Software Design Document

MujahidGPT

Submitted by

Mujahid Hussain (F22BINFT1M01235)

Submitted to

Mr. Muzamil ur Rehman

Department of Information Technology

Faculty of Computing

The Islamia University of Bahawalpur

Summary

MujahidGPT is a web-based AI-powered personal assistant chatbot developed to provide 24/7 access to JHK Solution's company information, services, team details, pricing, and contact information for clients. It enables natural language interaction, delivers accurate instant responses when company personnel are unavailable, and collects optional user feedback for improvements. The system is lightweight, reliable, fast, and fully accessible across Mac, Linux, and Windows via any modern browser.

The application is built using Python for core logic and backend processing, Gradio for the interactive chat interface, HTML and CSS for custom styling, and the Groq API for high-speed natural language understanding and generation (API keys sourced via thenajafigroup.online or official Groq console). Hosting is on Hugging Face Spaces for the live Gradio demo, with supplementary static content on Google Sites and source code/version control on GitHub. The design follows a modular, client-server approach with minimal local compute, offloading intelligence to Groq for efficiency, ensuring cross-platform compatibility, low latency (<2 seconds response time), and easy academic demonstration/maintenance. Security emphasizes API key protection, HTTPS, and no persistent user data collection.

This project reduces client wait times, enhances service accessibility, and aligns with JHK Solution's goal of efficient client support. The Software Design Document details the internal architecture, components, data flows, and implementation choices, serving as a technical guide for development, testing, integration, and future enhancements while referencing the SRS for requirements.

Table of Content

Table of Contents

📗 Software Design Document (SDD)

MujahidGPT

1. Introduction

1.1 Purpose of the Document

This Software Design Document (SDD) describes the internal design, technical structure, and implementation approach of MujahidGPT. It explains how the system fulfills the requirements from the Software Requirements Specification (SRS), serving as a guideline for developers, testers, the supervisor (Mr. Muzamil ur Rehman), and future maintainers by detailing architecture, modules, data flows, integration points, and security/performance mechanisms. Unlike the SRS (focused on what the system must do), this document focuses on how it is built.

1.2 Scope of the System

MujahidGPT is a browser-based conversational AI assistant for JHK Solution clients. It handles natural language queries about company services, provides instant accurate information, guides on service requests when staff are unavailable, and collects feedback. The scope of this document covers: overall architecture, module/component design, data structures, Groq API integration, UI implementation, security, error handling, and performance/scalability. Functional and non-functional details are referenced from the SRS and not repeated here.

1.3 Definitions, Acronyms, and Abbreviations

- LLM: Large Language Model
- API: Application Programming Interface
- Groq API: Fast inference service for LLMs (compatible with OpenAI format)
- Gradio: Python library for building interactive web UIs
- Hugging Face Spaces: Platform for hosting ML demos
- JHK Solution: The target company/client
- NLP: Natural Language Processing

- SDD: Software Design Document
- SRS: Software Requirements Specification

1.4 References

- Groq API Documentation (console.groq.com/docs)
- Gradio Documentation (gradio.app/docs)
- Hugging Face Spaces Guide (huggingface.co/docs/hub/spaces)
- Python Official Documentation
- IEEE Std 1016-2009 (Software Design Descriptions)
- Groq Quickstart and Models Reference

1.5 System Overview

MujahidGPT operates as a client-server web application. Users interact via a Gradio-powered chat interface in the browser; Python backend processes inputs, constructs prompts with JHK knowledge, calls the Groq API for intelligent responses, formats outputs, and logs interactions/feedback. No heavy database is used—static JSON for knowledge, file-based logging. Hosting on Hugging Face Spaces ensures easy deployment and scaling.

2. Design Considerations

2.1 Design Goals

- Deliver fast, reliable responses (<2s)
- Ensure simple, intuitive chat UI
- Maximize cross-platform accessibility (Mac/Linux/Windows browsers)
- Minimize local compute via Groq offloading
- Support easy maintenance and academic demonstration
- Emphasize modularity and separation of concerns

2.2 Assumptions and Constraints

Assumptions: Stable internet, Groq API availability, accurate JHK data provided.

Constraints: Python/Gradio ecosystem only, free-tier hosting limits, no custom servers, reliance on external Groq API, browser-only access.

2.3 Development Environment

- Language: Python 3.10+
- UI: Gradio + HTML/CSS
- API: Groq (groq Python library)
- Hosting: Hugging Face Spaces
- Version Control: GitHub
- Testing: Local browser + Spaces preview

3. System Architecture Design

3.1 Architectural Overview

Adapted three-tier structure:

- Presentation: Browser-rendered Gradio chat UI
- Application: Python scripts (session handling, prompt engineering, API calls, formatting)
- Data: JSON files (knowledge) + file logs (conversations/feedback) External integration: Groq API endpoint for LLM inference.

