## **1. System Overview**

The system will consist of the following components:

1. **User Interface (Chat System)**: Allows users to input natural language queries.
2. **NLP Layer (LLM Integration)**: Converts user queries into structured JSON queries.
3. **Database Interaction Layer (Dexie.js)**: Executes the structured queries on the database.
4. **Post-Processing Layer**: Summarizes or processes query results using the LLM for user-friendly output.

## **2. Component Design**

### **2.1. Database Schema**

The database uses **Dexie.js** and contains the following tables:

#### **Table: results**

Stores the latest results for inputs processed with prompts.

| **Field** | **Type** | **Description** |
| --- | --- | --- |
| id | String | Unique ID: combination of row\_column. |
| row | String | Input row identifier. |
| column | String | Prompt column identifier. |
| input | String | Input text. |
| prompt | String | Prompt text. |
| result | String | Generated result from processing. |
| version | Number | Version number. |
| updatedAt | Number | Timestamp of last update. |

#### **Table: result\_history**

Maintains historical versions of cell results.

| **Field** | **Type** | **Description** |
| --- | --- | --- |
| historyId | String | Unique history ID (row\_column\_v#). |
| resultId | String | Reference to the result ID. |
| version | Number | Version number. |
| input | String | Input text. |
| prompt | String | Prompt text. |
| result | String | Result at that version. |
| updatedAt | Number | Timestamp of the version. |

#### **Table: transaction\_log**

Logs all operations for auditing purposes.

| **Field** | **Type** | **Description** |
| --- | --- | --- |
| transactionId | String | Unique ID for the transaction. |
| operation | String | INSERT, UPDATE, DELETE. |
| table | String | Table name affected. |
| row | String | Row ID. |
| column | String | Column ID. |
| oldValue | Object | Previous value. |
| newValue | Object | New value. |
| timestamp | Number | Time of operation. |

### **2.2. Workflow**

1. **User Query Input**:
   * Example: *"What is the focus of the database content?"*
2. **Query Processing via LLM**:
   * Convert the natural language query into a structured JSON query.
3. **Query Execution via Dexie.js**:
   * Execute the Dexie query to fetch, aggregate, or analyze the data.
4. **Post-Processing via LLM**:
   * Analyze and summarize the query results.
5. **Return Results**:
   * User receives a natural language response.

## **3. Code Implementation**

### **3.1. Initialize the Dexie Database**

javascript

Copy code

import Dexie from "dexie";

// Initialize the database

const db = new Dexie("SpreadsheetDB");

db.version(1).stores({

results: "id, row, column, version, updatedAt",

result\_history: "historyId, resultId, version, updatedAt",

transaction\_log: "transactionId, timestamp, table, operation"

});

### **3.2. LLM Query Handler**

**Function to send queries to an LLM (OpenAI example):**

javascript

Copy code

async function sendToLLM(prompt, data = null) {

const response = await fetch("https://api.openai.com/v1/chat/completions", {

method: "POST",

headers: {

"Content-Type": "application/json",

Authorization: `Bearer YOUR\_API\_KEY`

},

body: JSON.stringify({

model: "gpt-4",

messages: [

{ role: "system", content: prompt },

{ role: "user", content: data ? JSON.stringify(data) : "" }

],

temperature: 0

})

});

const result = await response.json();

return result.choices[0].message.content;

}

### **3.3. NLP Query Processing**

Convert natural language queries into structured Dexie-compatible queries.

**Example Workflow:**

javascript

Copy code

async function processNaturalQuery(userQuery) {

// Step 1: Ask LLM to parse the query

const structuredQuery = await sendToLLM(

"Convert the user's natural language query into a structured JSON for database queries.",

userQuery

);

const parsedQuery = JSON.parse(structuredQuery);

// Step 2: Execute the parsed query

const results = await executeQuery(parsedQuery);

// Step 3: Summarize the results using the LLM

const summary = await sendToLLM(

"Summarize the following query results for the user:",

results

);

return summary;

}

### **3.4. Query Execution**

Execute the structured query on the Dexie database.

javascript

Copy code

async function executeQuery(query) {

const { action, table, conditions } = query;

if (action === "query") {

let dexieQuery = db[table];

for (const field in conditions) {

dexieQuery = dexieQuery.where(field).equals(conditions[field]);

}

return await dexieQuery.toArray();

}

if (action === "analyze\_focus") {

const allResults = await db.results.toArray();

return allResults;

}

throw new Error("Unsupported query action.");

}

### **3.5. Example Strategic Query Execution**

**User Input**:  
*"Tell me what the focus of the database content is."*

**Workflow**:

1. Send the query to the LLM.
2. Extract all data from the results table.
3. Use the LLM to analyze and summarize the focus.

**Implementation**:

javascript

Copy code

async function analyzeDatabaseFocus() {

const allResults = await db.results.toArray();

const focusSummary = await sendToLLM(

"Analyze the following data and summarize the focus or main themes.",

allResults

);

console.log("Focus Summary:", focusSummary);

return focusSummary;

}

### **3.6. UI for User Interaction**

html

Copy code

<div>

<input id="userQuery" type="text" placeholder="Ask about the database..." />

<button onclick="runStrategicQuery()">Submit</button>

<div id="response"></div>

</div>

<script>

async function runStrategicQuery() {

const query = document.getElementById("userQuery").value;

const response = await processNaturalQuery(query);

document.getElementById("response").innerText = response;

}

</script>

## **4. Summary of Strategic Query Support**

| **Query** | **Dexie Task** | **LLM Task** |
| --- | --- | --- |
| What is the focus of the database? | Fetch all rows from results. | Summarize recurring patterns. |
| What rows are most updated? | Analyze transaction\_log for counts. | Summarize most frequently updated. |
| Compare versions of a cell | Fetch versions from result\_history. | Highlight differences between them. |
| Summarize results for row A1 | Fetch rows with row: "A1". | Summarize results for the user. |

This detailed specification includes all components, schema design, and code examples to build a **conversational interface** for your Dexie.js-powered spreadsheet database with **strategic query support**. Let me know if you'd like further assistance or refinements!