

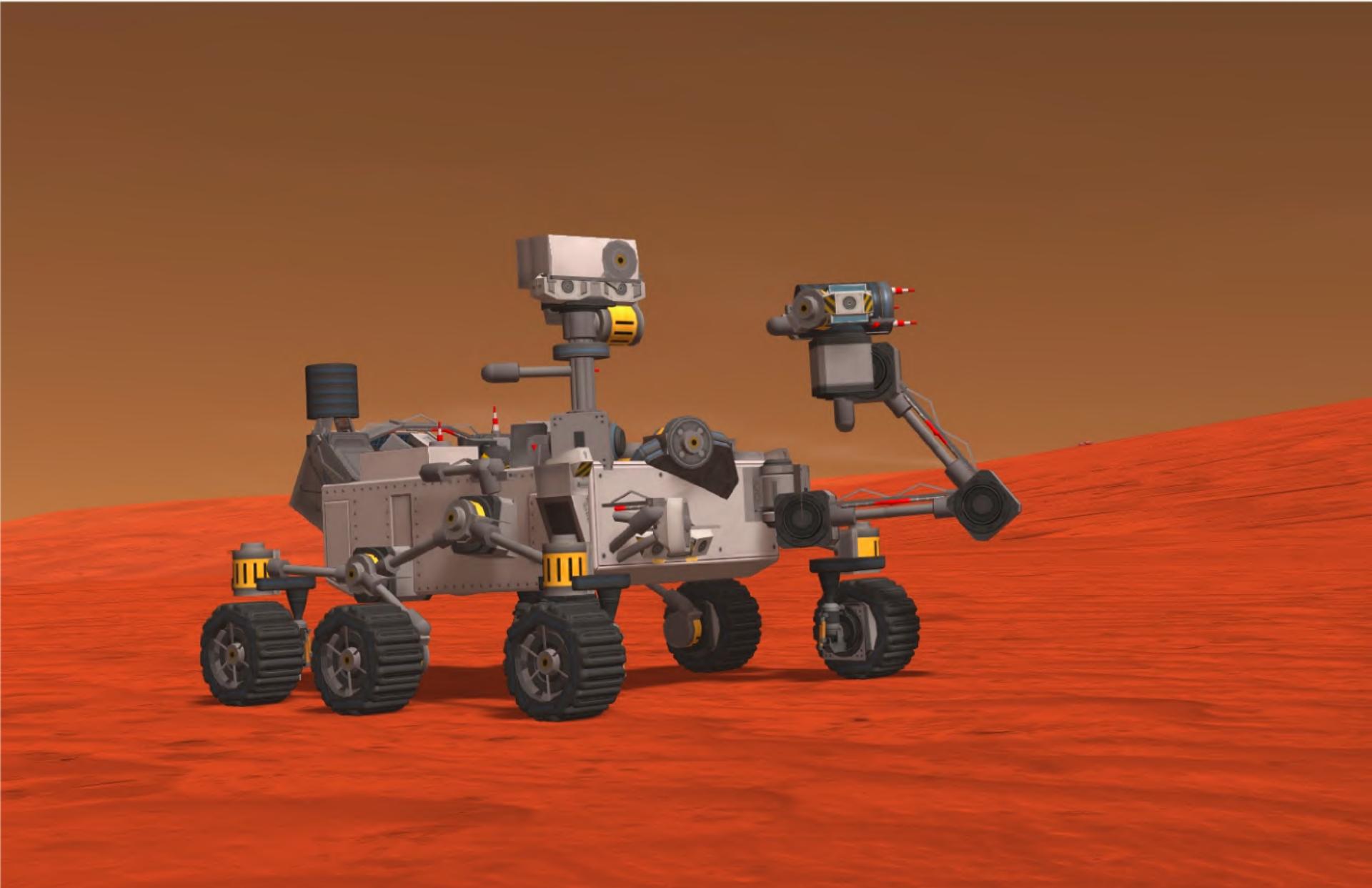
LOVRO VRLEC

03 / 2023

Moon Rover App

Today's Agenda

- 01 Problem
- 02 Use Cases
- 03 Model
- 04 API
- 05 Error Handling Design
- 06 Implementation
- 07 Q&A



SECTION 01 -

Problem

A simulation of a moon rover moving on a square tabletop (dim. 5x5 units)

