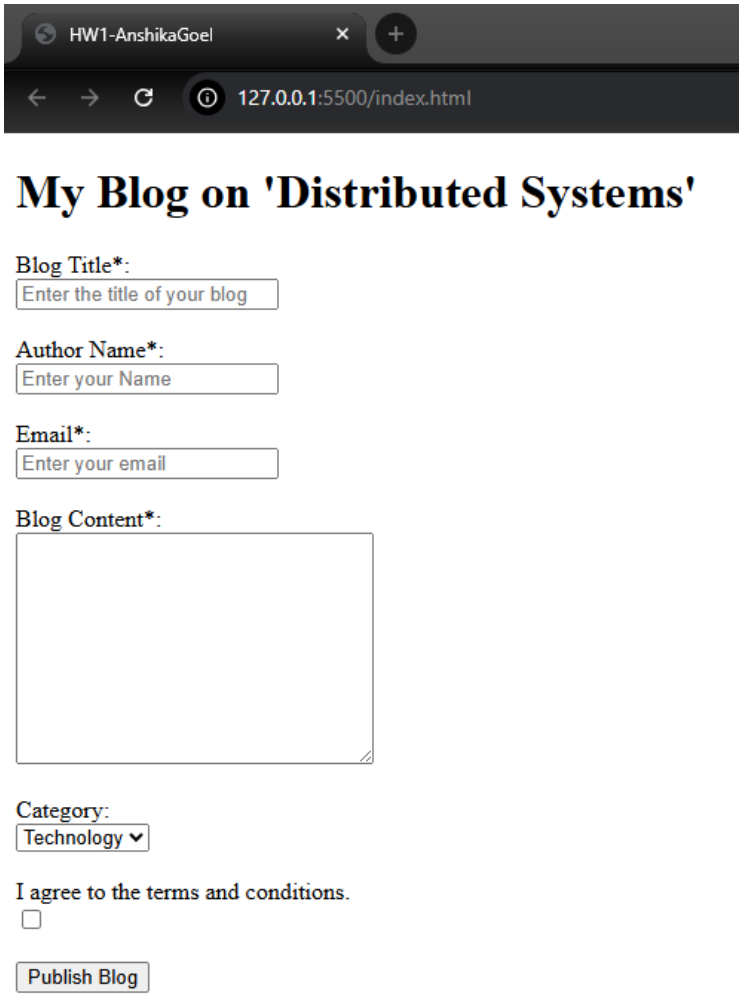
GitHub Repo: https://github.com/GoelAnshika99/DATA236_HW1_AnshikaGoel_018317819

Part 1

HTML

1. Create an HTML page with the title "HW1-{Your Name}". (0.5 points)

```html
<html lange = 'en'>
    <head>
        <meta charset="UTF-8">
        <title>HW1-AnshikaGoel</title>
    </head>
```



2. Add a heading tag to name the title of the blog topic. (Note. you have to use the header tag which renders the biggest font size). (0.5 points)

```html
<body>
    <h1>My Blog on 'Distributed Systems'</h1>
```

My Blog on 'Distributed Systems'

3. Create a form for the blog. The form should include the following:
    a. A text input for the blog title, placeholder text “Enter the title of your blog”. This field should be required and the cursor should automatically focus on this field when the page loads. (0.5 points)

```html
<form id="myForm">
    <label for="blogTitle">Blog Title*:</label><br>
    <input type="text" id="blogTitle" name="blogTitle" placeholder="Enter the title of your blog"
    autofocus required><br><br>
```

Blog Title*:
Enter the title of your blog

    b. A text input for the author name, placeholder "Enter your name”, and required. (0.5 points)

```html
<label for="authorName">Author Name*:</label><br>
<input type="text" id="authorName" name="authorName" placeholder="Enter your Name" required><br><br>
```

Author Name*:
Enter your Name

    c. An email input for the email address, placeholder “Enter your email”, and required. (0.5 points)

```html
<label for="authorEmail">Email*:</label><br>
<input type="email" id="authorEmail" name="authorEmail" placeholder="Enter your email"
required><br><br>
```

Email*:
Enter your email

    d. A text area for the blog content, placeholder “Write your content here…”, and required. (0.5 points)

```html
<label for="blogContent">Blog Content*:</label><br>
<textarea name="blogContent" id="blogContent" placeholer="Write your content here....." rows="10"
cols="30" required></textarea><br><br>
```