Three-Tier Adapted Architecture Diagram (described): Browser → Gradio frontend → Python backend logic → Groq API (cloud) → response back; parallel static JSON load for knowledge.

Deployment Diagram (described): Client browser → Hugging Face Spaces (Gradio app container) → Groq cloud API; optional Google Sites for static info pages; GitHub repo for source.

3.2 Technology Stack

- Frontend: Gradio, HTML, CSS
- Backend: Python, groq library
- Data: JSON files

- Hosting: Hugging Face Spaces, Google Sites, GitHub

4. Module Design

4.1 Chat Interface Module

Gradio-based chat handles input textbox, message history display, typing indicator, feedback buttons (thumbs up/down).

4.2 Conversation Manager Module

Maintains session context, appends history to prompts, injects JHK knowledge/system instructions.

4.3 API Integration Module

Securely calls Groq chat completions endpoint, handles retries/errors, parses responses.

4.4 Knowledge Retrieval Module

Loads JSON company data, performs keyword/semantic matching to enrich prompts.

4.5 Feedback & Logging Module

Captures ratings/comments, appends to timestamped files for admin review.

5. Data Design

5.1 Data Overview

Lightweight file-based storage (no RDBMS) for simplicity and deployment ease. Follows structured format to ensure quick access/integrity.

Entity-Relationship Overview (described): CompanyInfo (static) → ConversationSession (ephemeral) → FeedbackEntry (logged).

Data Structure Diagram (described): JSON tree for knowledge; flat append logs for sessions/feedback.

5.2 Data Descriptions

- company_knowledge.json: Services, contacts, FAQs
- session_logs/: Timestamped conversation JSON
- feedback_logs/: User ratings/comments

6. Component/Class Design

6.1 Component Design Overview

Modular Python components (functions/classes) for reusability.

Hierarchy Diagram (described): Main app → ChatSession → KnowledgeBase + GroqClient + ResponseFormatter.

6.2 Component Responsibilities

- ChatSession: Manages history/context
- KnowledgeBase: Loads/queries static data
- GroqClient: API calls + error handling
- ResponseFormatter: Cleans/styles output

7. Sequence Design

7.1 Sequence Design Overview

Sequences show component interactions over time for key flows.

User Query Sequence Diagram (described): User → Gradio UI → Conversation Manager → Knowledge Retrieval → GroqClient → API → Formatter → UI display.

API Call & Response Sequence Diagram (described): Prompt construction → Groq call → parse → fallback if error.

Feedback Submission Sequence Diagram (described): Thumbs click → log append → thank you message.

8. User Interface Design

8.1 UI Design Overview

Single-page responsive chat: input box, bubbles (user blue, bot green), feedback icons. Focuses on simplicity and speed.

UI Navigation Flow Diagram (described): Chat home → optional help/about links.

8.2 Chat Interface

Text input, scrollable history, typing animation, clear error/feedback prompts.

9. Security Design

9.1 Security Design Overview

Protects API keys, prevents injection, anonymous access.

Authentication & Access Flow Diagram (described): No login; env vars for keys, HTTPS enforced.

9.2 Data Security

Keys in environment/secrets, HTTPS transmission, no sensitive user storage.

9.3 Input Validation

Length limits, sanitization against prompt attacks.

10. Error Handling and Validation Design

10.1 Error Handling Overview

Graceful fallbacks (static replies on API fail), user-friendly messages.

Error Handling Flow Diagram (described): Detect → log → fallback → inform user.

10.2 Validation Strategy

Query length, content checks, duplicate prevention in logs.

11. Performance and Scalability Design

11.1 Performance Design

Optimized prompts, fast Groq inference for <2s responses.

Request–Response Flow Diagram (described): Input → process → API → output.

11.2 Scalability Design

Hugging Face auto-scaling; modular for knowledge/model updates.

Scalability Expansion Diagram (described): Add models/knowledge without redesign.

12. Conclusion

This SDD provides a detailed, practical design for MujahidGPT, realizing SRS requirements through efficient, modern tools. It ensures reliable, fast, accessible implementation suitable for JHK Solution support and academic evaluation.

Note:

Functional/non-functional requirements are in the SRS. This document focuses on technical design and implementation.

13. References

[1] Groq API Reference, Groq Console, 2026.

[2] Gradio Documentation, gradio.app, 2026.

[3] Hugging Face Spaces SDK Guide, huggingface.co/docs, 2026.

[4] IEEE Std 1016-2009, Software Design Descriptions.

[5] Python Documentation, python.org.