Activity One: System Testing Policy

The system testing policy for the Orientation Chatbot project is designed to ensure that all components of the system function correctly and cohesively as a whole. This policy outlines the principles and responsibilities for system testing, focusing on ensuring the chatbot performs accurately under various conditions, supports multilingual interaction, and remains stable under peak usage loads.

Key principles include:
1. Complete coverage of all major user stories (Ask Orientation Questions, Multilingual Support, Input Validation, Scalability, and Prompt Engineering).
2. Reuse of functional and automated test cases developed in earlier workshops for consistency.
3. System-level testing to be executed in a production-simulated environment.
4. Regular review of test results by the development team and stakeholders.
5. Testing activities must align with the agile sprint schedule and must be completed before deployment.

Activity Two: Detailed System Testing Plan

The system testing plan for the Orientation Chatbot includes both automated and manual tests across five key user stories developed in Workshops 1–8. Testing will be conducted using pytest, Flask's testing client, and performance tools like JMeter as applicable.

Testing Objectives：

- Validate that the chatbot responds correctly to predefined and custom queries.
- Ensure input validation rules are strictly enforced (non-empty, < 500 characters).
- Test multilingual functionality for seamless international student support.
- Verify chatbot performance under simulated high load conditions.
- Ensure modular prompt templates return contextually appropriate responses.

Test Categories：

- Functional Testing
- Performance Testing
- Security Testing (e.g., XSS input sanitization)

- Usability Testing
- Regression Testing

Test Environment：

Testing will be conducted in a controlled environment that mirrors the production deployment stack:

- Backend: Flask App (app.py)
- Frontend: HTML + JS (index.html)
- Styles: CSS (styles.css)
- Language Model: Ollama integrated with LLaMA 3.1
- Testing Tools: pytest, Flask test client, JMeter

Test Execution Plan：

- Execute all previously created automated test cases (15+ test cases).
- Monitor results using pytest logs and performance stats.
- Record pass/fail status, response time, and any anomalies.
- Iterate testing post-fix and conduct regression validation.
- Finalize testing with a demo for the lecturer before deployment approval.