Blog Content*:

Write your content here.....

e. A dropdown for category selection, and options "Technology," "Lifestyle," "Travel," and "Education." (0.5 points)

```
<label for="blogCategory">Category:</label><br>
<select name="blogCategory" id="blogCategory">
    <option value="Technology">Technology</option>
    <option value="Travel">Travel</option>
    <option value="Lifestyle">Lifestyle</option>
    <option value="Education">Education</option>
</select><br><br>
```

Category:

Technology

f. A checkbox and a label with the text “I agree to the terms and conditions.” (0.5 points)

```
<label for="termsConditions">I agree to the terms and conditions.</label><br>
<input type="checkbox" id="termsConditions"><br><br>
```

I agree to the terms and conditions.

g. A submit button with the text “Publish Blog”. (0.5 points)

```
<button type="submit">Publish Blog</button>
```

Publish Blog

4. Add a script tag to link your javascript code for part II at the end of your HTML file. (0.5 points)

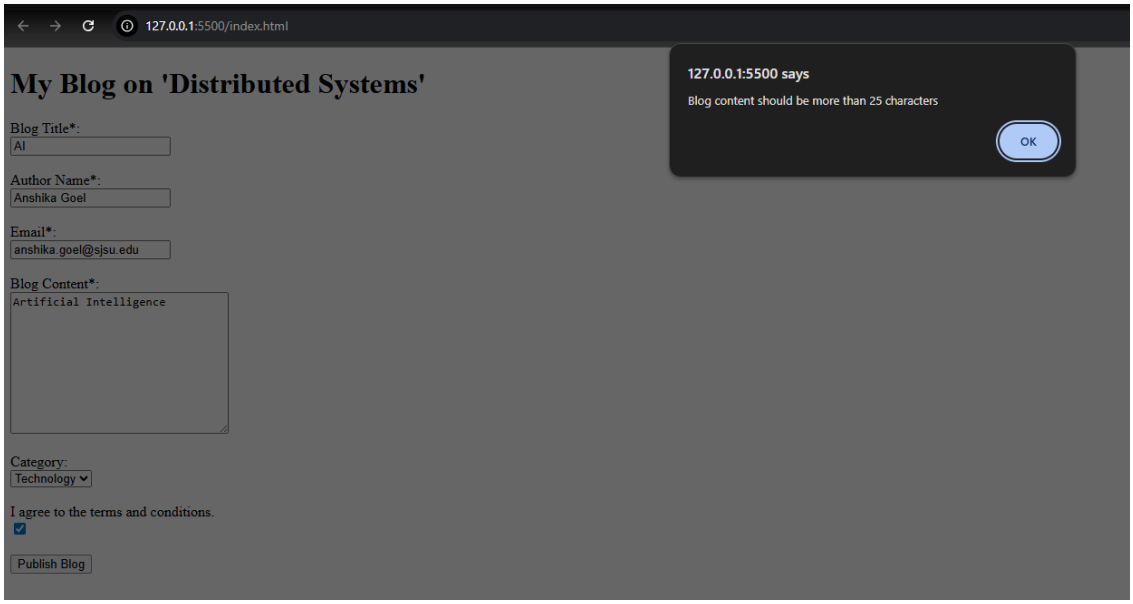
```
<script src="new_script.js"></script>
```

