Koneksi ke Database

```php
// Koneksi ke database (sesuaikan dengan informasi koneksi database Anda)
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "projectai";

$conn = new mysqli($servername, $username, $password, $dbname);
```

1. Baris 1-4: Menginisialisasi variabel dengan informasi koneksi database.
2. Baris 6: Membuat koneksi ke database menggunakan `mysqli`.

```php
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
```

3. Baris 8-10: Mengecek apakah koneksi berhasil. Jika gagal, menampilkan pesan error dan menghentikan eksekusi skrip.

Mengambil Data dari POST Request

```php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $start = isset($_POST['start']) ? $_POST['start'] : '';
    $end = isset($_POST['end']) ? $_POST['end'] : '';
```

4. Baris 12: Mengecek apakah metode request adalah POST.
5. Baris 13-14: Mengambil data `start` dan `end` dari form POST. Jika tidak ada, menginisialisasi dengan string kosong.

Mengecek Validitas Input

```php
if (!empty($start) && !empty($end)) {
    // Implementasi Best First Search
    $paths = findShortestPaths($start, $end, $conn);
    if ($paths) {
        echo "Rute Jalur Dari $start Ke $end:<br>";
        foreach ($paths as $path) {
            if ($path['cost'] > 0) { // Hanya tampilkan jalur dengan total biaya lebih
                echo "- " . $start . " -> " . implode(" -> ", $path['path']) . " denga
            }
        }
    } else {
        echo "There is no path from $start to $end.";
    }
} else {
    echo "Please select start and end locations.";
}
```

6. Baris 16: Mengecek apakah input `start` dan `end` tidak kosong.
7. Baris 17-31: Jika valid, panggil fungsi `findShortestPaths` untuk mencari jalur terpendek dari `start` ke `end`.
8. Baris 18: Memeriksa apakah ada jalur yang ditemukan.
9. Baris 19-26: Menampilkan jalur yang ditemukan, jika ada, dengan biaya total yang lebih dari 0.
10. Baris 27: Menampilkan pesan jika tidak ada jalur yang ditemukan.
11. Baris 29-31: Menampilkan pesan jika input `start` dan `end` kosong.

Menutup Koneksi Database

```php
// Tutup koneksi database setelah selesai digunakan
$conn->close();
```

12. Baris 34: Menutup koneksi database.

Fungsi Best First Search

```php
// Fungsi untuk mencari jalur terpendek menggunakan Best First Search
function findShortestPaths($start, $end, $conn) {
    $queue = new SplPriorityQueue();
    $queue->insert([$start, [], 0], 0); // Menyimpan path dan total cost
    $visited = [];

    $shortestPaths = []; // Array untuk menyimpan semua jalur terpendek
```

13. Baris 38: Mendefinisikan fungsi `findShortestPaths`.
14. Baris 39: Membuat antrian prioritas.
15. Baris 40: Memasukkan node awal ke antrian prioritas dengan path kosong dan cost 0.
16. Baris 41: Menginisialisasi array `visited` untuk menyimpan node yang sudah dikunjungi.
17. Baris 43: Menginisialisasi array `shortestPaths` untuk menyimpan jalur-jalur terpendek.

```php
    while (!$queue->isEmpty()) {
        [$current, $path, $cost] = $queue->extract();
        if ($current == $end) {
            $shortestPaths[] = ['path' => $path, 'cost' => $cost]; // Menyimpan jalur terp
            continue; // Lanjut ke iterasi berikutnya untuk mencari jalur lainnya
        }

        if (!isset($visited[$current])) {
            $visited[$current] = true;

            // Ambil tetangga-tetangga dari current node dari database
            $sql = "SELECT node2, weight FROM connections_bidirectional WHERE node1='$curr
            $result = $conn->query($sql);
            if ($result->num_rows > 0) {
                while ($row = $result->fetch_assoc()) {
                    $neighbor = $row["node2"];
                    $neighborCost = $row["weight"];
                    if (!isset($visited[$neighbor])) {
                        $queue->insert([$neighbor, array_merge($path, [$neighbor]), $cost
                    }
                }
            }
        }
    }

    return $shortestPaths; // Mengembalikan array dari jalur-jalur terpendek
}
```

18. Baris 45: Melakukan iterasi selama antrian tidak kosong.
19. Baris 46: Mengambil elemen dengan prioritas tertinggi dari antrian.
20. Baris 47-50: Jika node saat ini adalah tujuan (`end`), menyimpan jalur dan biaya, lalu melanjutkan iterasi untuk mencari jalur lainnya.
21. Baris 52: Mengecek apakah node saat ini sudah dikunjungi.
22. Baris 53: Menandai node saat ini sebagai dikunjungi.
23. Baris 55-66: Mengambil tetangga-tetangga node saat ini dari database dan menambahkan ke antrian jika belum dikunjungi.
24. Baris 67: Mengembalikan array `shortestPaths` yang berisi semua jalur terpendek.