1st and 2nd prize for Inventohub contest 1

\*\*Evaluation Criteria\*\*

\*\*1. Creativity (Weight: 30%)\*\*

- \*\*Definition\*\*: How novel and original are the ideas generated by the model?

- \*\*Metrics\*\*:

- Number of unique ideas generated.

- Diversity of ideas across different fields (e.g., engineering, biomedical, green tech).

- Use of analogical thinking (e.g., solving a problem in one field using insights from another).

- \*\*Example\*\*: A model that suggests using \*\*biomimicry\*\* (e.g., designing a drone inspired by bird flight) scores higher than one that suggests conventional solutions.

---

\*\*2. Technical Feasibility (Weight: 30%)\*\*

- \*\*Definition\*\*: How practical and technically sound are the solutions proposed by the model?

- \*\*Metrics\*\*:

- Alignment with engineering principles and scientific laws.

- Use of realistic materials, technologies, and processes.

- Avoidance of "hallucinations" (e.g., suggesting impossible designs or materials).

- \*\*Example\*\*: A model that proposes a \*\*solar-powered water purification system\*\* using existing materials scores higher than one that suggests a system requiring non-existent technology.

\*\*3. Practical Impact (Weight: 20%)\*\*

- \*\*Definition\*\*: How impactful and useful are the solutions in real-world scenarios?

- \*\*Metrics\*\*:

- Potential to solve pressing global challenges (e.g., climate change, healthcare, energy).

- Scalability and cost-effectiveness of the proposed solutions.

- Alignment with user needs (e.g., individual inventors, startups).

- \*\*Example\*\*: A model that suggests a \*\*low-cost, portable diagnostic device\*\* for rural areas scores higher than one that suggests a high-end, expensive solution.

---

\*\*4. Patentability (Weight: 10%)\*\*

- \*\*Definition\*\*: How likely are the ideas to be patentable (i.e., novel, non-obvious, and useful)?

- \*\*Metrics\*\*:

- Number of ideas that pass a preliminary patent search (e.g., no prior art found).

- Uniqueness of the proposed solutions compared to existing patents.

- \*\*Example\*\*: A model that generates ideas with no prior art in the \*\*USPTO database\*\* scores higher than one that suggests already patented solutions.

---

\*\*5. User Experience (Weight: 10%)\*\*

- \*\*Definition\*\*: How intuitive and user-friendly is the model's interface?

- \*\*Metrics\*\*:

- Ease of use (e.g., clear prompts, simple interface).

- Speed and responsiveness of the model.

- Quality of explanations provided for generated ideas.

- \*\*Example\*\*: A model with a \*\*Streamlit app\*\* that allows users to input problems and receive detailed, step-by-step solutions scores higher than one with a complex, hard-to-use interface.

---

\*\*Scoring System\*\*

- Each criterion is scored on a scale of \*\*1 to 10\*\*.

- The scores are weighted according to the percentages above.

- The total score determines the winner.

| \*\*Criterion\*\* | \*\*Weight\*\* |

| Creativity | 30%

| Technical Feasibility | 30%

| Practical Impact | 20%

| Patentability | 10%

| User Experience | 10%

| \*\*Total\*\* | 100%

A \*\*step-by-step guide\*\* and \*\*Python code\*\* to collect, clean, and preprocess data from the sources you mentioned (patents, research papers, general knowledge, and historical inventions). This guide assumes you have basic familiarity with Python and libraries like `requests`, `pandas`, and `BeautifulSoup`.

---

\*\*Step-by-Step Guide\*\*

\*\*Step 1: Set Up Your Environment\*\*

1. Install required Python libraries:

```bash

pip install requests pandas beautifulsoup4 lxml

```

2. Create a project folder and organize it:

```

inventor\_llm\_data/

├── data/

│ ├── patents/

│ ├── papers/

│ ├── general\_knowledge/

│ └── inventions/

├── scripts/

│ ├── collect\_patents.py

│ ├── collect\_papers.py

│ ├── collect\_general\_knowledge.py

│ └── collect\_inventions.py

└── requirements.txt

```

---

\*\*Step 2: Collect Patent Data\*\*

\*\*Sources\*\*:

- USPTO Open Data API

- Google Patents

- CNIPA (Chinese patents)

\*\*Steps\*\*:

