Train Ticket Machine

You are asked to write a small user interface of a train ticket machine.
These machines have a direct but unreliable connection to the central system and use a touchscreen display which works as follows.
As the user types each character of the station's name on the touchscreen, the display should:

1. Update to show all valid choices for the next character
2. List of possible matching stations.

The illustration below shows what is needed when 'D A R T' has been entered.

```
User input: D A R T __
```

A	B	C	D	E		DARTFORD
F	G	H	I	J		DARTON
K	L	M	N	**O**		
P	Q	R	S	T		
U	V	W	X	Y		
Z						

This URI simulates the central system response: https://raw.githubusercontent.com/abax-as/coding-challenge/master/station_codes.json

Requirements:

1. Typing a search string will show:
    1. All stations that start with the search string.
    2. All valid next characters for each matched station.
2. Space is a valid character when returning a list of next characters.
3. The user can select a station from the list of stations found at any time.
4. The selected station will be used further for routing and pricing purposes (you don't need to build it, but give an indication).

Operational requirements:

1. Runtime speed is very important, loading time is not.
2. Make no assumptions about the data source in real life.
3. In some cases filling in the station name may be cumbersome for the user, there should be a list of recent searches stored, easily available for later use.
4. Prepare your page for longer loading times and errors that can be received.
5. To demonstrate your experience with WebComponents, showcase your ability to work with tools like i.e. https://storybook.js.org/ and document your work.
6. We would like to see your approach to observability, thefore using tools to trace users behaviour is required. Use the solution you prefer and justify your choices.
7. If you have any doubts regarding your UX/UI choices or would like to test your ideas, we would be thrilled to see A/B testing scenarios implemented with using https://www.statsig.com. The specific scenario matters less than efectivelly incorporating the tool.

Expected Scenarios:

- **Given** a list of stations 'DARTFORD', 'DARTON', 'TOWER HILL', 'DERBY'
  - **When** input 'DART'
  - **Then** should return:
    1. The characters of 'F', 'O'
    2. The stations 'DARTFORD', 'DARTON'.

- **Given** a list of stations 'LIVERPOOL', 'LIVERPOOL LIME STREET', 'PADDINGTON'
  - **When** input 'LIVERPOOL'
  - **Then** should return:
    1. The characters of ' '
    2. The stations 'LIVERPOOL', 'LIVERPOOL LIME STREET'

- **Given** a list of stations 'EUSTON', 'LONDON BRIDGE', 'VICTORIA'
  - **When** input 'KINGS CROSS'
  - **Then** should return:
    1. no next characters
    2. no stations

Evaluation Guidelines:

1. **Understanding and interpretation of the domain**
   - Context
   - Boundaries
   - Ubiquitous Language

2. **Delivery quality**
    - Complete solution meeting all requirements
    - No typographical errors

3. **Code readability**
    - Variables and naming
    - Consistent code formatting
    - Adequate documentation

4. **Code quality**
    - Coding against tests
    - Code coverage & complexity
    - Correct usage of data structures and techniques
    - The right level of abstraction

5. **Solution quality**
    - Structure and organization
    - Separation of concerns

6. **Bonus Points**
    - Patterns & Practises
    - Production readiness
    - TypeScript
    - VueJS
    - Docker