```java
package pt.ipp.isep.dei.mdisc.SprintC;

import java.io.IOException;
import java.util.*;

public class US18Handler {
    public static void executeUS18(Scanner scanner) {
        System.out.println("Digite o nome do ficheiro CSV da matriz:");
        String matrixFileName = scanner.nextLine();
        String matrixPath =
"src/main/java/pt/ipp/isep/dei/mdisc/SprintC/input/" + matrixFileName;

        System.out.println("Digite o nome do ficheiro para o CSV dos
pontos:");
        String pointsFileName = scanner.nextLine();
        String pointsPath =
"src/main/java/pt/ipp/isep/dei/mdisc/SprintC/input/" + pointsFileName;

        try {
            int[][] weights = CSVReader.readMatrixFromCSV(matrixPath);
            String[] points = CSVReader.readPointsFromCSV(pointsPath);

            List<Integer> assemblyPoints = new ArrayList<>();
            for (int i = 0; i < points.length; i++) {
                if (points[i].startsWith("AP")) {
                    assemblyPoints.add(i);
                }
            }

            EmergencyPathFinder epf = new EmergencyPathFinder(weights,
points, assemblyPoints);
            List<String> outputPaths = new ArrayList<>();

            for (int i = 0; i < points.length; i++) {
                if (!assemblyPoints.contains(i)) {
                    int closestAPIdx = -1;
                    int minDist = Integer.MAX_VALUE;
                    for (int apIdx : assemblyPoints) {
                        List<Integer> path = epf.dijkstra(i, apIdx);
                        int dist = path.size() - 1; // assuming each step
has uniform cost
                        if (dist < minDist) {
                            minDist = dist;
                            closestAPIdx = apIdx;
                        }
                    }
                    List<Integer> path = epf.dijkstra(i, closestAPIdx);
                    String pathString = formatPath(path, points);
                    String indexPathString = formatIndexPath(path);
                    int cost = calculatePathCost(path, weights);
                    outputPaths.add(indexPathString + "; Custo: " + cost
+ "\n" + pathString + "; Custo total: " + cost);
                }
            }


CSVReader.writePathsToCSV("src/main/java/pt/ipp/isep/dei/mdisc/SprintC/ou
tput/US18/shortest_routes_all_us18.csv", outputPaths);
```

```
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private static String formatPath(List<Integer> path, String[] points)
{
        StringBuilder sb = new StringBuilder();
        for (int index : path) {
            sb.append(points[index]).append(" -> ");
        }
        if (sb.length() > 0) {
            sb.setLength(sb.length() - 4); // remove the last " -> "
        }
        return sb.toString();
    }

    private static String formatIndexPath(List<Integer> path) {
        StringBuilder sb = new StringBuilder();
        for (int index : path) {
            sb.append((index + 1)).append(" -> ");
        }
        if (sb.length() > 0) {
            sb.setLength(sb.length() - 4); // remove the last " -> "
        }
        return sb.toString();
    }

    private static int calculatePathCost(List<Integer> path, int[][]
weights) {
        int cost = 0;
        for (int i = 0; i < path.size() - 1; i++) {
            cost += weights[path.get(i)][path.get(i + 1)];
        }
        return cost;
    }
}
```