Medopad - Software Engineer assignment

Description
You are given a board with dimensions 4x5 (width x height) containing multiple pieces. There are 10 pieces of varying dimensions (width and height):
• 1x2 (4 pieces)
• 2x2 (1 piece)
• 2x1 (1 piece)
• 1x1 (4 pieces)

The following is a visualisation of the pieces with the board in the initial state :

```
XXXXXX
XabbcX
XabbcX
XdeefX
XdghfX
Xi  jX
XXZZXX
```

The pieces described above in the visualisation:
• 1x2 (4 pieces: a, d, c, f)
• 2x2 (1 piece: b)
• 2x1 (1 piece: e)
• 1x1 (4 pieces: g, h, i, j)

X is a border, and Z is a gap through which the main piece b can move.

To solve the puzzle, you must move the pieces around the board and move b out of the exit Z.

The rules are:
• A piece may only move vertically or horizontally into empty space next to it. (example moves are illustrated below)
• If there are 2 empty spaces in given direction, the given piece may move 1 or 2 spaces (counting as 1 move)
• Pieces may not occupy the space Z.
• The piece b may move through Z solving the problem.
• A piece can not overlap with another piece.
• There are always 2 empty spaces on the board.

Objective:
Find the solution to the puzzle moving pieces around the board with b eventually moving through Z in as few moves as possible (not just the first solution).

Once solved, show the moves from initial state to the final state in the optimal solution.

Things to consider:

- The internet is your oyster.
- Ideally, your implementation should be usable as part of a bigger project by people unfamiliar with it.
- You may also consider how your system can support different UIs (and, if you want, you can add a GUI for inputting different boards)
- Lastly, if you want, you may expand the solution to provide additional capabilities

Deliverables:

- Choose any language you want within reason.
- Source code
- Any additional project files to allow us to build and test your solution (if relevant)
- Any known issues, restrictions or assumptions made

Example Moves (each is 1 move)

1) `i` **moves right 1 block**

```
XXXXXX      XXXXXX
XabbcX      XabbcX
XabbcX      XabbcX
XdeefX  ->  XdeefX
XdghfX      XdghfX
Xi  jX      X i jX
XXZZXX      XXZZXX
```

2) `i` **moves right 2 block**

```
XXXXXX      XXXXXX
XabbcX      XabbcX
XabbcX      XabbcX
XdeefX  ->  XdeefX
XdghfX      XdghfX
Xi  jX      X  ijX
XXZZXX      XXZZXX
```

3) `d` **moves down 1 block**

```
XXXXXX      XXXXXX
XabbcX      XabbcX
XabbcX      XabbcX
Xdeef X  ->  X eefX
Xdghf X      Xdghf X
X  ijX      Xd ijX
XXZZXX      XXZZXX
```

4) illustration where `b` **could move out through** `Z`

```
XXXXXX      XXXXXX
XaeecX      XaeecX
XaghcX      XaghcX
Xdbbf X  ->  Xd  f X
Xdbbf X      Xd  f X
Xi  jX      Xi  jX
XXZZXX      XXZZXX
```