Canvas Assistant Chatbot: Design Report

Author: Mukhil Sundararaj Gowthaman
Course: CS-GY 6543 - Human-Computer Interaction
NetID: mg8192
Date: November 2, 2025
Deployed Application: https://hci-chatbot-eight.vercel.app/

Executive Summary

The Canvas Assistant is a conversational chatbot designed to help creative freelancers manage their finances through the Canvas financial app. Built with a hybrid architecture combining rule-based intent detection and optional AI enhancement, the chatbot provides guided assistance for invoicing, expense tracking, tax savings, and project management. The system prioritizes clarity, explainability, and ethical guardrails, explicitly declining financial advice requests while offering transparent explanations of its reasoning. The implementation demonstrates graceful degradation, functioning fully without AI while providing enhanced natural language responses when available. This report documents the design guidelines followed, analyzes strengths and limitations, and addresses ethical considerations in chatbot interactions.

---

1. Chatbot Overview and Goals

The Canvas Assistant chatbot serves as a digital helper for creative freelancers using the Canvas financial management application. The primary goal is to reduce financial anxiety and administrative burden by providing instant, clear guidance on app features and workflows. The chatbot handles common tasks such as creating invoices, tracking expenses, understanding tax savings mechanisms, and managing projects.

The system is designed with a dual approach: a reliable rule-based foundation that ensures consistent, predictable responses, complemented by optional AI enhancement for more natural, context-aware interactions. This hybrid model ensures the chatbot remains fully functional even when AI services are unavailable, while providing enhanced user experience when AI is enabled.

---

2. Design Guidelines Followed

2.1 Hybrid Architecture: Rule-Based + AI Enhancement

The chatbot implements a hybrid architecture that combines deterministic rule-based intent detection with optional AI-powered response generation. The core intent detection logic (`detect_intent` function) uses keyword matching and normalization to identify user intent reliably. When an OpenAI API key is available, the system enhances responses using GPT-4o-mini via the `generate_ai_response` function, which is lazy-loaded to prevent import failures on serverless platforms.

This design follows the principle of graceful degradation: the chatbot functions identically with or without AI, ensuring reliability and avoiding vendor lock-in. The AI enhancement layer operates as an optional refinement rather than a dependency, maintaining system stability.

2.2 Intent-Based Rule Matching and Normalization

The chatbot uses a structured intent detection system with 11+ distinct intents covering greeting, app information, invoicing (multi-step), tax savings, expenses, payment errors, projects, bank connections, financial advice detection, payment clarification, and fallback handling. User messages are normalized using the `normalize` function, which lowercases text, strips whitespace, and collapses multiple spaces, ensuring consistent keyword matching regardless of formatting variations.

2.3 Progressive Disclosure and Multi-Step Flows

Complex topics are broken into manageable, step-by-step interactions. For example, when a user asks about invoicing, the chatbot first presents a numbered menu (1. Create, 2. Track, 3. Handle late payments) rather than overwhelming the user with all information at once. The system tracks follow-up context using `lastIntent` in the conversation context, allowing it to interpret subsequent messages like "1" or "create" as selections from the menu.

The Tax Jar feature similarly uses progressive disclosure: initial responses explain the concept, then offer follow-up options to learn about percentage suggestions or customization settings. This approach reduces cognitive load and guides users through complex workflows.

2.4 Clear Rationale and Suggestions

Every response includes a `rationale` field explaining why the chatbot chose that particular response, which is displayed to users via a "Why this?" toggle in the UI. This transparency helps users understand the system's reasoning and builds appropriate trust. Additionally, each response includes contextually relevant suggestion chips (e.g., "Invoices", "Tax Jar", "Expenses") that enable quick navigation and reduce typing effort.

The `build_response` function generates these suggestions dynamically based on the detected intent, ensuring users always have clear next steps available.