1. Use the \*\*USPTO Open Data API\*\* to fetch patent data.

2. Use \*\*Google Patents\*\* for global patents (scrape or use their dataset).

3. For \*\*CNIPA\*\*, download datasets and translate if necessary.

\*\*Python Code\*\*:

```python

import requests

import pandas as pd

# USPTO Open Data API

def fetch\_uspto\_patents(query, max\_results=100):

base\_url = "https://developer.uspto.gov/ibd-api/v1/patent/application"

params = {

"searchText": query,

"rows": max\_results

}

response = requests.get(base\_url, params=params)

if response.status\_code == 200:

return response.json()

else:

print(f"Error fetching USPTO data: {response.status\_code}")

return []

# Example: Fetch patents related to "water purification"

uspto\_patents = fetch\_uspto\_patents("water purification")

pd.DataFrame(uspto\_patents).to\_csv("data/patents/uspto\_patents.csv", index=False)

# Google Patents (Scraping example)

from bs4 import BeautifulSoup

def fetch\_google\_patents(query, max\_results=10):

url = f"https://patents.google.com/?q={query}"

response = requests.get(url)

if response.status\_code == 200:

soup = BeautifulSoup(response.text, "html.parser")

patents = []

for result in soup.select("div.result"):

title = result.select\_one("h3").text.strip()

link = "https://patents.google.com" + result.select\_one("a")["href"]

patents.append({"title": title, "link": link})

return patents[:max\_results]

else:

print(f"Error fetching Google Patents: {response.status\_code}")

return []

# Example: Fetch patents related to "solar energy"

google\_patents = fetch\_google\_patents("solar energy")

pd.DataFrame(google\_patents).to\_csv("data/patents/google\_patents.csv", index=False)

```

\*\*Step 3: Collect Research Papers\*\*

\*\*Sources\*\*:

- arXiv (physics, math, computer science)

- PubMed (biomedical research)

\*\*Steps\*\*:

1. Use arXiv's API to fetch papers.

2. Use PubMed's API to fetch biomedical papers.

\*\*Python Code\*\*:

```python

# arXiv API

def fetch\_arxiv\_papers(query, max\_results=10):

base\_url = "http://export.arxiv.org/api/query"

params = {

"search\_query": query,

"max\_results": max\_results

}

response = requests.get(base\_url, params=params)

if response.status\_code == 200:

return response.text # Returns XML data

else:

print(f"Error fetching arXiv data: {response.status\_code}")

return ""

# Example: Fetch papers related to "machine learning"

arxiv\_papers = fetch\_arxiv\_papers("machine learning")

with open("data/papers/arxiv\_papers.xml", "w") as f:

f.write(arxiv\_papers)

# PubMed API

def fetch\_pubmed\_papers(query, max\_results=10):

base\_url = "https://eutils.ncbi.nlm.nih.gov/entrez/eutils/esearch.fcgi"

params = {

"db": "pubmed",

"term": query,

"retmax": max\_results

}

response = requests.get(base\_url, params=params)

if response.status\_code == 200:

return response.text # Returns XML data

else:

print(f"Error fetching PubMed data: {response.status\_code}")

return ""

# Example: Fetch papers related to "cancer treatment"

pubmed\_papers = fetch\_pubmed\_papers("cancer treatment")

with open("data/papers/pubmed\_papers.xml", "w") as f:

f.write(pubmed\_papers)

```

\*\*Step 4: Collect General Knowledge\*\*

\*\*Sources\*\*:

- Wikipedia

- Common Crawl

\*\*Steps\*\*:

1. Use Wikipedia's API to fetch articles.

2. Use Common Crawl datasets (preprocessed).

\*\*Python Code\*\*:

```python

# Wikipedia API

def fetch\_wikipedia\_articles(query, max\_results=10):

base\_url = "https://en.wikipedia.org/w/api.php"

params = {

"action": "query",

"list": "search",

"srsearch": query,

"format": "json",

"srlimit": max\_results

}

response = requests.get(base\_url, params=params)

if response.status\_code == 200:

return response.json()["query"]["search"]

else:

print(f"Error fetching Wikipedia data: {response.status\_code}")

return []

# Example: Fetch articles related to "renewable energy"

wikipedia\_articles = fetch\_wikipedia\_articles("renewable energy")

pd.DataFrame(wikipedia\_articles).to\_csv("data/general\_knowledge/wikipedia\_articles.csv", index=False)