No other obstructions on the surface

Placement on the tabletop + coordinates + showing location and directions

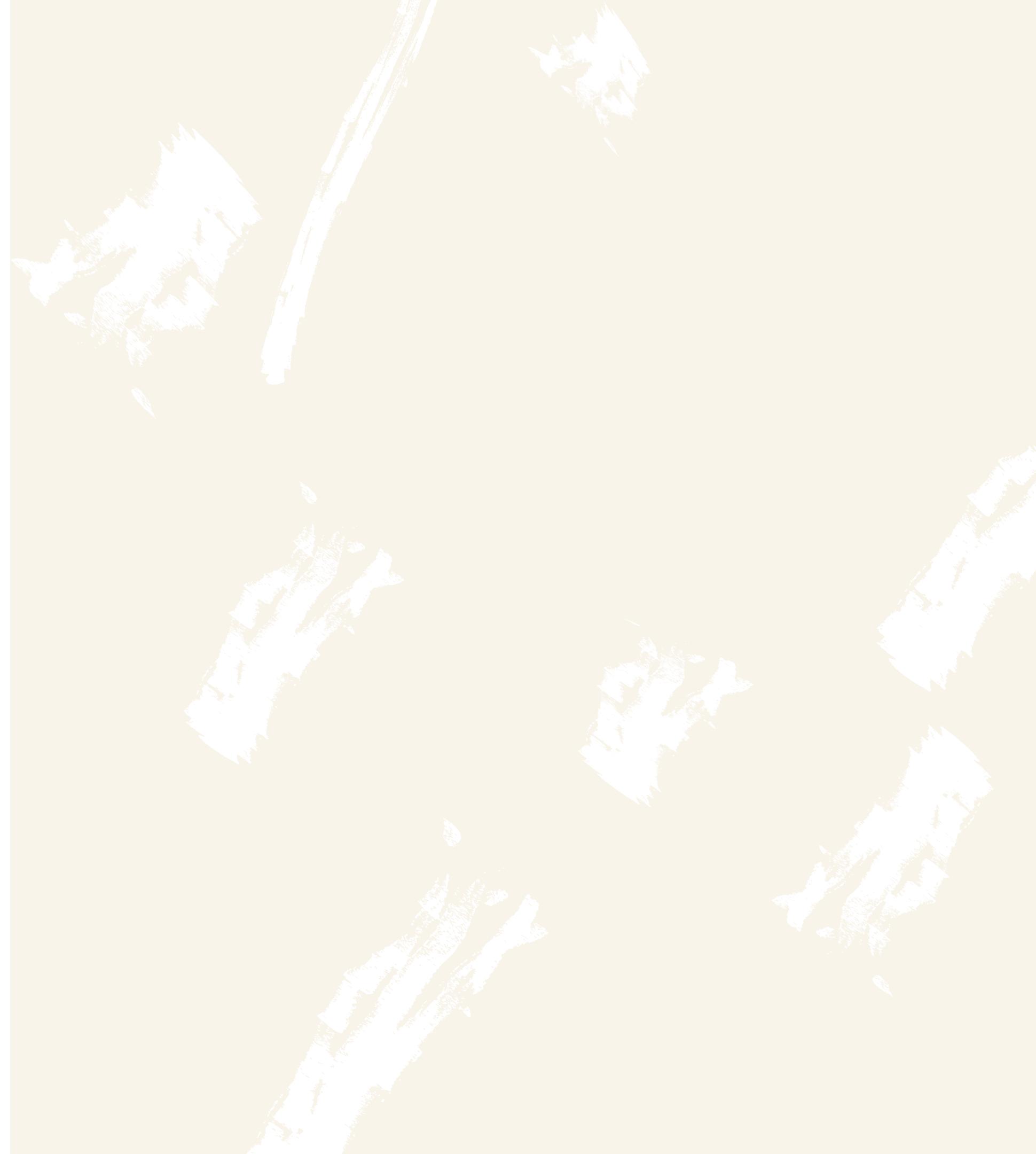
The moon rover can turn on the spot and its facing changes accordingly