Javascript

1. Write a javascript function using an arrow function to validate:

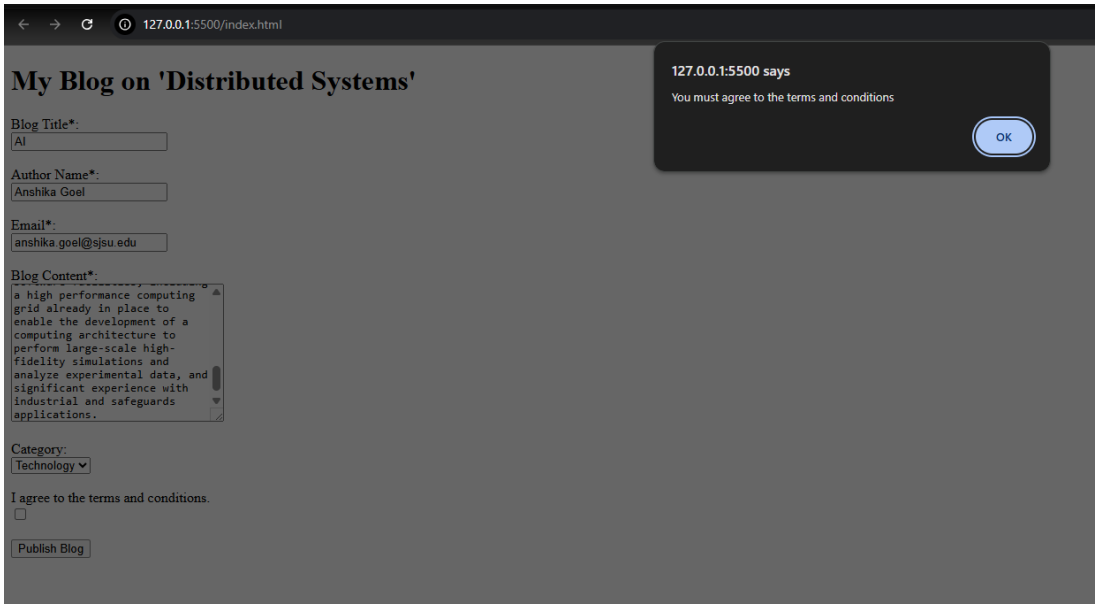
a. Verify if the blog content is more than 25 characters. If the validation fails, display an alert with the message “Blog content should be more than 25 characters”. (1 points)

```
if (blogContent.length <= 25) {
    alert("Blog content should be more than 25 characters");
    return false;
}
```

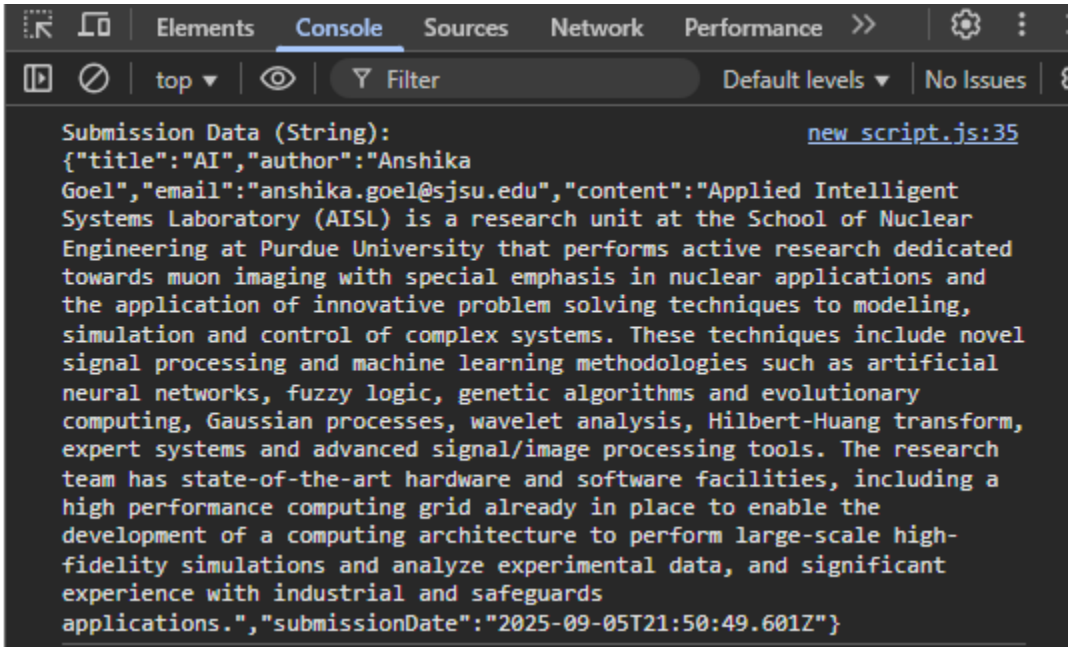


b. Verify if the terms and conditions check box is checked. If the validation fails, display an alert with the message “You must agree to the terms and conditions”. (1 points)

```
if (!termsConditions) {
    alert("You must agree to the terms and conditions");
    return false;
}
```



