

In `code/shadp`

I was trying out Shadcn, vercel/postgres, Typescript, Tailwind

## Main learning points

- \* Fetching and posting (Typescript and data.ts server actions)
- \* User input check (zod)
- \* Pure postgres queries (vercel/postgres)
- \* fast styling (Tailwind)

## Fetching and posting data (Server Actions way)

Expect this type

```
export interface VocabItem {
  vocab_id: number;
  vocab_image_url?: string;
  vocab_word: string;
  vocab_definition: string;
  vocab_context?: string;
  vocab_example?: string;
  vocab_created: Date;
}
```

```
export async function fetchAllVocabInstantFeedback() {
  try {
    const data = await sql`VocabItem`>`SELECT * FROM vocabularies`
    return data.rows;
  } catch (error) {
    console.error("Database Error:", error);
    throw new Error(
      "Failed to fetch vocabulary data for the instant feedback"
    );
  }
}
```

```
export async function insertVocabItem(newItem: NewVocabItem) {
  try {
    await sql`
      INSERT INTO vocabularycards
        (vocab_image_url, vocab_word, vocab_definition, vocab_color)
      VALUES (${newItem.vocab_image_url}, ${newItem.vocab_word},
        ${newItem.vocab_definition}, ${newItem.vocab_color});
    `;
    return true; // Return true if the insert was successful
  } catch (error) {
    console.error("Database Error:", error);
    return false; // Return true if the insert was successful
  }
}
```

New VocabItem  
complies with this  
db table

- \* Server actions abstract  
get, post

- \* You'd use get, post in API router. But this way you can use types and a single file for all.

Notice:

The incoming data is **missing vocab-id, vocab-created**. Those are set on the db upon creation!  
so no need to have them

```
CREATE TABLE VocabularyCards (
  vocab_id SERIAL PRIMARY KEY,
  vocab_image_url VARCHAR(255),
  vocab_word VARCHAR(100) NOT NULL,
  vocab_definition TEXT NOT NULL,
  vocab_context TEXT,
  vocab_example TEXT,
  vocab_created TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
);
```

```
interface NewVocabItem {
  vocab_image_url?: string;
  vocab_word: string;
  vocab_definition: string;
  vocab_context?: string;
  vocab_example?: string;
}
```

In Page.tlx:

## Important!!

```
import { unstable_noStore } from "next/cache";
unstable_noStore();
```

↳ For not caching! (otherwise refreshing won't show new items)

```
export default async function Page() {  
  //const words = await fetchAllVocabInstantFeedback();  
  const words = await fetchAllVocabInstantFeedback();  
}
```

```
<div className="grid auto-rows-max grid-col"
  {words.map((wordItem, i) => (
    <VocabCard key={i} word={wordItem} />
  ))}
/>
```

```
const foreSchema = z.object({
  image_url: z.union([z.literal(""), z.string().trim().url()]),
  word: z
    .string()
    .min(1, { message: "Word must have at least 1 character." }),
  definition: z
    .string()
    .min(2, { message: "Definition must have at least 2 characters." })
    .describe("A passage where you found this word."),
  example: z
    .string()
    .describe("Try to make an example usage of the word yourself."),
});
```

## form validation (20d)

based on this  
New object type

```
interface NewVocabItem {
    vocab_image_url?: string;
    vocab_word: string;
    vocab_definition: string;
    vocab_context?: string;
    vocab_example?: string;
}
```

\* Caching must be turned off in the components where we get new data. Otherwise not even refresh will show new database items! **Ex:**

```
import { unstable_noStore } from "next/cache";  
unstable_noStore();
```

 in Page.jsx

```
import { unstable_noStore } from "next/cache";
unstable_noStore();
```

in Page.tsx