Assignment 2

Scope & Assumptions -

This project uses an existing, publicly available lightweight LLM (1–8B parameters, quantized) for CPU-only inference on a single low-cost host (2–4 vCPU, ≤8–16 GB RAM). No training or fine-tuning is in scope. The frontend architecture is framework-agnostic (no React/Angular assumptions) and interacts with the backend via REST/SSE APIs. Caching is local to reduce recomputation and outbound calls.

Functional Requirements:

Core Requirements:

1. Chat Interface: The system shall provide a browser-based interface for users to enter prompts and receive responses from the LLM.
2. Session Management: The system shall maintain history of conversations, allowing users to create, view, rename, and delete sessions.
3. Response Caching: The system shall cache responses locally to reduce repeated computation and improve response times.
4. Inference API: The backend shall expose APIs for prompt–response inference, supporting synchronous requests and streamed output.
5. Parameter Control: The inference API shall accept basic parameters such as maximum tokens, temperature, and top_p.
6. Usage Reporting: The system shall return token usage and latency statistics with each response.
7. Cache Management: The system shall provide admin functions to clear the cache and display cache statistics (hit rate, memory usage).
8. Admin Console: The system shall include an administrative interface for monitoring CPU/RAM usage, cache state, and system health.
9. System Monitoring: The backend shall expose health check endpoints and telemetry for administrators.
10. Authentication: The system shall allow secure login for users and administrators.
11. Role-Based Access: Administrative features shall only be available to users with administrator privileges.
12. Developer API Access: The system shall issue API keys for developers to access the inference service programmatically.
13. API Key Management: The system shall support rotation and revocation of developer API keys.
14. Configuration: The system shall allow runtime configuration of model path, quantization settings, cache size, and ports.
15. Graceful Shutdown: The system shall drain in-flight requests and persist cache metadata during shutdown.

Additional / Creative Requirements:

16. Data Portability: Users shall be able to export and import sessions in JSON format for backup or migration.
17. Dark/Light Theme Toggle: The frontend shall allow users to switch between dark and light themes for accessibility.
18. Session Summarization: The system shall provide automatic summaries of long sessions using the model.
19. Developer Dashboard: The system shall display per-user API usage and cache hit/miss statistics.

Non-Functional Requirements:

Core Requirements:

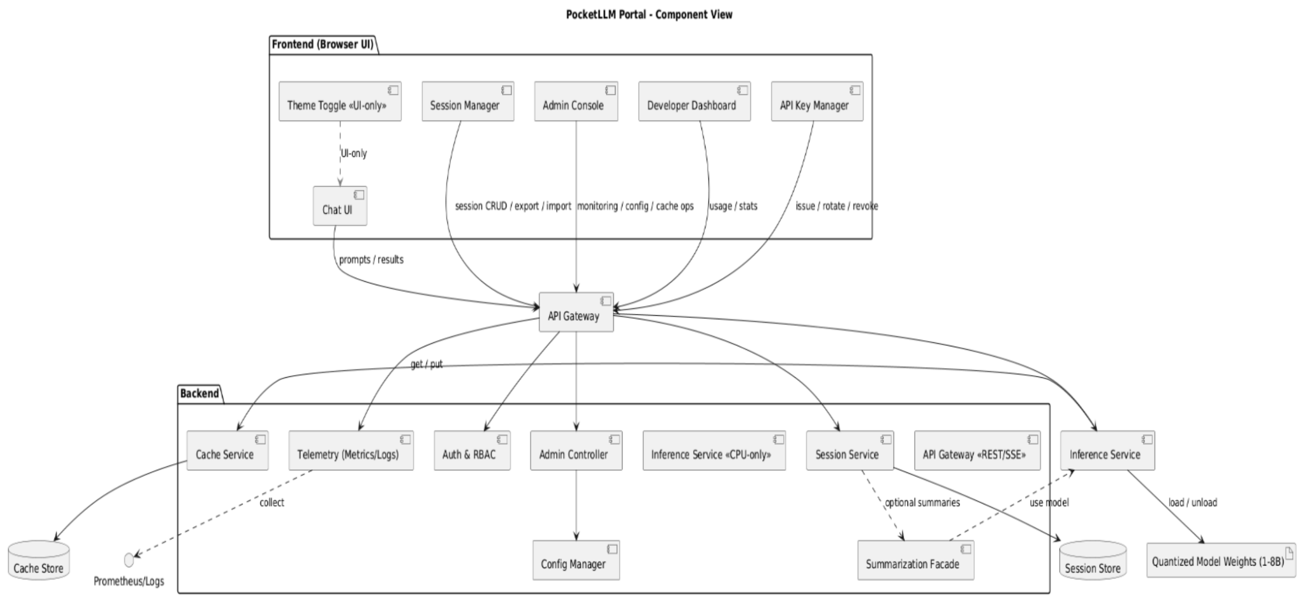
1. Resource Constraints: The system shall run entirely on CPU within 2–4 vCPUs and up to 8–16 GB RAM.
2. Model Size: The system shall support compact quantized models in the 1–8 billion parameter range.
3. Performance: Cached responses shall be delivered in under 1 second; non-cached responses shall be delivered within 2–10 seconds depending on hardware.
4. Throughput: The system shall sustain at least 1.5–3 tokens per second depending on CPU resources.
5. Reliability: The system shall restart within 10 seconds without loss of cache or session data.
6. Logging: The system shall record structured logs of requests, errors, and cache activity for monitoring.
7. Usability: The web interface shall be simple, responsive, and intuitive for non-technical users.
8. Accessibility: The interface shall comply with WCAG 2.1 AA, supporting keyboard navigation and screen readers.
9. Security: User credentials shall be securely stored; API keys shall be long, random, and revocable.
10. Transport Security: All communication between client and server shall use HTTPS/TLS.
11. Portability: The system shall be deployable as a Docker container or standalone binary on consumer desktops or entry-level cloud VMs.
12. Maintainability: The architecture shall be modular, enabling independent updates to frontend and backend components.