2. After the form submission is successful, convert the form data into a JSON string and log the output in the console. (2 points)



3. Use object destructuring to extract the title and email fields from the parsed object and log their values in the console. (2 points)

```
Blog Title: AI                                          new script.js:39
Author Name: Anshika Goel                               new script.js:40
Author Email: anshika.goel@sjsu.edu                     new script.js:41
Blog Content: Applied Intelligent Systems Laboratory    new script.js:42
(AISL) is a research unit at the School of Nuclear Engineering at Purdue
University that performs active research dedicated towards muon imaging
with special emphasis in nuclear applications and the application of
innovative problem solving techniques to modeling, simulation and control
of complex systems. These techniques include novel signal processing and
machine learning methodologies such as artificial neural networks, fuzzy
logic, genetic algorithms and evolutionary computing, Gaussian processes,
wavelet analysis, Hilbert-Huang transform, expert systems and advanced
signal/image processing tools. The research team has state-of-the-art
hardware and software facilities, including a high performance computing
grid already in place to enable the development of a computing
architecture to perform large-scale high-fidelity simulations and analyze
experimental data, and significant experience with industrial and
safeguards applications.
```

4. Use the spread operator to add a new field “submissionDate” with the current date and time to the parsed object. Log the updated parsed object in the console. (2 points)

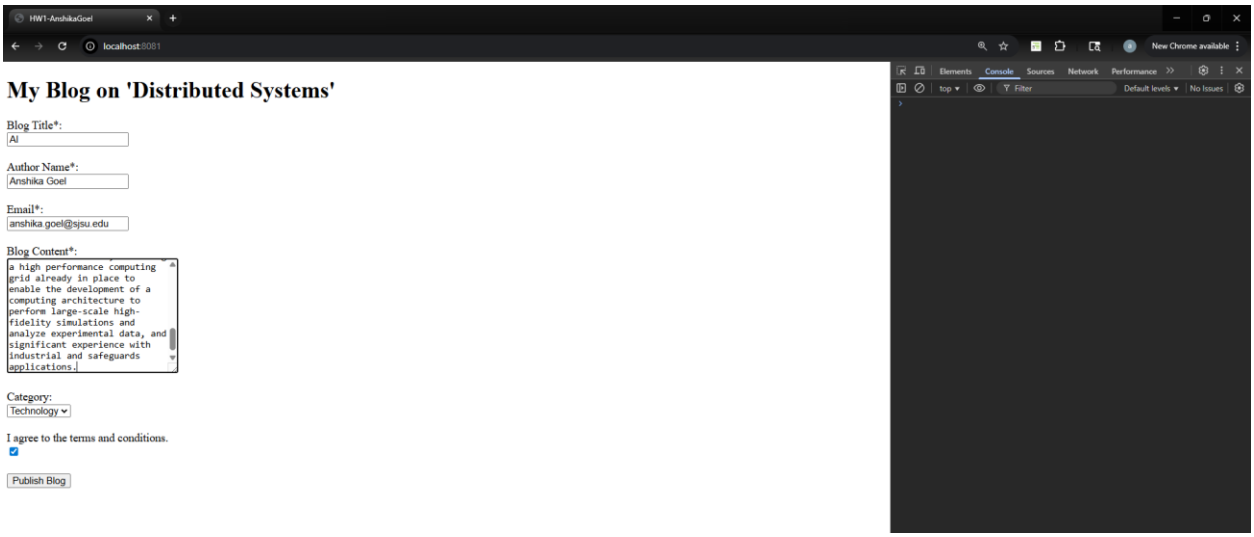
```
Updated Submission:                                     new script.js:46
 ▾ Object ℹ
     author: "Anshika Goel"
     content: "Applied Intelligent Systems Laboratory (AISL) is a research
     email: "anshika.goel@sjsu.edu"
     id: "submission-1"
     submissionDate: "2025-09-05T21:50:49.601Z"
     title: "AI"
   ▸ [[Prototype]]: Object
```