Freedom to roam around + prevention from falling to destruction

SECTION 02 -

Use Cases

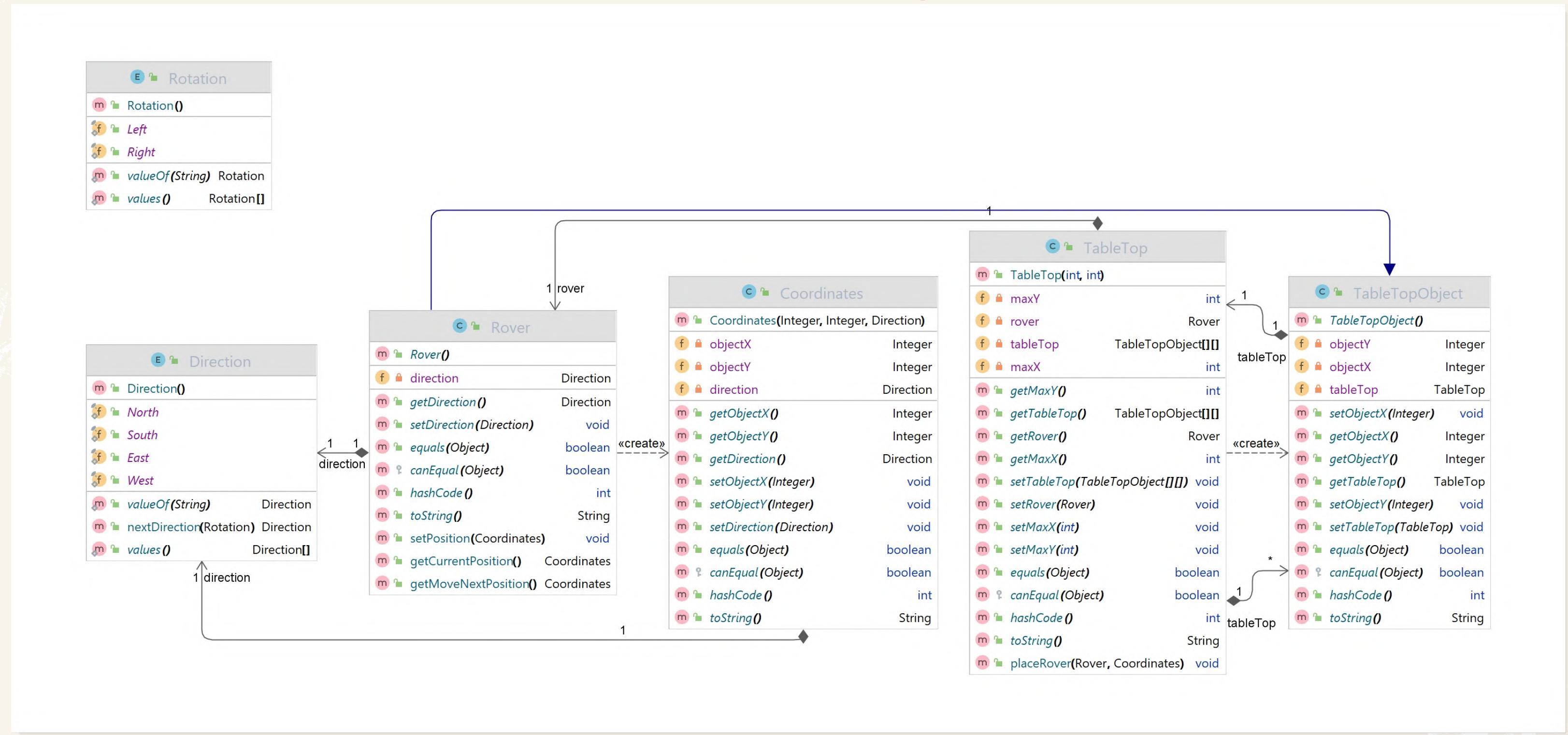
- The rover is placed on the surface to the **specific coordinates and facing**
- It is possible to send a **move command** to the rover
- The rover moves one unit **in the facing direction**
- It is possible to send a **turn command** to the rover (left / right) + its facing changes accordingly
- It is possible to **request a report** from the rover about its position and its facing
- Commanding the rover should be realised via a **REST API**



SECTION 03 -

Model

Model Design



SECTION 04 -

API

API Summary

rover-controller

- POST** [`/api/rover/turn`](#) Turns rover direction on tabletop ▼
- POST** [`/api/rover/place`](#) Places rover on tabletop ▼
- POST** [`/api/rover/move`](#) Moves rover on tabletop in rover direction ▼
- GET** [`/api/rover/report`](#) Rover reports its current location on tabletop ▼

Place API

POST /api/rover/place Places rover on tabletop

Parameters

No parameters

Request body required

application/json

Example Value | Schema

```
{  
  "objectX": 0,  
  "objectY": 0,  
  "direction": "North"  
}
```

Responses

Code	Description	Links
200	Rover placed on tabletop	No links
400	Invalid tabletop position supplied	No links

Media type

application/json

Controls Accept header.

