# Mars Rover Documentation (Team Tanganda)

This Document serves to explain the Implementation of the **marsrover** which was done by team **Tanganda**. To simplify this explanation, I am just going to break this down and explain each module, its class, and the methods of each class.

First of all, we took over code which was done by the pervious team. Their implementation involved three modules and the main class. These are the models they had:

1. **Model**: In which they defined the parameters of the client connection.
2. **Util**: In which they defined the actual values of the parameters for connection, i.e. hostname, port number and the server.
3. **Service**: In which they had 2 more sub modules, which are:

* **Api**: where they defined the request method.
* **Impl**: where they defined the methods to actually send a request to the server and another method to get sever response.

The above work was done by the previous team. To find this work on GitHub, you go to the repository and then click **commits.** The first commit is the one.

# WHAT WE DID (continuation)

We went on to break down the implementation into different modules to make sure each module performs a simple task. We created six (6) more modules, from the three (3) they had. To make them 9 in total. The following are the modules and their classes and their tasks:

1. Action: This module contains classes which deals with the movements or actions of the rover, i.e. turn left, turn right, move. These are the classes involved:

* **IAction.java**: this is an Interface that will be implemented by all other action classes.
* **LeftAction**.**java**: this class is responsible for turning left of the rover.
* **RightAction**.**java**: this class is responsible for turning right of the rover.
* **MoveAction**.**java**: this class is responsible for moving forward of the rover.

1. Domain: This module contains one class which deals with the position of the rover.

* **Position.java:** Using the campus (North, West, East, South) , This class determines the position of the rover, and when moving the rover, checks where the rover is facing and then moves the rover accordingly.
* **Example:** If the rover is facing NORTH, method **Position getLeft ()** returns WEST.

1. **Exception:** This modules contains two classes which are responsible for exception or error handling. The classes are:

* **CommandInvalidException.java:** This class is responsible for catching invalid commands entered by the user.
* **ExceededBoundryException.java**: This class is responsible for catching the area boundary of the rover before it moves. When the user enters the command.

1. **Model:** This module contains three model classes which are the models used by the client. The classes are:

* **Controller.java:** This class is the controller model which defines controller parameters.
* **Area.java:** This class is the area model which defines Area parameters.
* **Coordinate.java:** This class is the coordinate model which defines the parameters of the coordinates. The X coordinate and the Y coordinate.

1. **Parse:** This module contains one class which is responsible of parsing the command to the controller to move the rover.

**CommandParse.java:** is the class which is responsible for parsing the commands. Parsing of commands are done by the user e.g. LLM means turn left, turn left, move.

1. Util: This module contains three classes which are responsible of defining the actual values which are going to be used for implementation.

* **ActionUtil.java**: This class is responsible of defining the actual values of actions, e.g. L for LEFT, R for Right, and M for Move.
* **AreaUtil.java**: This class is responsible for defining the set methods of the Area.
* **ClientUtil.java**: This class is responsible for defining the set methods of the client.

1. Validator: This module contains only one class which is responsible for validating the boundary of the movement of the rover.

**BoundaryValidator.java** is the class which validates to check if the rover movement falls within the boundary by checking the X coordinate and Y coordinate.

1. Service: This module contains only one class which is responsible for controlling the rover.

**ControllerService.java:** is the class that actually controls the rover. The class contains four methods:

* **MoveTo ():** This method is responsible of taking commands of the user to see where the rover is supposed to go.
* **setBoundaryValidator ():** this method is a set method for boundary validation.
* **setCommandParse ():** this method is a set method for command parsing.

1. **Controller:** This module contains two classes which are responsible of making requests to the server.

* **Controller.java:** This class is responsible for carrying http request to the server.
* **ControllerExceptionHandler.java**: This class is responsible for catching http request errors or exceptions, to the server.