2.5 Error Tolerance and Graceful Fallback

The system implements multiple layers of error handling:

- **Intent-level fallback**: If no intent matches, the `fallback` intent provides a friendly message suggesting common topics.
- **AI-level fallback**: If AI generation fails or returns an invalid response, the system automatically falls back to the rule-based response.
- **Network-level fallback**: The frontend displays user-friendly error messages if API requests fail, maintaining a professional user experience.

2.6 Conversational Niceties and Personalization

The chatbot includes time-aware greetings via the `dynamic_greeting` function, which adjusts messages based on the time of day (morning, afternoon, evening). It also handles conversational markers like "thank you" and "goodbye" appropriately, maintaining a friendly, professional tone throughout interactions.

2.7 Financial Advice Guardrails

A critical design guideline is the explicit prohibition of financial advice. The chatbot includes a dedicated `financial_advice_decline` intent that detects queries requesting financial, investment, or tax filing advice (e.g., "should I register as an LLC", "what should I invest in"). When detected, the chatbot politely declines and redirects users to organize their data for discussion with professional advisors, never providing specific recommendations.

2.8 Persona-Driven AI System Prompt

When AI is enabled, the system uses a comprehensive system prompt that defines the chatbot's persona, interaction style, procedural rules, and guardrails. The prompt emphasizes:

- **Proactive guidance** without being prescriptive
- **Clarity over clutter** with concise, well-structured responses
- **Calm, reassuring tone** to reduce financial anxiety
- **Explicit boundaries** prohibiting financial advice
- **Escalation paths** to human support when needed

This ensures AI-generated responses align with the chatbot's ethical framework and design principles.

2.9 Security and Deployability

The system is designed for serverless deployment on Vercel, with proper CORS configuration for cross-origin requests. The AI client is lazy-loaded to avoid import-time failures, and environment variables are used for sensitive configuration (API keys). The frontend automatically detects local vs. production environments, routing requests appropriately.

---

3. Interaction Flows and Examples

3.1 Invoicing Multi-Step Flow

User: "How do I create an invoice?"
Bot: "I can definitely help with invoices. Are you looking to: 1) Create a new invoice 2) Track an existing invoice 3) Handle a late payment"
User: "1"
Bot: "Great — here's how to create an invoice: 1) Open Invoices and tap '+'. 2) Add client, line items, and due date. 3) Preview, then Send. Tip: Add payment terms so clients know when and how to pay."

This flow demonstrates progressive disclosure: the chatbot first clarifies the user's specific need before providing detailed instructions.

3.2 Tax Jar Follow-Up Flow

User: "What is the Tax Jar?"
Bot: "The 'Tax Jar' automatically sets aside a portion of your income for taxes. We recommend a percentage based on your income, but you can adjust it anytime in Settings. Would you like to know more about how we suggest a percentage, or how to customize it?"
User: "How do you suggest a percentage?"
Bot: "We estimate a savings percentage using your recent income pace and typical self-employment tax rates. It's a starting point — not advice — and you can fine-tune it anytime. Rule of thumb: 20–30% works for many freelancers, but your situation may differ."

This flow shows how the chatbot maintains context and provides branching options for deeper exploration.

3.3 Payment Ambiguity Clarification

User: "I need help with payment"
Bot: "Got it. Are you trying to log a payment you received from a client, or track an expense you paid?"

This demonstrates the chatbot's ability to recognize ambiguous queries and ask clarifying questions rather than guessing.

3.4 Financial Advice Detection and Decline

User: "Should I register as an LLC?"
Bot: "As an AI assistant for the Canvas app, I can't provide financial advice, but I can help you organize your business data to discuss with a professional advisor. Would you like help tracking your expenses, invoices, or Tax Jar savings?"

This flow illustrates the ethical guardrail in action, politely redirecting users away from advice requests while maintaining helpfulness.

