MCP Problem Statement (Patient Record Summary)

Problem:

Dental clinicians spend varying time, from low to significant, reviewing past patient records for every patient that has an appointment. Each dental visits the patient has had, it accumulates into a long history of observations, treatment plans, and such. This introduces introductory delays in appointments, which can take up significant time when reviewing ongoing long-term treatments and would require pre-appointment time to review the patient records.

Proposed Solution:

Deploy an MCP-based tool workflow which reads the dental patient records to produce an in-depth summary, with embedded citation to related records, of treatments, procedures, recent scans and tests, and such, for dentists to read. This will automatically curate relevant information from the patient's records to inform dentists on ongoing treatments and understand the "situation" of the patient faster, whilst being consistent and accurate, and would provide links to "direct evidence".

Full-Agentic Workflow Visualization

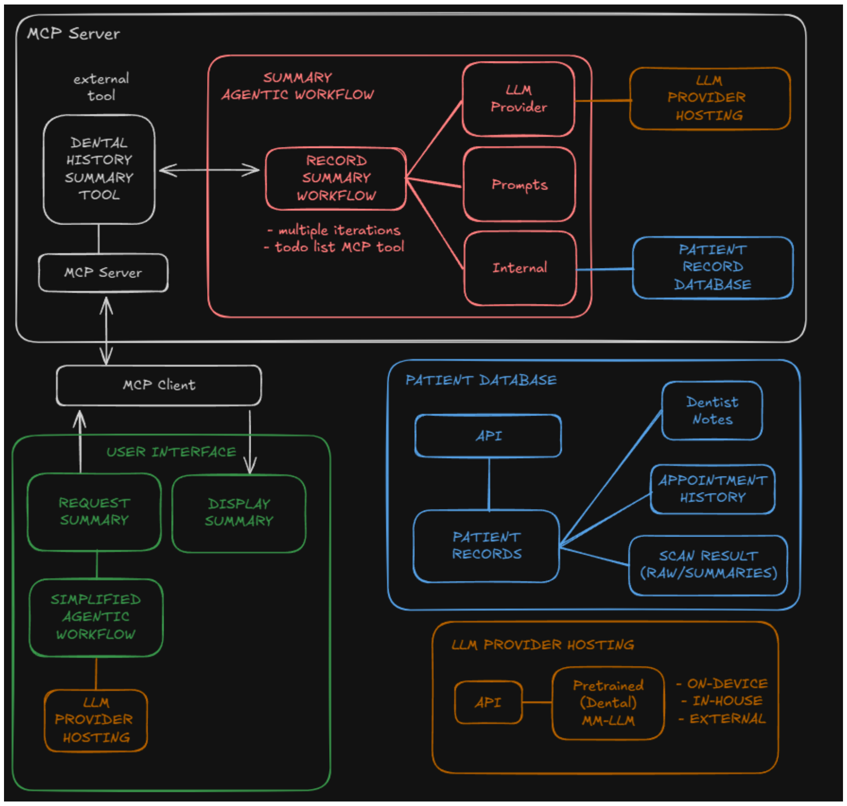

DIAGRAM 1 – FULL AGENTIC MCP SUMMARY WORKFLOW

This diagram demonstrates a full agentic workflow where the User Interface calls the MCP server using agents, and the MCP itself has a tool that uses an agentic workflow to produce the summary. This requires both the user interface device and the MCP server device have access to LLM providers. Compared to a direct API approach, this can be riskier as the LLM attack vector increases to two devices from one.