```

\*\*Step 5: Collect Historical Inventions\*\*

\*\*Sources\*\*:

- Hackaday

- Instructables

\*\*Steps\*\*:

1. Scrape Hackaday and Instructables for project descriptions.

\*\*Python Code\*\*:

```python

# Scrape Hackaday

def scrape\_hackaday(query, max\_results=10):

url = f"https://hackaday.com/?s={query}"

response = requests.get(url)

if response.status\_code == 200:

soup = BeautifulSoup(response.text, "html.parser")

projects = []

for result in soup.select("article.post"):

title = result.select\_one("h2.entry-title").text.strip()

link = result.select\_one("a")["href"]

projects.append({"title": title, "link": link})

return projects[:max\_results]

else:

print(f"Error scraping Hackaday: {response.status\_code}")

return []

# Example: Scrape projects related to "robotics"

hackaday\_projects = scrape\_hackaday("robotics")

pd.DataFrame(hackaday\_projects).to\_csv("data/inventions/hackaday\_projects.csv", index=False)

```

---

\*\*Step 6: Clean and Preprocess Data\*\*

1. Combine all datasets into a single DataFrame.

2. Remove duplicates and irrelevant content.

3. Tokenize and structure data into Q/A pairs.

\*\*Python Code\*\*:

```python

import pandas as pd

from sklearn.model\_selection import train\_test\_split

# Load all datasets

patents = pd.read\_csv("data/patents/uspto\_patents.csv")

papers = pd.read\_csv("data/papers/arxiv\_papers.csv")

general\_knowledge = pd.read\_csv("data/general\_knowledge/wikipedia\_articles.csv")

inventions = pd.read\_csv("data/inventions/hackaday\_projects.csv")

# Combine datasets

data = pd.concat([patents, papers, general\_knowledge, inventions], ignore\_index=True)

# Remove duplicates

data.drop\_duplicates(subset=["title"], inplace=True)

# Save cleaned data

data.to\_csv("data/cleaned\_data.csv", index=False)

```

---

\*\*Step 7: Finalize Dataset\*\*

- Spend \*\*1-2 weeks\*\* collecting and cleaning data.

- Aim for \*\*10,000-20,000 high-quality documents\*\*.

===============================================================

\*\*Timeline Breakdown with Roadblock Management\*\*

\*\*Day 1: Understanding Requirements & Setup\*\*

- \*\*Task\*\*: Review the provided plan, code, and datasets.

- \*\*Outcome\*\*: Set up the environment (install libraries, organize folders, etc.).

- \*\*Time\*\*: \*\*1 day\*\*.

\*\*Roadblocks & Tips\*\*:

- \*\*Issue\*\*: Missing dependencies or version conflicts.

- \*\*Tip\*\*: Use a virtual environment (e.g., `venv` or `conda`) and a `requirements.txt` file to ensure consistent library versions.

- \*\*Issue\*\*: Lack of clarity on requirements.

- \*\*Tip\*\*: Schedule a quick call with stakeholders to clarify goals and expectations.

---

\*\*Days 2-3: Data Collection\*\*

- \*\*Task\*\*: Run the provided Python scripts to collect data from:

- USPTO, Google Patents, and CNIPA (patents).

- arXiv and PubMed (research papers).

- Wikipedia and Common Crawl (general knowledge).

- Hackaday and Instructables (historical inventions).

- \*\*Outcome\*\*: Raw datasets stored in the `data/` folder.

- \*\*Time\*\*: \*\*2 days\*\*.

\*\*Roadblocks & Tips\*\*:

- \*\*Issue\*\*: API rate limits or downtime.

- \*\*Tip\*\*: Implement retry logic with exponential backoff in the code. Use caching to avoid repeated API calls.

- \*\*Issue\*\*: Scraping blocked by websites.

- \*\*Tip\*\*: Use headers to mimic a real browser (e.g., `User-Agent`). Rotate IP addresses if necessary.

- \*\*Issue\*\*: CNIPA data in Chinese.

- \*\*Tip\*\*: Use Google Translate API or a library like `googletrans` for translation.

