Activity 1:

1. Ask Orientation Questions

- **Tasks:**
  1. Design chatbot interface for question input (2 days)
  2. Develop NLP model for recognizing orientation-related queries (3 days)
  3. Integrate with Student Affairs data for real-time answers (3 days)
  4. Implement predefined questions & quick replies (2 days)
  5. Test response accuracy and performance (3 days)
- **Total Estimate:** 5 Story Points (~5 Days)

2. Multilingual Support for International Students

- **Tasks:**

  1. Integrate Google Translate API into app.py (4 days)
  2. Add language selection dropdown in index.html (2 days)
  3. Validate translations through test cases (e.g., "体检时间" → "Medical Check-Up time") (3 days)
  4. Ensure context retention in translated responses (3 days)
  5. Update documentation for multilingual support (1 day)

- **Total Estimate:** 6 Story Points (~10 Days)

3. Scalability and Load Testing

- **Tasks:**

  1. Set up a test environment for load testing (2 days)
  2. Simulate 300 concurrent users with testing tools (3 days)
  3. Monitor system performance and adjust parameters (3 days)
  4. Optimize response time under 3 seconds (2 days)

- **Total Estimate:** 8 Story Points (~10 Days)

4. Basic Input Format Validation

- **Tasks:**

  1. Enforce 500-character limit for /chat requests in app.py (1 day)
  2. Reject empty inputs with HTTP 400 (1 day)
  3. Add error messages for invalid inputs (1 day)
  4. Update test.py to validate input rules (2 days)

- **Total Estimate:** 1 Story Points (~5 Days)

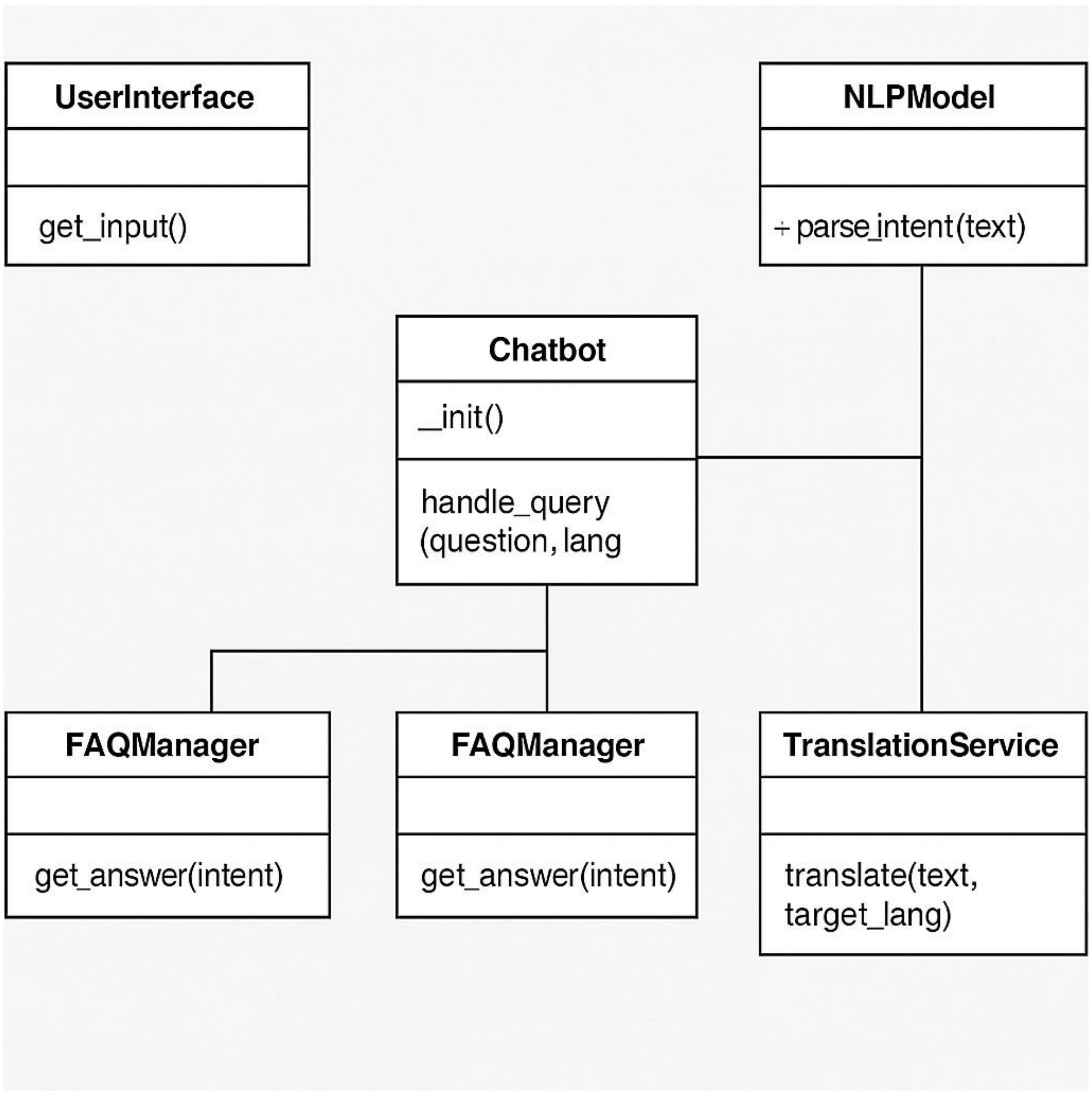
5. Prompt Engineering for Optimized Chatbot Responses