Example Value | Schema

```
{  
  "objectX": 0,  
  "objectY": 0,  
  "direction": "North"  
}
```

Turn API

POST /api/rover/turn Turns rover direction on tabletop

Parameters

No parameters

Request body required

application/json

Example Value | Schema

```
"Left"
```

Responses

Code	Description	Links
200	Rover turned to desired direction	No links
400	Invalid turn direction supplied	No links
404	Rover not found	No links

Media type

application/json

Controls Accept header.

Example Value | Schema

```
"string"
```

Move API

POST /api/rover/move Moves rover on tabletop in rover direction

Parameters

No parameters

Responses

Code	Description	Links
200	Rover moved forward one step on tabletop	No links
	Media type	
	application/json	
	Controls Accept header.	
	Example Value Schema	
	"string"	
404	Rover not found	No links

Try it out

Report API

GET /api/rover/report Rover reports its current location on tabletop

Parameters

No parameters

Responses

Code	Description	Links
200	Rover placed on tabletop	No links
400	Invalid tabletop position supplied	No links
404	Rover not found	No links

Media type

application/json

Controls Accept header.

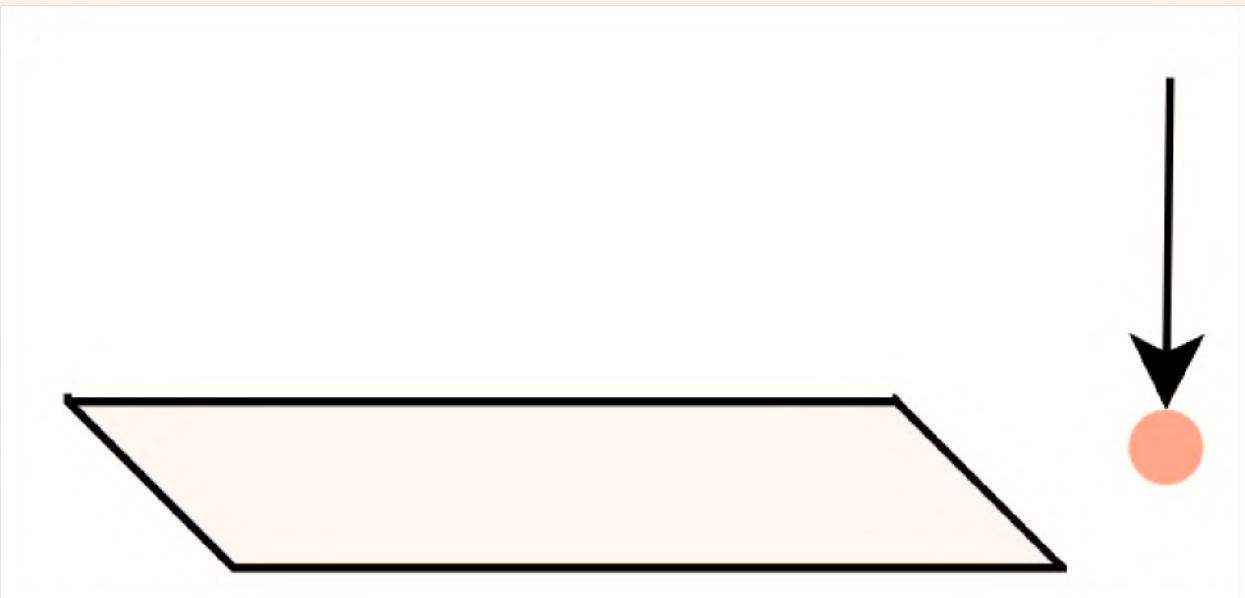
Example Value | Schema

```
{  
  "objectX": 0,  
  "objectY": 0,  
  "direction": "North"  
}
```

