**Backend — Lang Portal**

**Difficulty:** Level 200

**Business** **Goal:**

A language learning school wants to build a prototype of learning portal which will act as three things:

* Inventory of possible vocabulary that can be learned
* Act as a record store, providing correct and wrong score on practice vocabulary
* A unified launchpad to launch different learning apps

You have been tasked with creating the backend API of the application.

**Technical Restrictions:**

* Use SQLite3 as the database
* You can use any language or framework
* Does not require authentication/authorization, assume there is a single user

Leverage AI-coding assistants to write your backend code:

* Cursor
* Windsurf Codeium
* Github Copilot
* Amazon Q Developer
* Google Code Assist

Technical Specification

**Routes**

* **GET** /words - Get paginated list of words with review statistics
* **GET** /groups - Get paginated list of word groups with word counts
* **GET** /groups/:id - Get words from a specific group (This is intended to be used by target apps)
* **POST** /study\_sessions - Create a new study session for a group
* **POST** /study\_sessions/:id/review - Log a review attempt for a word during a study session

**GET /words**

* page: Page number (default: 1)
* sort\_by: Sort field ('kanji', 'romaji', 'english', 'correct\_count', 'wrong\_count') (default: 'kanji')
* order: Sort order ('asc' or 'desc') (default: 'asc')

**GET /groups/:id**

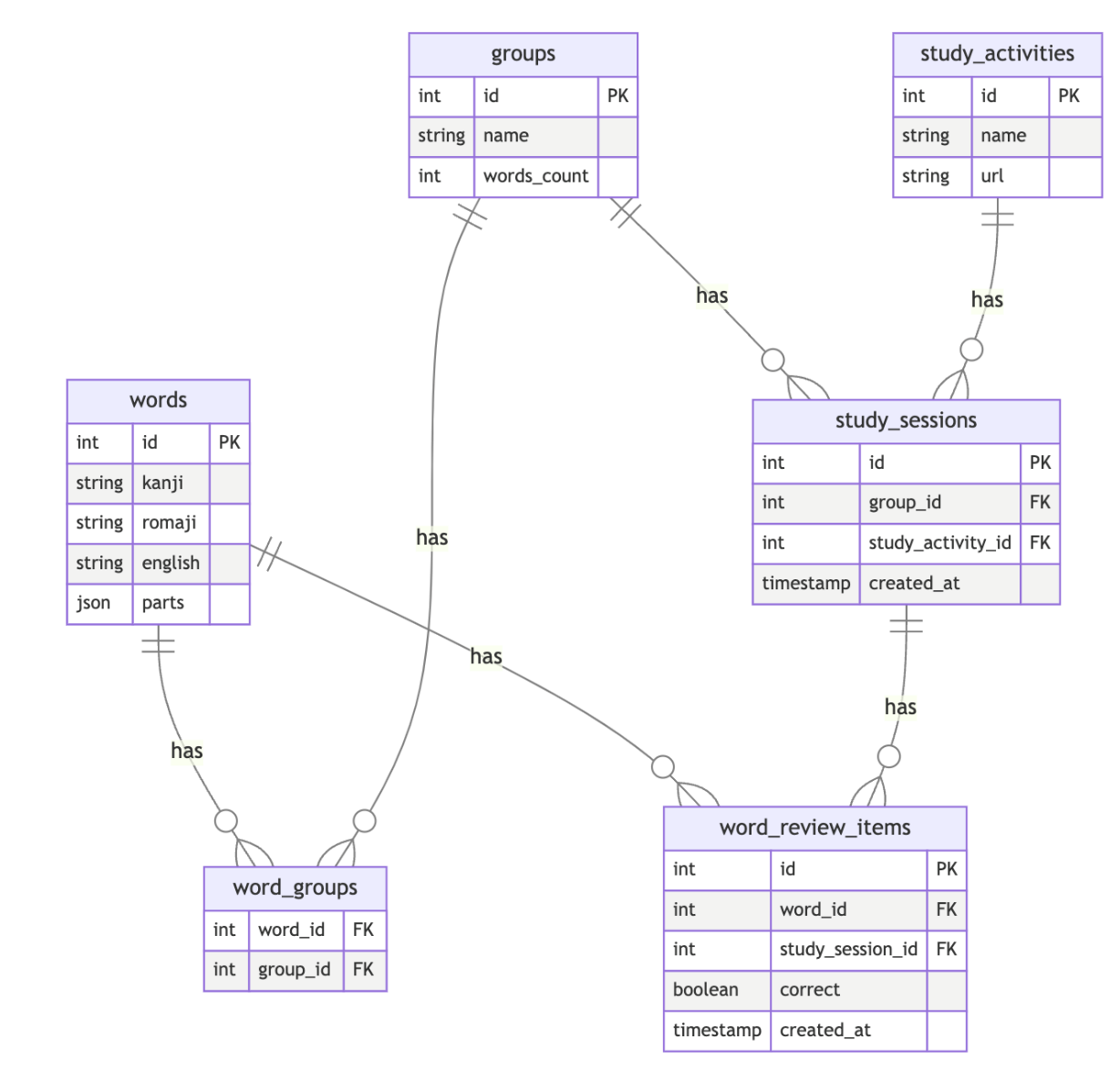
* page: Page number (default: 1)
* sort\_by: Sort field ('name', 'words\_count') (default: 'name')
* order: Sort order ('asc' or 'desc') (default: 'asc')