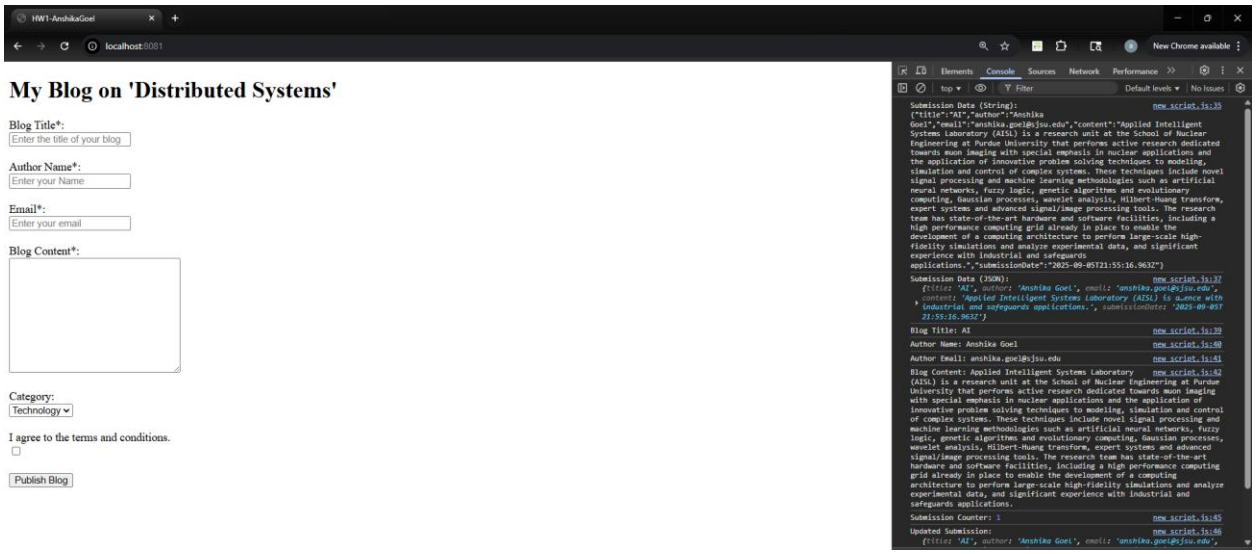
5. Create a closure to track how many times the form has been successfully submitted and log the submission count each time the form is submitted. (2 points)

```
Submission Counter: 1                                   new script.js:45
```