- **Tasks:**

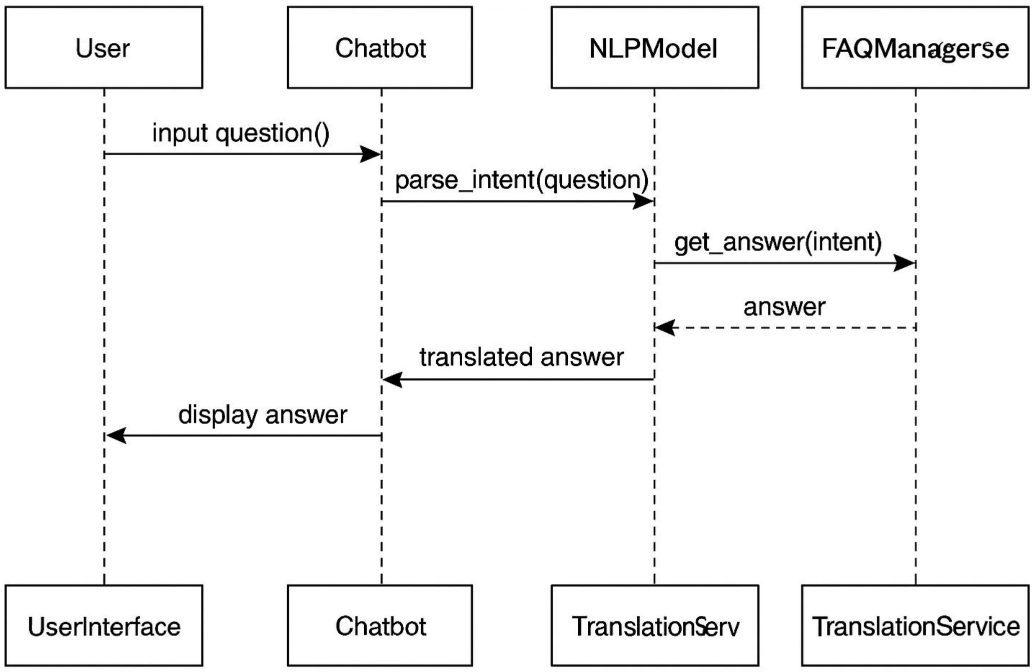
  1. Develop structured prompt templates (3 days)
  2. Implement version control for prompts (3 days)
  3. Create a sandbox for prompt testing (4 days)
  4. Conduct test conversations for accuracy (3 days)
  5. Document best practices and guidelines (2 days)

- **Total Estimate:** 5 Story Points (~15 Days)

Activity 2:



Activity 3:



Activity 4:

Code：

```
# Updated code with multilingual support and input validation
class UserInterface:
    def get_input(self):
        return input("Ask a question (max 500 characters): ")

class Chatbot:
    def __init__(self):
        self.nlp = NLPModel()
        self.faq = FAQManager()
        self.translator = TranslationService()

    def handle_query(self, question, lang='en'):
        if not question or len(question) > 500:
            raise ValueError("Invalid input")
        intent = self.nlp.parse_intent(question)
        answer = self.faq.get_answer(intent)
        return self.translator.translate(answer, lang)

class TranslationService:
    def translate(self, text, target_lang):
        # Integrate Google Translate API here
        return f"[Translated to {target_lang}] {text}"

# Example usage
ui = UserInterface()
chatbot = Chatbot()
question = ui.get_input()
try:
    response = chatbot.handle_query(question, lang='zh')
    print(f"Chatbot: {response}")
except ValueError:
    print("Error: Please enter a valid question (max 500 characters).")
```