Shelf Identification Problem

At Infilect, our artificial intelligence solutions are used to help retailers optimize their shelf space. This involves identifying and analyzing the arrangement of products on the shelf, including determining the shape and size of the area occupied by each brand.

Consider a grocery store that sells four different Britannia products: Britannia Good Day, Britannia Marie Gold, Britannia NutriChoice, and Britannia Milk Bikis. These products are arranged on a shelf in the store, represented by the following two-dimensional layout:

```
[
 ['G', 'G', 'G', 'M', 'M', 'M', 'M'],
 ['G', 'B', 'G', 'M', 'N', 'N', 'M'],
 ['G', 'G', 'G', 'M', 'N', 'N', 'M'],
 ['B', 'B', 'B', 'B', 'B', 'N', 'N']
]
```

In this representation, each sublist corresponds to a row on the shelf, and each element within the sublist represents a product on that row ('G' for Good Day, 'M' for Marie Gold, 'N' for NutriChoice, and 'B' for Milk Bikis).

Your task is to write an API service that takes this shelf layout as input and identifies the shape of each brand. If the area can be identified as a horizontal rectangle, vertical rectangle, square, or irregular polygon, label it as such.

Sample Input & Expected Result

Example 1:

```
[
 ['G', 'M', 'N', 'B'],
 ['G', 'M', 'N', 'B'],
 ['G', 'M', 'N', 'B'],
 ['G', 'M', 'N', 'B']
]
```

Expected output:

```
{
'G':  {'shape':'vertical rectangle','location':['left'],},
'M':  {'shape':'vertical rectangle','location': ['left'],},
'B':  {'shape':'vertical rectangle','location':['right'],},
'N':  {'shape':'vertical rectangle','location': ['right'],}
}
```

Example 2:

```
[
 ['G', 'G', 'M', 'M'],
 ['G', 'G', 'M', 'M'],
 ['B', 'B', 'N', 'N'],
 ['B', 'B', 'N', 'N']
]
```

Expected output:

```
{
'G': {'shape':'square', 'location':['top left'],},
'M': {'shape':'square', 'location':['top right'],},
'B': {'shape':'square', 'location':['bottom left'],},
'N': {'shape':'square', 'location': ['bottom right'],}
}
```

Example 3:

```
[
 ['G', 'G', 'G', 'M', 'M', 'M', 'M'],
 ['G', 'B', 'G', 'M', 'N', 'N', 'M'],
 ['G', 'G', 'G', 'M', 'N', 'N', 'M'],
 ['B', 'B', 'B', 'B', 'B', 'N', 'N']
]
```

Expected output:

```
{
'G': {'shape':'polygon', 'location': ['top left'],}
'M': {
    'shape':'polygon',
    'location': ['right'],
},
'B': {'shape':'horizontal rectangle', 'location':  ['bottom left'],},
'N': {'shape':'polygon', 'location': 'middle right',}
}
```

Assignment Instructions:

1. **Programming Languages**:
    1.1. You may choose to use Python (***version 3.10 or higher***) for this assignment.
2. **API Framework:**
    2.1. You are free to select any API framework of your choice to solve the given problem.
3. **Built-in Algorithms:**
    3.1. Do not use built-in libraries for operations such as sorting or searching. You are expected to implement these functionalities from scratch.
4. **Dockerization:**
    4.1. Your application must be containerized using Docker. This means that it should run inside a Docker container. Ensure that a Dockerfile is included in your submission.
5. **Documentation:**
    5.1. Please include a comprehensive README file. This should detail:
        5.1.1. How to set up and run your solution.
        5.1.2. A complete implementation guide.
        5.1.3. Test cases and how to execute them.
6. **Submission Deadline:**
    6.1. Package all your code, Dockerfile, README, and any other necessary files into a .zip archive.
    6.2. Email the zipped file to deepa@infilect.com email address within 2 days after receiving this assignment.