**Design Document for Research Agent**

**Project Overview**

The Research Agent is an advanced tool designed to assist users in conducting comprehensive research on any topic. The agent gathers information from the web, organizes it, and outputs it in user-friendly formats such as Word documents, PDFs, or PowerPoint presentations. The tool will utilize DeepSeek R1 for enhanced web scraping and OpenAI GPT-4 for summarization and content organization.

**Features and Requirements**

**1. User Input Module**

* **Features:**
  + Input field for the research topic.
  + Dropdown or toggle to select output format (Word, PDF, PPT).
  + Options for customization (e.g., include images, citations).
  + Real-time validation for empty input fields.
* **Tech Stack:**
  + Frontend: React.js, Tailwind CSS for UI design.
  + Backend: Express.js or FastAPI to handle input.

**2. Web Scraping and Data Extraction**

* **Features:**
  + Extract search results using APIs like Google Custom Search or Bing Search.
  + DeepSeek R1 integration for precision scraping.
  + Filtering irrelevant data using NLP techniques.
* **Libraries/Tools:**
  + BeautifulSoup or Scrapy (Python) for web scraping.
  + DeepSeek R1 for advanced scraping.

**3. Data Summarization and Organization**

* **Features:**
  + Summarize large volumes of text using AI models.
  + Organize content into predefined sections:
    - **Introduction**: High-level overview of the topic.
    - **Main Content**: Detailed insights and data.
    - **Conclusion**: Summarized findings and key takeaways.
  + Generate citations for extracted data.
* **Libraries/Tools:**
  + OpenAI GPT-4 API or HuggingFace Transformers for summarization.
  + SpaCy or NLTK for cleaning and processing text.

**4. Output Generation Module**

* **Features:**
  + Generate output in:
    - **Word Document** (using python-docx).
    - **PDF** (using fpdf).
    - **PowerPoint Presentation** (using python-pptx).
  + Custom templates for PPT slides.
  + Support for adding images and graphs.
* **Tech Stack:**
  + Python for file generation.
  + Template files for customization.

**5. Backend API**

* **Features:**
  + Receive user input and process requests.
  + Fetch, summarize, and organize data.
  + Handle errors (e.g., no results found, API rate limits).
* **Endpoints:**
  + POST /api/research: Accepts research topic and format.
  + GET /api/status: Provides the status of ongoing research requests.
* **Tech Stack:**
  + Python (FastAPI) or Node.js (Express).

**6. Authentication and Rate Limiting**

* **Features:**
  + Ensure API keys for external services are secure.
  + Implement rate limiting to prevent abuse.
* **Tech Stack:**
  + Middleware for rate limiting (e.g., express-rate-limit or FastAPI’s dependencies).

**7. Deployment and Scalability**

* **Features:**
  + Host the backend on AWS Lambda, Google Cloud Run, or Heroku.
  + Frontend deployment using Vercel or Netlify.
  + Dockerize the application for portability.
* **Tools:**
  + Docker, Kubernetes (if scaling).

**8. Real-time Progress Tracking**

* **Features:**
  + Show progress bar or status updates for:
    - Web scraping.
    - Data summarization.
    - File generation.
  + Notify users when the output is ready for download.
* **Libraries/Tools:**
  + WebSockets (e.g., Socket.IO) for real-time updates.

**9. Plagiarism Check (Future Feature)**

* **Features:**
  + Validate that content is original and cite sources appropriately.
* **Libraries/Tools:**
  + Integration with third-party APIs like Copyscape.

**10. Admin Dashboard (Future Feature)**

* **Features:**
  + Monitor API usage and performance.
  + Manage user feedback and bug reports.
  + Generate reports for system health.
* **Tech Stack:**
  + Admin panel framework (e.g., Strapi, React Admin).

**Architecture Overview**

**1. Frontend**

* **Framework**: React.js.
* **Styling**: Tailwind CSS for responsive design.
* **Key Pages:**
  + Home: Input form for research.
  + Status: Displays progress updates.
  + Download: Allows users to download results.

**2. Backend**

* **Framework**: FastAPI or Express.js.
* **Modules:**
  + Input validation and processing.
  + Integration with APIs (e.g., Google Custom Search).
  + Summarization and file generation.

**3. Database**

* **Purpose:**
  + Store user queries, progress states, and output files.
* **Choice:** MongoDB (NoSQL) or PostgreSQL (relational).

**4. External Integrations**

* **APIs:**
  + Google Custom Search API.
  + DeepSeek R1.
  + OpenAI GPT-4 API.

**5. File Storage**

* **Purpose:**
  + Temporarily store generated documents and presentations.
* **Choice:**
  + AWS S3 or Google Cloud Storage.

**Development Milestones**

**Phase 1: Core Functionality**

* Develop frontend with input and format selection.
* Implement backend to process input and fetch web data.
* Add basic summarization and file generation.

**Phase 2: Advanced Features**

* Integrate DeepSeek R1.
* Add real-time progress tracking.
* Improve file generation with custom templates.

**Phase 3: Deployment and Scaling**

* Deploy the application.
* Optimize for scalability.
* Add monitoring tools.

**Phase 4: Future Enhancements**

* Add plagiarism checking.
* Implement an admin dashboard.

**Tools and Libraries**

**Frontend**

* React.js
* Tailwind CSS

**Backend**

* FastAPI (Python) or Express.js (Node.js)

**Data Processing**

* BeautifulSoup, Scrapy
* OpenAI GPT-4 API, SpaCy

**File Generation**

* python-docx
* fpdf
* python-pptx

**Deployment**

* AWS Lambda, Google Cloud Run
* Vercel, Netlify

**Challenges and Solutions**

**Challenge 1: Data Relevance**

* **Solution:** Use advanced scraping techniques and AI models for filtering.

**Challenge 2: API Rate Limits**

* **Solution:** Implement caching and rate-limiting mechanisms.

**Challenge 3: Scalability**

* **Solution:** Use containerization and cloud-based deployment.

**Conclusion**

This design document outlines all the necessary components, features, and tools required to build the Research Agent. The project will deliver a user-friendly tool that automates the research process and provides high-quality, customizable outputs.