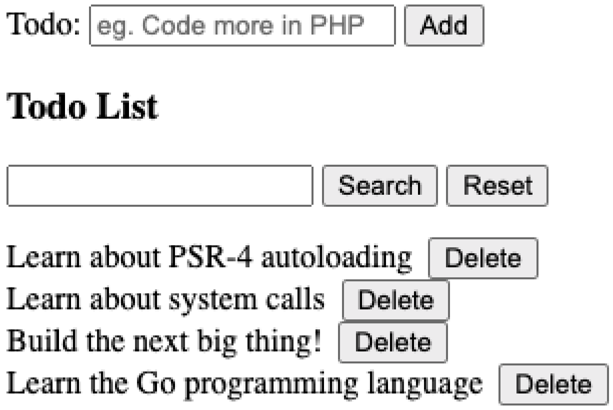
COMP 3015, BCIT Computing
Practice Final Exam Part 2

Problem 1: Searchable CRUD application todo list

Write a script called **index.php** which provides forms to add a “todo”, search existing todos and reset the search (show everything again). Each todo can also be deleted.



Todo List

Learn about PSR-4 autoloading

Learn about system calls

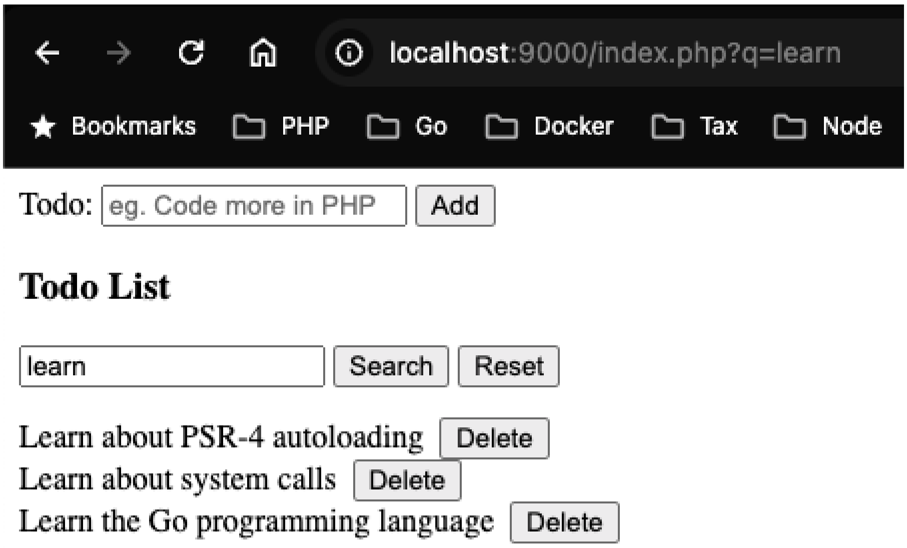
Build the next big thing!

Learn the Go programming language

Fig 1. The todo application with 4 “todo” items added.

When a new todo is added, an HTTP POST request containing the todo data should be made to the server. The implementation of deletion should also make a POST request to the server.

When a term is searched, an HTTP GET request should be made to the server. The list should be filtered to only contain matching items:



localhost:9000/index.php?q=learn

Todo List

learn

Learn about PSR-4 autoloading

Learn about system calls

Learn the Go programming language

Fig 2. The todo application with a search term of “learn”.

As this is not a SQL course, here’s a hint on the search. You can use the MySQL “LIKE” operator in order to implement a simple search:

```
$dsn = "mysql:host=localhost;dbname=c3015_final";
$pdo = new PDO($dsn, "usernameHere", "passwordHere");
…
$pdoStatement = $pdo->prepare("SELECT * FROM todos WHERE title LIKE ?");
$pdoStatement->execute(["%$query%"]);
```

When the “reset” button is clicked, the filter should be removed and all todo items shown again.

The SQL commands to create the database are as follows:

```
CREATE DATABASE c3015_final;
USE c3015_final;
CREATE TABLE todos (
	id INT UNSIGNED auto_increment,
	title VARCHAR(255),
	PRIMARY KEY(id)
);
```

Problem 2: Letter Counting

Given a text file of the book “Alice’s Adventures in Wonderland”, count the number of occurrences of each letter in the book (case-insensitive). Generate an associative array containing data in a **letter => occurrences** format.

For example, if instead of the text of Alice’s Adventures in Wonderland, the input file contained “BCIT zzzzz” the program would produce:

```
[
   “z” => 5,
   “b” => 1,
   “c” => 1,
   “i” => 1,
   “t” => 1,
]
```

Note that the order of the letters in the structure does not matter.