---

\*\*Days 4-5: Data Cleaning & Preprocessing\*\*

- \*\*Task\*\*:

- Combine datasets into a single DataFrame.

- Remove duplicates and irrelevant content.

- Tokenize and structure data into Q/A pairs.

- \*\*Outcome\*\*: Cleaned dataset (`cleaned\_data.csv`).

- \*\*Time\*\*: \*\*2 days\*\*.

\*\*Roadblocks & Tips\*\*:

- \*\*Issue\*\*: Inconsistent data formats.

- \*\*Tip\*\*: Write custom parsers for each data source to standardize formats.

- \*\*Issue\*\*: Missing or incomplete data.

- \*\*Tip\*\*: Use imputation techniques (e.g., fill missing values with placeholders) or exclude incomplete entries.

- \*\*Issue\*\*: Large dataset size.

- \*\*Tip\*\*: Use chunking (e.g., `pandas.read\_csv(chunksize=...)`) to process data in smaller batches.

---

\*\*Days 6-7: Fine-Tuning the Model\*\*

- \*\*Task\*\*:

- Load a pre-trained model (e.g., Mistral 7B or Llama 2 7B).

- Fine-tune the model using the cleaned dataset.

- Use LoRA for efficient fine-tuning.

- \*\*Outcome\*\*: Fine-tuned model checkpoint.

- \*\*Time\*\*: \*\*2 days\*\*.

\*\*Roadblocks & Tips\*\*:

- \*\*Issue\*\*: Hardware limitations (e.g., insufficient GPU memory).

- \*\*Tip\*\*: Use mixed precision training (`fp16`) or gradient checkpointing to reduce memory usage.

- \*\*Issue\*\*: Slow training.

- \*\*Tip\*\*: Use distributed training (e.g., Hugging Face Accelerate) or cloud GPUs (e.g., AWS, GCP).

- \*\*Issue\*\*: Overfitting.

- \*\*Tip\*\*: Use early stopping and monitor validation loss during training.

---

\*\*Day 8: Evaluation\*\*

- \*\*Task\*\*:

- Test the model on a small set of inventor-related queries.

- Evaluate outputs for creativity, accuracy, and relevance.

- \*\*Outcome\*\*: Initial results and feedback.

- \*\*Time\*\*: \*\*1 day\*\*.

\*\*Roadblocks & Tips\*\*:

- \*\*Issue\*\*: Model outputs are irrelevant or nonsensical.

- \*\*Tip\*\*: Adjust the temperature parameter (lower for accuracy, higher for creativity) or use contrastive search.

- \*\*Issue\*\*: Lack of evaluation metrics.

- \*\*Tip\*\*: Use automated metrics (e.g., BLEU, ROUGE) and recruit a small group of inventors for human evaluation.

---

\*\*Day 9: Presentation of First Results\*\*

- \*\*Task\*\*:

- Prepare a demo (e.g., a Streamlit app or Jupyter notebook).

- Showcase the model’s ability to generate ideas, solve problems, and assist with patent research.

- \*\*Outcome\*\*: First results presented to stakeholders.

- \*\*Time\*\*: \*\*1 day\*\*.

\*\*Roadblocks & Tips\*\*:

- \*\*Issue\*\*: Demo not ready on time.

- \*\*Tip\*\*: Prioritize core functionalities (e.g., idea generation) and defer advanced features (e.g., patent search) for later.

- \*\*Issue\*\*: Stakeholder dissatisfaction.

- \*\*Tip\*\*: Set clear expectations upfront and highlight the iterative nature of the project.

---

\*\*Total Time: 9 Days\*\*

- \*\*Data Collection\*\*: 2 days.

- \*\*Data Cleaning & Preprocessing\*\*: 2 days.

- \*\*Model Fine-Tuning\*\*: 2 days.

- \*\*Evaluation\*\*: 1 day.

- \*\*Presentation\*\*: 1 day.

- \*\*Buffer\*\*: 1 day (for unexpected delays).