---

4. Strengths and Limitations

4.1 Strengths

Hybrid Approach with Graceful Degradation: The combination of rule-based reliability and optional AI enhancement provides the best of both worlds: consistent, predictable responses when AI is unavailable, and natural, context-aware interactions when AI is enabled. This architecture ensures the chatbot never fails due to external service dependencies, while still benefiting from AI enhancement when available.

Clear Explainability: The rationale display feature allows users to understand why the chatbot responded in a particular way, promoting transparency and building appropriate trust. This aligns with HCI principles of explainable AI systems.

Comprehensive Domain Coverage: With 11+ distinct intents, the chatbot covers the primary use cases of the Canvas app, from basic greetings to complex multi-step workflows. The suggestion system guides users through these capabilities efficiently.

Ethical Guardrails: The explicit financial advice detection and decline mechanism protects users from potentially harmful misinformation and maintains legal compliance boundaries.

Low Latency: The rule-based primary path ensures fast response times, with AI enhancement adding minimal overhead when enabled.

Multi-Step Flow Support: The context tracking system enables sophisticated multi-turn conversations, allowing users to navigate complex workflows naturally.

4.2 Limitations

Rule Coverage Gaps: Keyword-based matching may miss paraphrases or alternative phrasings of the same intent. For example, "invoice creation" might not match if only "invoice" or "create invoice" are in the keyword list. This limitation is partially mitigated by AI enhancement when available, but remains a constraint in rule-only mode.

Ambiguity Edge Cases: Some user queries may be genuinely ambiguous, and while the system includes clarification mechanisms (e.g., payment vs. expense), not all edge cases are handled. Users may need to rephrase their queries if the initial intent detection fails.

Limited Personalization and State: The system tracks only the immediate conversation context (`lastIntent`), without persistent memory across sessions or user profiles. This means the chatbot cannot remember previous conversations or personalize responses based on user history.

AI Dependency Optional: While graceful degradation is a strength, users without an API key may experience responses that feel more "robotic" or templated. The static responses, while functional, lack the natural language variation that AI provides.

No Multi-Turn Memory: Beyond the immediate context for follow-up flows, the chatbot does not maintain conversation history. This limits its ability to reference earlier parts of a conversation or build upon previous interactions.

Accessibility Testing TBD: While the frontend includes ARIA labels (`aria-live`, `aria-label`) and follows responsive design principles, comprehensive accessibility testing with users who rely on assistive technologies has not been conducted. The implementation follows best practices, but user validation is needed.

Feedback Controls Non-Functional: The thumbs up/down feedback controls in the UI are currently placeholders (no-op functions). While the UI supports feedback collection, the infrastructure to store and analyze this data is not implemented.

5. Ethical Considerations

5.1 Transparency and Disclaimers

The chatbot includes explicit disclaimers about its limitations. The frontend displays a footer note: "Canvas Assistant offers general guidance, not legal or tax advice." The system prompt for AI-generated responses explicitly states the chatbot's boundaries and prohibits financial advice. This transparency ensures users understand the chatbot's role and limitations, preventing over-reliance on potentially inaccurate information.

5.2 Data Minimization

The chatbot implements a stateless design: no personally identifiable information (PII) is stored, and conversation history is not persisted. Each request is processed independently, with only the immediate context (`lastIntent`) passed between turns. This minimizes privacy risks and ensures user data is not retained beyond the immediate interaction.

5.3 Security Claims

When discussing bank connections, the chatbot makes specific security claims: "Your credentials are encrypted and never stored by Canvas." While this accurately reflects the intended design (integration via Plaid), the chatbot should be careful not to overstate security guarantees. The implementation relies on third-party services (Plaid) for actual security, and the chatbot's role is limited to explaining the process.

5.4 Error Handling and User Safety

The system implements graceful error handling at multiple levels: AI generation failures fall back to static responses, network errors display user-friendly messages, and invalid responses are filtered. This ensures users never encounter system errors or technical jargon, maintaining a professional experience even when underlying systems fail.