Direct MCP-Agentic Workflow Visualization

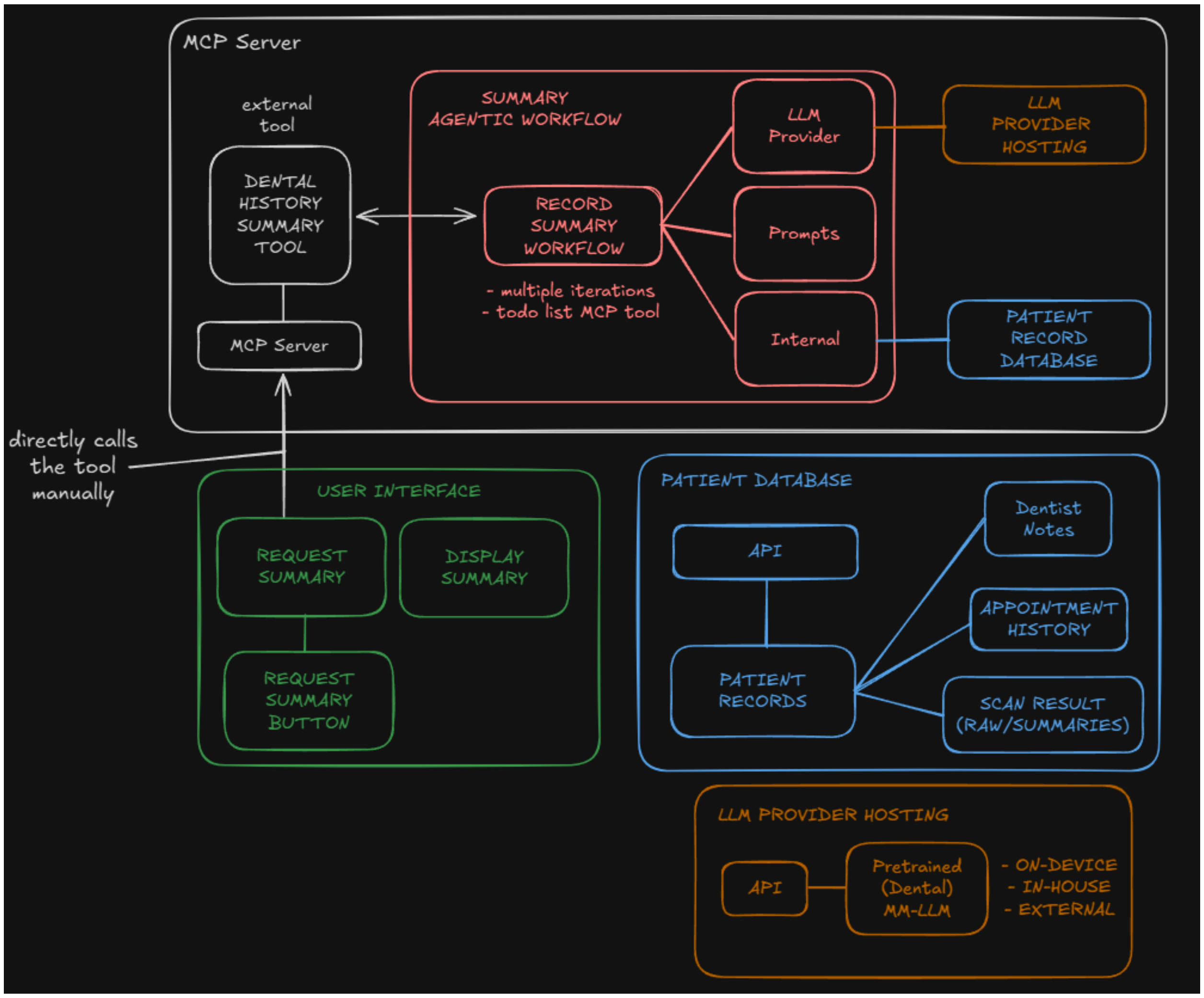

DIAGRAM 2 – DIRECT MCP-AGENTIC SUMMARY WORKFLOW

This diagram demonstrates an alternative workflow where the User Interface calls the MCP server, as if it's an API without the use of agents, and the MCP server uses the agentic workflow to then produce the summary. This reduces the overhead and trims the necessity of having a LLM provider and agentic multi-iteration workflow per-device and only limits it to the server where the summarization occurs. This provides considerably more safety and reduces likelihood for prompt injection as well, as the attack vector is only on the MCP server.

Tools and Services for Tool Implementation

Component	**Tools/Service Options**
Patient Record Database (PRD)	**Database Backends:** - MySQL - PostgreSQL - Microsoft SQL Server - MongoDB - CouchDB
PRD Application Programming Interface (API)	**API Backends:** - FastAPI (Python) - Flask (Python) - Custom by Platform
Summarization	**Per-Device:** - LLM Studio - Ollama - Transformers Package - Torch package **On-Site:** - LLM Studio - Ollama - Custom API (API Backends) **External Providers:** - **OpenAI (request specialized privacy API)** - **Grok (request specialized privacy API)** **Models:** - Custom Trained (dental-focused)
Storing Summarization	**Location:** - Within (global) patients record databases under another table. - Unique (global) database for cache purposes and the ability to delete them all at once. - Per-device with expiry. - Per-site with expiry.