---

### \*\*Key Roadblocks & Mitigation Strategies\*\*

| \*\*Roadblock\*\* | \*\*Mitigation Strategy\*\* |

|------------------------------------|-----------------------------------------------------------------------------------------|

| API rate limits or downtime | Implement retry logic, use caching, and monitor API health. |

| Scraping blocked by websites| Use headers, rotate IPs, and respect `robots.txt` rules. |

| Inconsistent data formats | Write custom parsers for each data source. |

| Hardware limitations | Use mixed precision training, gradient checkpointing, or cloud GPUs. |

| Overfitting | Use early stopping and monitor validation loss. |

| Irrelevant model outputs | Adjust temperature or use contrastive search. |

| Stakeholder dissatisfaction | Set clear expectations and highlight iterative progress. |

---

\*\*Deliverables After 9 Days\*\*

1. \*\*Fine-Tuned Model\*\*: A model capable of generating ideas, solving technical problems, and assisting with patent research.

2. \*\*Demo\*\*: A simple interface (e.g., Streamlit app) to showcase the model’s capabilities.

3. \*\*Initial Results\*\*: Examples of the model’s outputs for inventor-related tasks.

---

Based on our requirements and focus on individual inventors, here’s a \*\*specific plan\*\* with the \*\*best tools\*\* and a \*\*step-by-step roadmap\*\* to build your LLM for inventors. This plan balances cost-effectiveness, scalability, and practicality.

---

### \*\*Step 1: Data Collection\*\* **this is a our assignment**

\*\*Tools\*\*:

1. \*\*Patent Data\*\*:

- Use the \*\*USPTO Open Data API\*\* (free) and \*\*Google Patents\*\* (free) for US and global patents.

- For Chinese patents, use \*\*CNIPA’s open datasets\*\* (free, but may require translation).(its optional)

2. \*\*Research Papers\*\*:

- Use \*\*arXiv\*\* (free) for physics, math, and computer science.

- Use \*\*PubMed\*\* (free) for biomedical research.

3. \*\*General Knowledge\*\*:

- Use \*\*Wikipedia\*\* (free) and \*\*Common Crawl\*\* (free) for broad knowledge.

4. \*\*Historical Inventions\*\*:

- Scrape platforms like \*\*Hackaday\*\* and \*\*Instructables\*\* (free, but check terms of use).

\*\*Plan\*\*:

- Spend \*\*1-2 weeks\*\* collecting and cleaning data.

- Focus on \*\*10,000-20,000 high-quality documents\*\* (patents, papers, and case studies) for the initial dataset.

---

### \*\*Step 2: Preprocessing\*\*

\*\*Tools\*\*:

1. \*\*Data Cleaning\*\*:

- Use \*\*Python libraries\*\* like Pandas and BeautifulSoup for cleaning and structuring data.

2. \*\*Tokenization\*\*:

- Use \*\*Hugging Face’s Tokenizers\*\* (free) to handle technical jargon.

3. \*\*Structuring Data\*\*:

- Format data into \*\*Q/A pairs\*\* (e.g., "Problem: X → Solution: Y") for fine-tuning.

\*\*Plan\*\*:

- Spend \*\*1 week\*\* preprocessing the data.

- Use \*\*Hugging Face Datasets\*\* to store and manage the cleaned data.

---

### \*\*Step 3: Model Selection & Fine-Tuning\*\*

\*\*Tools\*\*:

1. \*\*Base Model\*\*:

- Use \*\*Mistral 7B\*\* (open-source, lightweight, and efficient).

2. \*\*Fine-Tuning\*\*:

- Use \*\*LoRA (Low-Rank Adaptation)\*\* for cost-effective fine-tuning.

- Train on \*\*Google Colab\*\* (free tier) or \*\*Hugging Face AutoTrain\*\* (low-cost).

3. \*\*Framework\*\*:

- Use \*\*Hugging Face Transformers\*\* (free) for fine-tuning.

\*\*Plan\*\*:

- Spend \*\*2-3 weeks\*\* fine-tuning the model.

- Start with a \*\*small subset of data\*\* (1,000-2,000 examples) to test the pipeline.

- Gradually scale to the full dataset.

---

### \*\*Step 4: Evaluation\*\*

\*\*Tools\*\*:

1. \*\*Automated Metrics\*\*:

- Use \*\*BLEU\*\* and \*\*ROUGE\*\* scores to evaluate text generation quality.

2. \*\*Human Evaluation\*\*:

- Recruit \*\*5-10 inventors\*\* from platforms like \*\*Hackaday\*\* or \*\*Instructables\*\* for feedback.

- Use \*\*Google Forms\*\* or \*\*Typeform\*\* to collect structured feedback.

