Build Web App with LLM

Tan Phat Nguyen

Contents

Techstacks

- Nuxt
    - Full-stack framework built on VueJS
- SQLite
    - Lightweight and fast
    - Vector support with extension
- Ollama
    - Model management
    - Easy to use

Ollama CLI

- Download CLI here
- Find a model here

```
ollama pull llama3.2:3b

ollama show llama3.2:3b

ollama run llama3.2:3b
```

- Served at http://localhost:11434

```
curl http://localhost:11434/api/generate -d '{
  "model": "llama3.2",
  "prompt": "Explain LLM in 3 sentences.',
  "stream": false
}'
```

Ollama-JS

- `ollama.generate`

```
import ollama from 'ollama/browser'

const response = await ollama.generate({
  model: 'llama3.2:3b',
  prompt: 'Explain LLM in 3 sentences.',
  stream: false
})

console.log(response.response)
```

```
TypeError: Failed to fetch
```

Ollama-JS

- `ollama.generate` with `steam = true`

```
import ollama from 'ollama/browser'

let output = ''
const response = await ollama.generate({
  model: 'llama3.2:3b',
  prompt: 'Explain LLM in 3 sentences.',
  stream: true
})

for await (const r of response) {
  output += r.response
  console.clear()
  console.log(output)
}
```

```
TypeError: Failed to fetch
```

Ollama-JS

- `ollama.chat`

```
import ollama from 'ollama/browser'

const response = await ollama.chat({
  model: 'llama3.2:3b',
  messages: [
    { role: 'user', content: 'Explain LLM in 3 sentences.' }
  ],
  stream: false
})

console.log(response.message.content)
```

```
TypeError: Failed to fetch
```

Ollama-JS

- `ollama.chat`

```
import ollama from 'ollama/browser'

const response = await ollama.chat({
  model: 'llama3.2:3b',
  messages: [
    { role: 'system', content: `You explain like I'm five years old.` },
    { role: 'user', content: 'Explain LLM in 3 sentences.' }
  ],
})

console.log(response.message)
```

```
TypeError: Failed to fetch
```

Integration in App

- Generate card's definition

```
async function generateDef(c: Card) {
  c.def = await $generate(`"${c.term}": Provide a short, plain text definition without any redundant information.`)
}
```

- `$generate` function

```
async function $generate(promt: string) {
  const response = await ollama.generate({
    model: 'llama3.2:3b',
    prompt,
    stream: false
  })

  return response.response
}
```

Integration in App

- Generate cards

```
const response = await $generate(
  `Using the reference cards: ${set.value.cards}, generate new, meaningful cards in JSON format using following
  JSON schema:
{
  "cards": [
    { "term": "string", "def": "string" }
  ]
}`
)
```

Validation with `zod`

```
import { z } from 'zod'

const schema = z.object({
  cards: z.object({
    term: z.string().describe('This is term'),
    def: z.string().describe('This is def')
  }).array()
})

const output = `{
  "cards": [
    { "term": "Capital of Germany", "def": "Berlin" }
  ]
}`

console.log(schema.parse(JSON.parse(output)))
```

```
{
  "cards": [
    {
      "term": "Capital of Germany",
      "def": "Berlin"
    }
  ]
}
```

Create JSON Schema with `zod-to-json-schema`

```
import { z } from 'zod'
import { zodToJsonSchema } from 'zod-to-json-schema'

const schema = z.object({
  cards: z.object({
    term: z.string().describe('This is term'),
    def: z.string().describe('This is def')
  }).array()
})

console.log(zodToJsonSchema(schema))
```

```
{
  "type": "object",
  "properties": {
    "cards": {
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "term": {
            "type": "string",
            "description": "This is term"
          },
          "def": {
            "type": "string",
            "description": "This is def"
          }
        },
        "required": [
          "term",
          "def"
        ],
        "additionalProperties": false
      }
    }
  },
  "required": [
    "cards"
  ],
  "additionalProperties": false,
  "$schema": "http://json-schema.org/draft-07/schema#"
}
```

Upgrade `$generate` function

```
async function $generate(promt: string) {
  const response = await ollama.generate({
    // ...
  })

  return response.response
}
```

Demo

SQLite with Vector

- using `sqlite-vec` extension

```
create virtual table "sets" using vec0 (
  id integer primary key autoincrement,
  +title text,
  +cards text,
  +tags text,
  embedding float[768],
  +createAt text,
);
```

```
  -- Auxiliary column: unindexed, fast lookups
  +title text,

  -- Vector text embedding with 768 dimensions
  embedding float[768],
```

- selecting most matched set

```
select
      id,
      title,
      cards,
      tags,
      createAt,
      vec_distance_cosine(embedding, ?) as distance
from sets
order by distance;
```

$$\begin{aligned}\text{Cosine Distance} &= 1 - \cos(\theta) \\ &= 1 - \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|}\end{aligned}$$

Embeddings

```
import ollama from 'ollama/browser'

const response = await ollama.embed({
  model: 'nomic-embed-text',
  input: 'Hello from the other side.',
})

console.log('Length:', response.embeddings[0].length)
console.log('First 10:', response.embeddings[0].slice(0, 10))
```

```
TypeError: Failed to fetch
```

```
// Before inserting new set into DB
set.embedding = await $embed(JSON.stringify({
  title: set.title,
  cards: set.cards,
  tags: set.tags,
}))
```

Demo

Chat with LLM

- Basic

```
const messages = [
  { role: 'system', content: `You are an assistant for my studies.` }
]
```

Chat with LLM

- Upload file

```
type Message = {
  role: 'system' | 'assistant' | 'user',
  content: string
}
```

```
messages.value.push({
  role: 'system',
  content: 'You are a helpful assistant knowledgeable about the following document:',
  type: 'file',
  name: file.name,
})

for (const c of $chunk(content.toString(), { max: 2048 })) {
  messages.value.push({
    role: 'system',
    content: c,
    type: 'hidden',
  })
}
```

Demo

Our Journey

1. Find Technology Stack (Nuxt, SQLite, Ollama)
2. Experiment with Ollama (CLI, JS library)
3. Integrate Ollama into the Application (Structure Output)
4. Develop a Simple Use Case with Embeddings (Search for Sets)
5. Build Chat Functionality (Basic, File Upload)
6. Add Summary Functionality (Based on Chat Features)

Thank you

https://chubetho.github.io/llm_slides

https://github.com/chubetho/LLM

Powered by Slidev