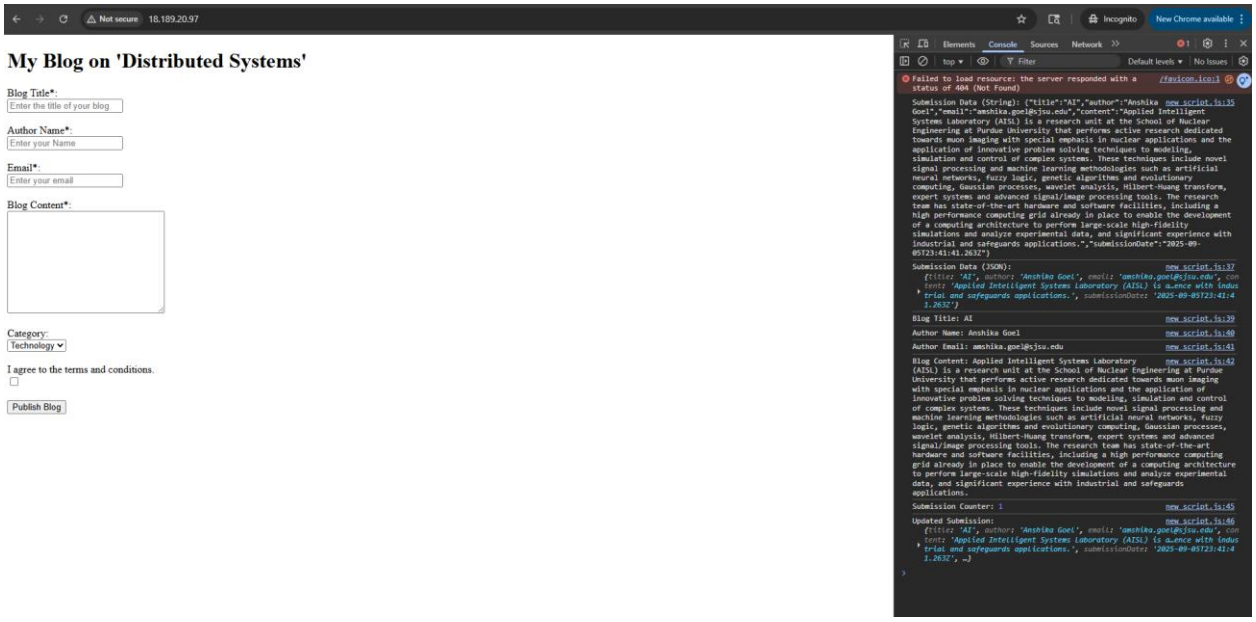
Deployment

Docker





AWS ECS



Part 2

1. Set up a small local LLM with Ollama.(use smollm:1.7b)

```
C:\Users\aradh>ollama --version
ollama version is 0.11.10
```

2. Write and Run a Python script that creates two agents (Planner, Reviewer) and a finalization step.

agents_demo_HW1_AnshikaGoel(018317819).py > main

```python
import argparse
import json
import re
import time
from typing import List
from pydantic import BaseModel, Field, field_validator
from langchain_ollama import ChatOllama
from langchain_core.messages import SystemMessage, HumanMessage
def summaryValidation(s):
    return len(re.findall(r"\b\w[\w'-]*\b", s or ""))
class DataBlock(BaseModel):
    tags: List[str] = Field(..., description="3 tags relevant to title/content.")
    summary: str = Field(..., description="One sentence summary with <= 25 words.")
    issues: List[str] = Field(default_factory=list)
    @field_validator("tags")
    @classmethod
    def _three_tags(cls, v):
        meta_ban = {"json", "planner", "reviewer", "finalizer", "agent", "llm", "model", "prompt"}
        clean, seen = [], set()
        for i in v or []:
            i = (i or "").strip().lower()
            if not i or i in meta_ban:
                continue
            i = re.sub(r"\b(post|here|topic|the|article|and|an|a|of|about|content)\b", "", i).strip()
            if i and i not in seen:
                clean.append(i)
                seen.add(i)
            if len(clean) == 3:
                break
        return clean[:3]
    @field_validator("summary")
    @classmethod
    def _limit_25(cls, v):
        v = re.sub(r"^\s*here\s+is\s+the\s+paraphrased.*?:\s*", "", v, flags=re.I)
        v = re.sub(r"^\s*(this|the)\s+(post|content|article)\s+.*?:\s*", "", v, flags=re.I)
        words = re.findall(r"\b\w[\w'-]*\b", (v or "").strip())
        return " ".join(words[:25])
class AgentJSON(BaseModel):
    thought: str
```