**POST /study\_sessions**

* group\_id: ID of the group to study (required)
* study\_activity\_id: ID of the study activity (required)

**POST /study\_sessions/:id/review**

**Database Schema**

****

**words —** Stores individual Japanese vocabulary words.

- `id` (Primary Key): Unique identifier for each word

- `kanji` (String, Required): The word written in Japanese kanji

- `romaji` (String, Required): Romanized version of the word

- `english` (String, Required): English translation of the word

- `parts` (JSON, Required): Word components stored in JSON format

**groups —** Manages collections of words.

- `id` (Primary Key): Unique identifier for each group

- `name` (String, Required): Name of the group

- `words\_count` (Integer, Default: 0): Counter cache for the number of words in the group

**word\_groups —** join-table enabling many-to-many relationship between words and groups.

- `word\_id` (Foreign Key): References words.id

- `group\_id` (Foreign Key): References groups.id

**study\_activities —** Defines different types of study activities available.

- `id` (Primary Key): Unique identifier for each activity

- `name` (String, Required): Name of the activity (e.g., "Flashcards", "Quiz")

- `url` (String, Required): The full URL of the study activity

**study\_sessions** — Records individual study sessions.

- `id` (Primary Key): Unique identifier for each session

- `group\_id` (Foreign Key): References groups.id

- `study\_activity\_id` (Foreign Key): References study\_activities.id

- `created\_at` (Timestamp, Default: Current Time): When the session was created

**word\_review\_items** — Tracks individual word reviews within study sessions.

- `id` (Primary Key): Unique identifier for each review

- `word\_id` (Foreign Key): References words.id

- `study\_session\_id` (Foreign Key): References study\_sessions.id

- `correct` (Boolean, Required): Whether the answer was correct

- `created\_at` (Timestamp, Default: Current Time): When the review occurred

**Relationships**

* word belongs to groups through  word\_groups
* group belongs to words through word\_groups
* session belongs to a group
* session belongs to a study\_activity
* session has many word\_review\_items
* word\_review\_item belongs to a study\_session
* word\_review\_item belongs to a word

**Design Notes**

* All tables use auto-incrementing primary keys
* Timestamps are automatically set on creation where applicable
* Foreign key constraints maintain referential integrity
* JSON storage for word parts allows flexible component storage
* Counter cache on groups.words\_count optimizes word counting queries