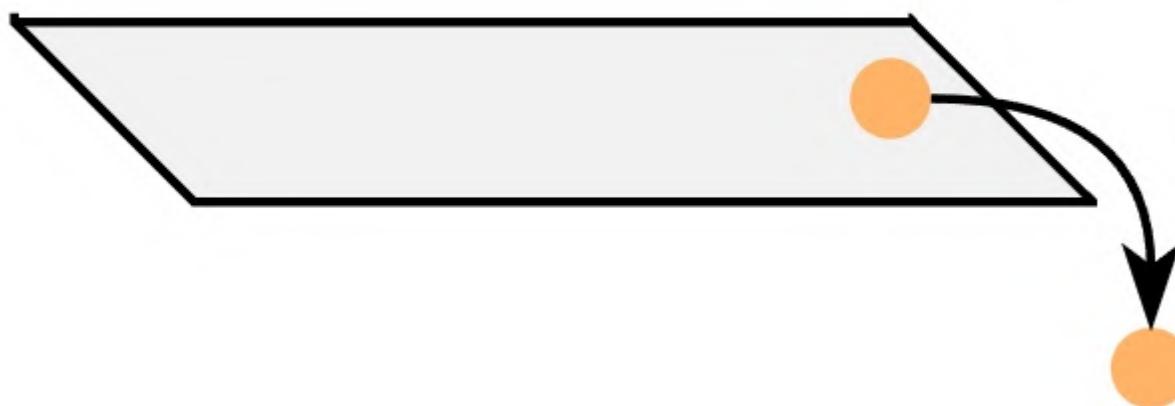
SECTION 05 -

Error Handling Design

**Placing of the rover outside
the bounds of the tabletop**

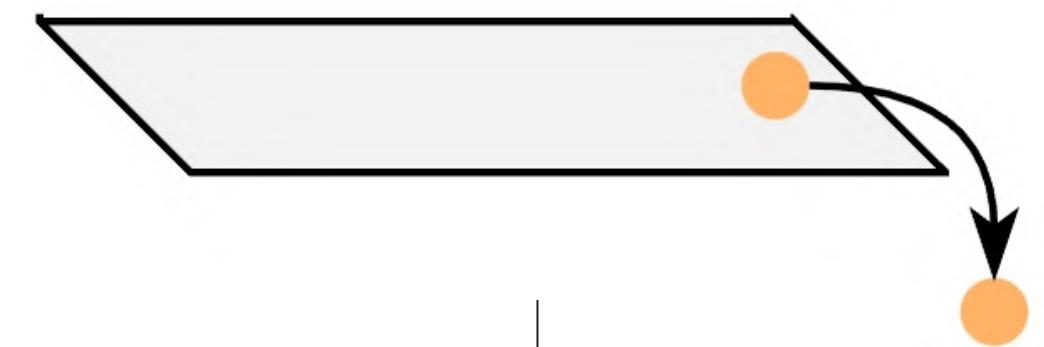


**Moving with the rover
off the tabletop**

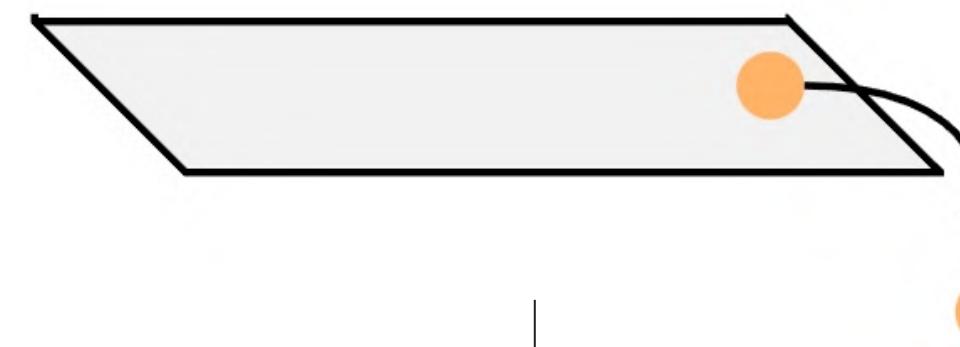
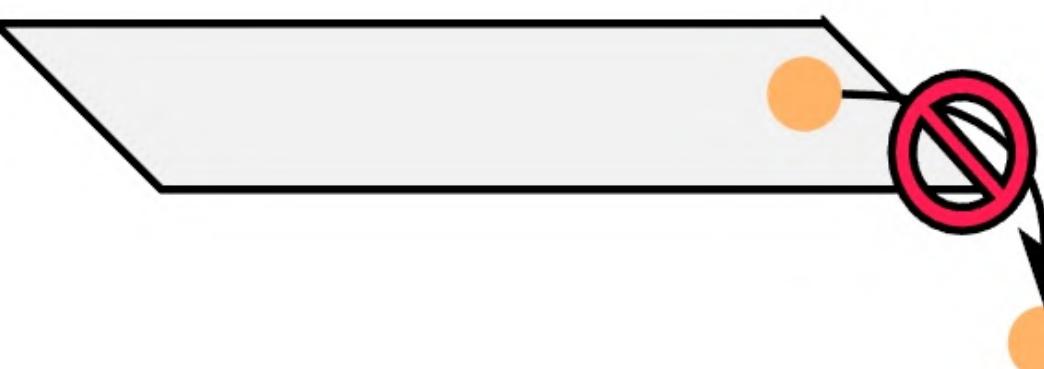