Ln 171, Col 33 Spaces: 4 UTF-8 CRLF {} Pytho

3. Capture outputs and submit a PDF with your console screenshots + answers.

```
(venv) C:\Users\aradh\Desktop\DATA 236 - Distributed Systems\Homework\Homework 1>python agents_demo_HW1_AnshikaGoel(018317819).py --title "Applied Intelligence Systems" --content "Applied Intelligent Systems Laboratory (AISL) is a resea
rch unit at the School of Nuclear Engineering at Purdue University that performs active research dedicated towards muon imaging with special emphasis in nuclear applications and the application of innovative problem solving techniques t
o modeling, simulation and control of complex systems. These techniques include novel signal processing and machine learning methodologies such as artificial neural networks, fuzzy logic, genetic algorithms and evolutionary computing, G
aussian processes, wavelet analysis, Hilbert-Huang transform, expert systems and advanced signal/image processing tools. The research team has state-of-the-art hardware and software facilities, including a high performance computing gri
d already in place to enable the development of a computing architecture to perform large-scale high-fidelity simulations and analyze experimental data, and significant experience with industrial and safeguards applications." --model sm
ollm:1.7b --base_url http://localhost:11434
--- Planner ---
{
  "thought": "Fallback",
  "message": "Draft analyzed and improved; provided concise tags and a <=25-word summary.",
  "data": {
    "tags": [],
    "summary": "Concise summary not provided by the model",
    "issues": [
      "autofix"
    ]
  }
}
--- Reviewer ---
{
  "thought": "Fallback",
  "message": "Draft analyzed and improved; provided concise tags and a <=25-word summary.",
  "data": {
    "tags": [
      "autofix"
    ],
    "summary": "Concise summary not provided by the model",
    "issues": []
  }
}
--- Finalized Output with Timestamp ---
{
  "thought": "Fallback",
  "message": "Draft analyzed and improved; provided concise tags and a <=25-word summary.",
  "data": {
    "tags": [
      "autofix",
      "systems",
      "research"
    ],
    "summary": "Concise summary not provided by the model",
    "issues": []
  },
  "timestamp": "2025-09-08T03:04:42Z"
}
--- Publish Package ---
{
  "title": "Applied Intelligence Systems",
  "email": "you@sjsu.edu",
  "content": "Applied Intelligent Systems Laboratory (AISL) is a research unit at the School of Nuclear Engineering at Purdue University that performs active research dedicated towards muon imaging with special emphasis in nuclear appli
cations and the application of innovative problem solving techniques to modeling, simulation and control of complex systems. These techniques include novel signal processing and machine learning methodologies such as artificial neural n
etworks, fuzzy logic, genetic algorithms and evolutionary computing, Gaussian processes, wavelet analysis, Hilbert-Huang transform, expert systems and advanced signal/image processing tools. The research team has state-of-the-art hardwa
re and software facilities, including a high performance computing grid already in place to enable the development of a computing architecture to perform large-scale high-fidelity simulations and analyze experimental data, and significa
nt experience with industrial and safeguards applications.",
  "agents": [
    {
      "role": "Planner",
      "content": "Draft analyzed and improved; provided concise tags and a <=25-word summary."
    },
    {
      "role": "Reviewer",
      "content": "Draft analyzed and improved; provided concise tags and a <=25-word summary."
    }
  ],
```

```
--- Publish Package ---
{
  "title": "Applied Intelligence Systems",
  "email": "you@sjsu.edu",
  "content": "Applied Intelligent Systems Laboratory (AISL) is a research unit at the School of Nuclear Engineering at Purdue University that performs active research dedicated towards muon imaging with special emphasis in nuclear appli
cations and the application of innovative problem solving techniques to modeling, simulation and control of complex systems. These techniques include novel signal processing and machine learning methodologies such as artificial neural n
etworks, fuzzy logic, genetic algorithms and evolutionary computing, Gaussian processes, wavelet analysis, Hilbert-Huang transform, expert systems and advanced signal/image processing tools. The research team has state-of-the-art hardwa
re and software facilities, including a high performance computing grid already in place to enable the development of a computing architecture to perform large-scale high-fidelity simulations and analyze experimental data, and significa
nt experience with industrial and safeguards applications.",
  "agents": [
    {
      "role": "Planner",
      "content": "Draft analyzed and improved; provided concise tags and a <=25-word summary."
    },
    {
      "role": "Reviewer",
      "content": "Draft analyzed and improved; provided concise tags and a <=25-word summary."
    }
  ],
  "final": {
    "thought": "Fallback",
    "message": "Draft analyzed and improved; provided concise tags and a <=25-word summary.",
    "data": {
      "tags": [
        "autofix",
        "systems",
        "research"
      ],
      "summary": "Concise summary not provided by the model",
      "issues": []
    },
    "timestamp": "2025-09-08T03:04:42Z"
  },
  "submissionDate": "2025-09-08T03:04:42Z"
}
```


Q1. Write the 3 final tags you obtained.


```
"tags": [
  "autofix",
  "systems",
  "research"
],
```


Q2. Paste the final summary (≤25 words).


```
"summary": "Concise summary not provided by the model",
```


Q3. Did the Reviewer agent change anything? (yes/no + explanation)

No, the reviewer didn't change anything. It seems like reviewer was in a fallback state and was not able to execute any meaningful changes.

4. Explanation of each step in your own words
    - The planner agent sends a prompt to model smollm:1.7b via ollama to create JSON string based on blog_title and blog_content. The JSON string should contain the following:
        - Three relevant tags to the blog content and blog title
        - Summary of the blog content which is not more than 25 words

      The JSON string is then extracted using regular expression and then passed onto the reviewer agent.
    - Once the JSON string is created it is then passed onto the reviewer agent to modify the planner output if there are any errors. If the JSON string is correct then, it is returned without any modifications.
    - Once the valid JSON is received by the finalizer, metadata like submission date and timestamp are added and the JSON string can now be used for further processing.