Additional / Creative Requirements:

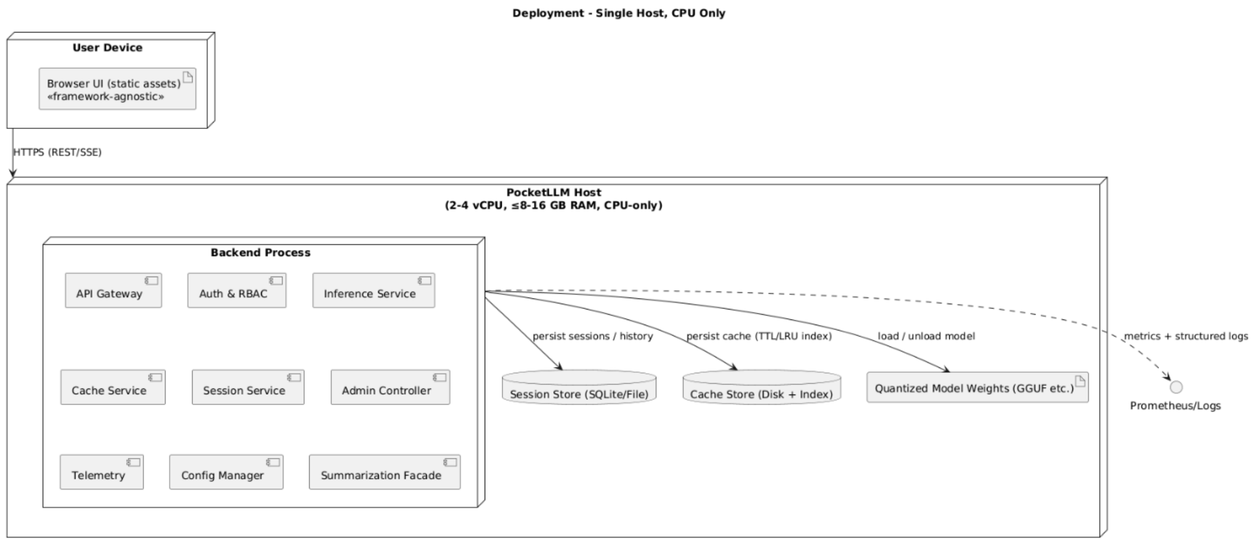
13. Cache Quality: The system shall target at least a 35% cache hit rate under repetitive usage, with LRU eviction and tunable TTL.
14. Observability: The system shall expose Prometheus metrics and provide structured JSON logs for integration with monitoring tools.
15. Privacy: The inference path shall not rely on outbound third-party API calls, ensuring all processing is local.
16. Data Retention Control: Administrators shall be able to configure automatic deletion of session histories after a defined retention period.

