4o)	Cost (GPT-4o-mini)
Query Transform	~300	-	$0.0001
Planning	~800	$0.02	-
Relevancy Check	~400	-	$0.0001
Evidence Extraction (×10)	~5000	$0.125	-
Synthesis	~2000	$0.05	-
Total	~8500	~$0.20	-

8. Future Extensions (Unlimited Time Design)

Key Future Features

1. Multi-Turn Conversational Research

```
User: "What is synthetic data?"
Agent: [Initial report]
User: "Focus more on healthcare applications"
Agent: [Refined report with healthcare emphasis]
```

```
User: "Compare with federated learning for privacy"
Agent: [Comparative analysis]
```

2. Knowledge Graph Construction

- Build persistent knowledge graphs from research
- Enable cross-query learning
- Support "What have we researched about X?" queries

3. Adaptive Depth Control

```
research_config = {
    'depth': 'comprehensive',  # quick | standard | comprehensive | exhaustive
    'source_priority': ['academic', 'news', 'web'],
    'time_budget_minutes': 10,
    'cost_budget_usd': 0.50
}
```

4. Real-Time Monitoring Dashboard

- Live progress visualization
- Cost tracking per research
- Quality metrics over time
- A/B testing different prompts

5. Citation & Fact-Checking Engine

- Cross-reference claims across sources
- Flag contradictions explicitly
- Provide confidence scores per claim
- Link directly to supporting evidence

6. Multi-Modal Research

- PDF/academic paper parsing
- Image analysis (charts, diagrams)
- Video transcript analysis
- Podcast/audio transcription

7. Collaborative Research

- Shared research workspaces
- Version-controlled reports
- Comments and annotations
- Team knowledge base

8. Domain-Specific Agents

```
research_domains = {
    'medical': MedicalResearchAgent(),      # PubMed, clinical trials
    'legal': LegalResearchAgent(),          # Case law, statutes
    'financial': FinancialResearchAgent(),  # SEC filings, earnings
    'technical': TechnicalResearchAgent(),  # GitHub, docs, RFCs
}
```

Technology Stack (Future)

Component	Current	Future
LLM	OpenAI GPT-4o	Claude + GPT-4 + local models (routing)
Vector DB	None	Pinecone / Weaviate
Task Queue	None	Celery + Redis
Knowledge Graph	None	Neo4j
Observability	Print logs	OpenTelemetry + Datadog
UI	Gradio	React + FastAPI
Auth	None	Auth0 / Clerk
Storage	Local files	S3 + PostgreSQL
Search	DuckDuckGo	Elasticsearch (cached results)

Vision: Fully Autonomous Research System