5.5 Bias and Coverage Limits

The keyword-based matching system may exhibit bias toward certain phrasings, dialects, or languages. Users who express queries in non-English languages, technical jargon, or regional dialects may experience reduced accuracy. Similarly, users with limited digital literacy may struggle with the interface or phrasing queries in ways the system recognizes. Future enhancements should include more diverse training data and multilingual support.

5.6 Accessibility and Inclusive Language

The frontend implements ARIA labels and responsive design, but comprehensive accessibility testing is needed. The system prompt emphasizes plain language and avoiding jargon, which supports users with varying levels of financial and technical knowledge. However, the chatbot's effectiveness for users with disabilities or assistive technologies requires validation through user testing.

5.7 Guardrails and Boundaries

The chatbot includes explicit guardrails prohibiting financial, investment, and tax filing advice. The `financial_advice_decline` intent actively detects and declines such requests, redirecting users to organize data for professional consultation. The system prompt reinforces these boundaries, and an escalation path (support@canvasapp.com) is provided for users who need human assistance.

5.8 AI Responsibility and Reliability

When AI is enabled, the system implements several responsible AI practices:

- **Lazy-loading**: The OpenAI client is imported only when needed, preventing import-time failures on serverless platforms.
- **Temperature and token limits**: AI responses use controlled parameters (temperature=0.5, max_tokens=120) to ensure consistency and prevent excessive generation.

- **Fallback mechanisms**: If AI generation fails or returns invalid responses, the system automatically falls back to rule-based responses, ensuring reliability.

These practices ensure the AI enhancement layer does not compromise system stability or user experience.

---

6. Implementation Details

6.1 Architecture

The chatbot is built with a Flask (Python) backend and a vanilla HTML/CSS/JavaScript frontend. The backend handles intent detection, response generation, and AI integration, while the frontend provides the chat interface and manages user interactions.

6.2 Runtime and Deployment

The application is deployed on Vercel as a serverless function. The `vercel.json` configuration routes API requests to `api/chat.py` (a serverless function) and static requests to `index.html`. The Flask app is exposed through the serverless function, which imports the main application module.

6.3 AI Integration

AI enhancement is provided by OpenAI's GPT-4o-mini model, accessed via the OpenAI API. The integration is optional: if the `OPENAI_API_KEY` environment variable is not set, the chatbot uses static rule-based responses exclusively. When enabled, the AI enhances responses while maintaining the same intent detection and guardrails.

6.4 API Endpoint

The chatbot exposes a single POST endpoint: `/api/chat`. The request format is:

```json
{
  "message": "user message text",
  "context": {
    "lastIntent": "optional previous intent"
  }
}
```

The response format is:

```json
{
  "response": "bot response text",
  "intent": "detected intent name",
  "followUpIntent": "optional follow-up context",
  "rationale": "explanation of response",
  "suggestions": ["suggestion", "chips", "array"]
}
```

6.5 Deployed Application

The chatbot is deployed and accessible at: **https://hci-chatbot-eight.vercel.app/**

The application is fully functional and can be tested interactively through the web interface. Users can explore the various intents, multi-step flows, and AI-enhanced responses (if API key is configured).

References

1. Norman, D. (2013). *The Design of Everyday Things: Revised and Expanded Edition.* Basic Books.
2. Nielsen, J. (1994). *Usability Engineering.* Morgan Kaufmann.
3. Bickmore, T. W., & Picard, R. (2005). "Establishing and maintaining long-term human-computer relationships." *ACM Transactions on Computer-Human Interaction*, 12(2), 293-327.
4. Følstad, A., & Brandtzæg, P. B. (2017). "Chatbots and the new world of HCI." *Interactions*, 24(4), 38-42.
5. OpenAI. (2024). "GPT-4o-mini Model Documentation." OpenAI Platform. https://platform.openai.com/docs/models/gpt-4o-mini

---

This report documents the design, implementation, and ethical considerations of the Canvas Assistant chatbot, developed as part of CS-GY 6543 - Human-Computer Interaction coursework.