<u>Architecture:</u>

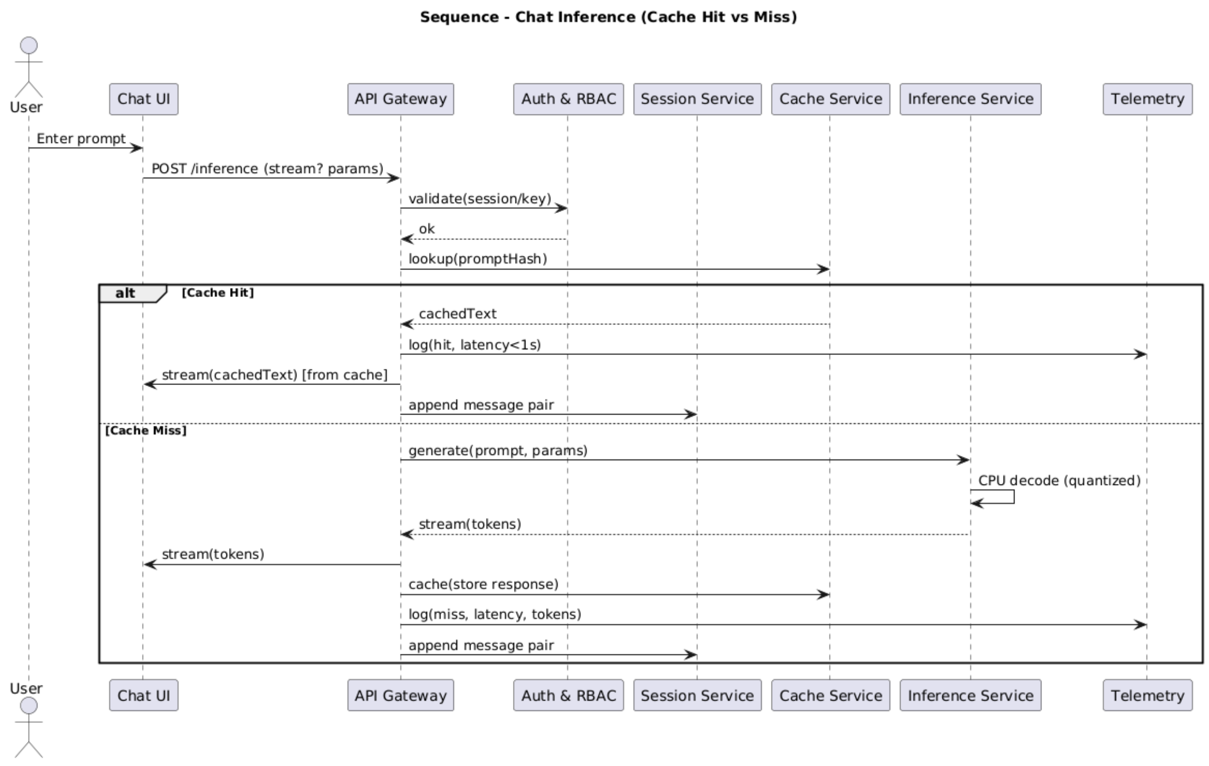
Component View:



Deployment View:


Deployment - Single Host, CPU Only
User Device
Browser UI (static assets)
«framework-agnostic»
HTTPS (REST/SSE)
PocketLLM Host
(2-4 vCPU, ≤8-16 GB RAM, CPU-only)
Backend Process
API Gateway
Auth & RBAC
Inference Service
Cache Service
Session Service
Admin Controller
Telemetry
Config Manager
Summarization Facade
persist sessions / history
persist cache (TTL/LRU index)
load / unload model
metrics + structured logs
Session Store (SQLite/File)
Cache Store (Disk + Index)
Quantized Model Weights (GGUF etc.)
Prometheus/Logs

Chat View:



Online tools:
Some of the requirement phrasing and industry standards (e.g., usability, accessibility, security) were refined with the assistance of ChatGPT to ensure completeness and professional terminology.