\*\*Plan\*\*:

- Spend \*\*1 week\*\* evaluating the model.

- Iterate based on feedback to improve outputs.

---

### \*\*Step 5: Deployment\*\*

\*\*Tools\*\*:

1. \*\*Web App\*\*:

- Use \*\*Streamlit\*\* (free and easy to use) for a simple interface.

2. \*\*Chatbot\*\*:

- Use \*\*LangChain\*\* (free) to build an interactive chatbot.

3. \*\*Hosting\*\*:

- Use \*\*Hugging Face Spaces\*\* (free for small apps) or \*\*Google Cloud Run\*\* (low-cost) for deployment.

\*\*Plan\*\*:

- Spend \*\*1-2 weeks\*\* building and deploying the app.

- Include features like:

- \*\*Patent search integration\*\* (via USPTO API).

- \*\*Sketch-to-text tools\*\* for describing prototypes.

- \*\*Simple Q/A interface\*\* for brainstorming and troubleshooting.

---

### \*\*Step 6: Ethical & Legal Compliance\*\*

\*\*Tools\*\*:

1. \*\*Bias Mitigation\*\*:

- Use \*\*Hugging Face’s Datasets\*\* to ensure diverse data.

2. \*\*Patent Compliance\*\*:

- Add a \*\*disclaimer\*\* to the app (e.g., "Verify novelty with a patent attorney").

3. \*\*Privacy\*\*:

- Use \*\*encryption\*\* for user inputs (e.g., HTTPS for web apps).

\*\*Plan\*\*:

- Spend \*\*1 week\*\* implementing ethical safeguards.

- Consult a legal expert for patent compliance.

---

### \*\*Step 7: Iteration & Scaling\*\*

\*\*Tools\*\*:

1. \*\*Feedback Collection\*\*:

- Use \*\*Google Analytics\*\* or \*\*Hotjar\*\* to track user interactions.

2. \*\*Model Updates\*\*:

- Use \*\*Hugging Face AutoTrain\*\* for continuous fine-tuning.

3. \*\*Scaling\*\*:

- Move to \*\*AWS\*\* or \*\*Google Cloud\*\* for larger-scale deployment.

\*\*Plan\*\*:

- Spend \*\*1-2 weeks\*\* collecting feedback and iterating.

- Scale the model to \*\*13B or 70B parameters\*\* as needed.

---

### \*\*Budget & Timeline\*\*

\*\*Budget\*\*:

- \*\*Initial Phase (3 months)\*\*:

- Data collection & preprocessing: \*\*$0\*\* (free tools).

- Fine-tuning: \*\*$100-$500\*\* (Google Colab Pro or Hugging Face AutoTrain).

- Deployment: \*\*$0-$100\*\* (Hugging Face Spaces or Google Cloud Run).

- \*\*Scaling Phase (6+ months)\*\*:

- Cloud hosting: \*\*$200-$500/month\*\* (AWS or GCP).

- Continuous fine-tuning: \*\*$100-$300/month\*\*.

\*\*Timeline\*\*:

- \*\*Month 1\*\*: Data collection & preprocessing.

- \*\*Month 2\*\*: Fine-tuning & evaluation.

- \*\*Month 3\*\*: Deployment & initial feedback.

- \*\*Months 4-6\*\*: Iteration & scaling.

---

### \*\*Recommended Tools Summary\*\*

1. \*\*Data Collection\*\*: USPTO API, Google Patents, arXiv, Wikipedia.

2. \*\*Preprocessing\*\*: Pandas, Hugging Face Tokenizers.

3. \*\*Fine-Tuning\*\*: Mistral 7B, LoRA, Hugging Face Transformers.

4. \*\*Evaluation\*\*: BLEU/ROUGE, Google Forms for human feedback.

5. \*\*Deployment\*\*: Streamlit, LangChain, Hugging Face Spaces.

6. \*\*Ethics\*\*: Hugging Face Datasets, legal consultation.

7. \*\*Scaling\*\*: AWS, Google Cloud, Hugging Face AutoTrain.

---

This plan is designed to be \*\*cost-effective\*\* and \*\*scalable\*\*, starting with free tools and gradually investing in cloud services as your user base grows. Let me know if you’d like further details on any step! 🚀