Moving with the rover off the tabletop

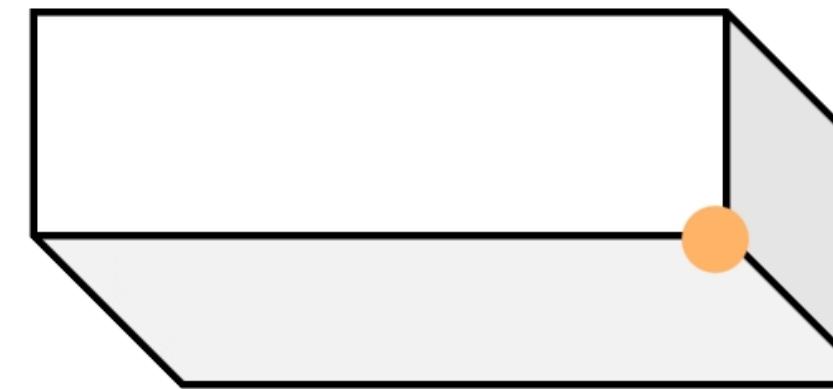
CHOICES



↓



↓



SECTION 06 -

Implementation

Live show of code

• • •

The screenshot shows a Java IDE interface with several tabs at the top: RoverController.java, MoonRoverPlaceAndMoveIT.java (selected), MoonRoverRotateAndMoveIT.java, and TableTop.java. The MoonRoverPlaceAndMoveIT.java tab has a green checkmark icon. The code in the editor is for a test class:

```
13 import org.springframework.test.web.servlet.MockMvc;
14 import org.springframework.test.web.servlet.request.MockMvcRequestBuilders;
15
16 import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.*;
17 import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.*;
18
19 /**
20 * created by lovro.vrlec@gmail.com on Mar, 2023
21 */
22 @ExtendWith(SpringExtension.class)
23 @SpringBootTest
24 @AutoConfigureMockMvc
25 public class MoonRoverPlaceAndMoveIT {
26
27     @Autowired
28     private MockMvc mockMvc;
29
30     ObjectMapper mapper = new ObjectMapper();
31
32     @Test
33     void testMoveRoverNorth() throws Exception {
34
35         Coordinates coordinatesPlace = new Coordinates( objectX: 3, objectY: 3, Direction.North );
36         Coordinates endPlace = new Coordinates( objectX: 3, objectY: 4, Direction.North );
37
38         mockMvc.perform( MockMvcRequestBuilders
39                         .post( urlTemplate: "/api/rover/place" )
40                         .content( mapper.writeValueAsString(coordinatesPlace) )
41                         .contentType(MediaType.APPLICATION_JSON)
42                         .accept(MediaType.APPLICATION_JSON) )
43                         .andExpect(status().isOk());
44
45         mockMvc.perform( MockMvcRequestBuilders
46                         .post( urlTemplate: "/api/rover/move" )
47                         .content( mapper.writeValueAsString(coordinatesPlace) )
48                         .contentType(MediaType.APPLICATION_JSON)
49                         .accept(MediaType.APPLICATION_JSON) )
50                         .andExpect(status().isOk());
51
52         mockMvc.perform( MockMvcRequestBuilders
53                         .get( urlTemplate: "/api/rover/report" )
54                         .content( mapper.writeValueAsString(coordinatesPlace) ) )
```

The code uses Spring Test and MockMvc to perform HTTP requests to a rover API. It creates a 'place' endpoint to set a rover's initial coordinates (3, 3, North) and a 'move' endpoint to move it one unit North (3, 4, North). Finally, it checks the 'report' endpoint to ensure the rover is at the expected position.

SECTION 07 -

Questions & Answers

Thank you

Let's work together.

LOVRO.VRLEC@GMAIL.COM

+385 99 3443540