Activity 1:

1. Testing Objectives

• Ensure the chatbot accurately answers orientation-related questions (e.g., event schedules, venue details, required documents).

• Validate the integration of the Ollama model (**llama3.1**) with Flask and command-line interfaces.

• Confirm that conversation history is maintained correctly across user interactions.

• Ensure API endpoints (e.g., **/chat**) handle requests securely and efficiently.

2. Testing Scope

• **Functional Testing**:

  • Web interface (**app.py**): Verify rendering of **index.html** and JSON response correctness.

  • Command-line interface (**test.py**): Validate real-time interaction and context retention.

  • Core modules: Information retrieval, session persistence, error handling.

• **Non-Functional Testing**:

  • Performance: Test **/chat** endpoint latency under load (using JMeter).

  • Security: Check for SQL injection risks in user input handling (despite SQLite usage).

  • Compatibility: Ensure cross-browser support for the Flask web interface.

3. Testing Methods

• **Automated API Testing**: Use Postman to simulate POST requests to **/chat** with sample questions.

• **Manual CLI Testing**: Execute **test.py** with edge-case inputs (e.g., empty strings, special characters).

• **Unit Testing**: Add pytest cases for Flask routes and Ollama response validation (not yet implemented in current code).

4. Responsibilities

• **Development Team**: Implement unit tests for **app.py** and **test.py**, fix critical defects.

• **Testing Team**: Design end-to-end test cases for both interfaces and execute load testing.

• **DevOps Team**: Configure CI/CD pipelines (e.g., GitHub Actions) for automated testing.

5. Quality Standards

• All API responses must return HTTP status codes correctly (e.g., 400 for invalid input).

• Command-line interface must handle input interruptions gracefully.

• Average response time ≤ 3 seconds under 100 concurrent users.

Activity 2:

1. Test Case Examples

Component	**Test Scenario**	**Expected Result**
Web Interface (**app.py**)	User submits "What is the dress code for the welcome session?"	Returns "Smart Casual (No shorts or slippers)" and HTTP 200.

Component	Test Scenario	Expected Result
CLI Interface (**test.py**)	User inputs "Tell me about Campus Exploration"	Returns details of the 14 Jan 2025 event.
Error Handling	Send empty JSON to **/chat** endpoint	Returns **{"error": "No question provided"}**, HTTP 400.
Session Persistence	Ask follow-up question: "Where is Block C?"	Correctly references previous context about Block C.

2. Testing Environment

- **Web Server**: Local Flask server (**app.run(debug=True)**).
- **CLI Execution**: Python 3.10+ environment with Ollama dependencies.
- **Tools**:
  - Postman (API validation), JMeter (load testing).
  - Pytest (unit testing, to be added).

3. Timeline

Phase	Timeframe	Deliverables
Unit Test Implementation	Week 1	pytest scripts for **/chat** and error handling
Functional Testing	Week 2	Pass/fail reports for web and CLI interfaces
Load & Security Testing	Week 3	JMeter reports, OWASP ZAP scan results

4. Risks & Mitigations

- **Risk 1**: Ollama model (**llama3.1**) fails to generate context-aware responses.
  Mitigation: Add validation logic to check response relevance against training data.
- **Risk 2**: Flask debug mode exposes security vulnerabilities.
  Mitigation: Disable debug mode in production and add input sanitization.

5. Exit Criteria

- All unit tests pass with ≥ 90% code coverage.
- Web interface responds correctly to 95% of sample orientation questions.
- No critical security vulnerabilities (e.g., XSS, SQLi) detected.