Accio Preliminary Round

HANDS-ON EVALUATION TASK

Role: Full Stack Developer

Problem Statement:

Consider the scenario of **N number of robots** being deployed in a warehouse as part of an automated picking operation. Robots use a **LxW grid** (graph with nodes and edges) for navigation, which covers the area of the warehouse. The robots are connected to a server via WiFi and periodically update their status via messages using **MQTT protocol**. These **status messages** contain ID of the robot (**robot_id**), the node where robots currently are (**current_node**), state of the robot (**robot_state**) the strength of the WiFi signal in that area (**signal_strength**), battery charge information (**battery_percentage**) and whether an obstacle is present in front of the robot or not (**obstacle_status**).

When a robot reaches a node on which a pick is to be performed, the robot will send a status message acknowledging the availability to pick (**robot_state: readyToPick**). Upon receiving this status, an **API call** has to be made to the Accio server to get the details of the pick that needs to be done at that node (**order_id, item_id, quantity**). This information is sent to the robots as **action messages** via **MQTT protocol**. Action message will contain **robot_id, order_id, item_id** and **quantity.** After the picking is completed, the robot will send a status message with confirmation of the pick (**robot_state: pickCompleted**). This will initiate another **API call** to the Accio Server to acknowledge the successful completion of picking tasks.

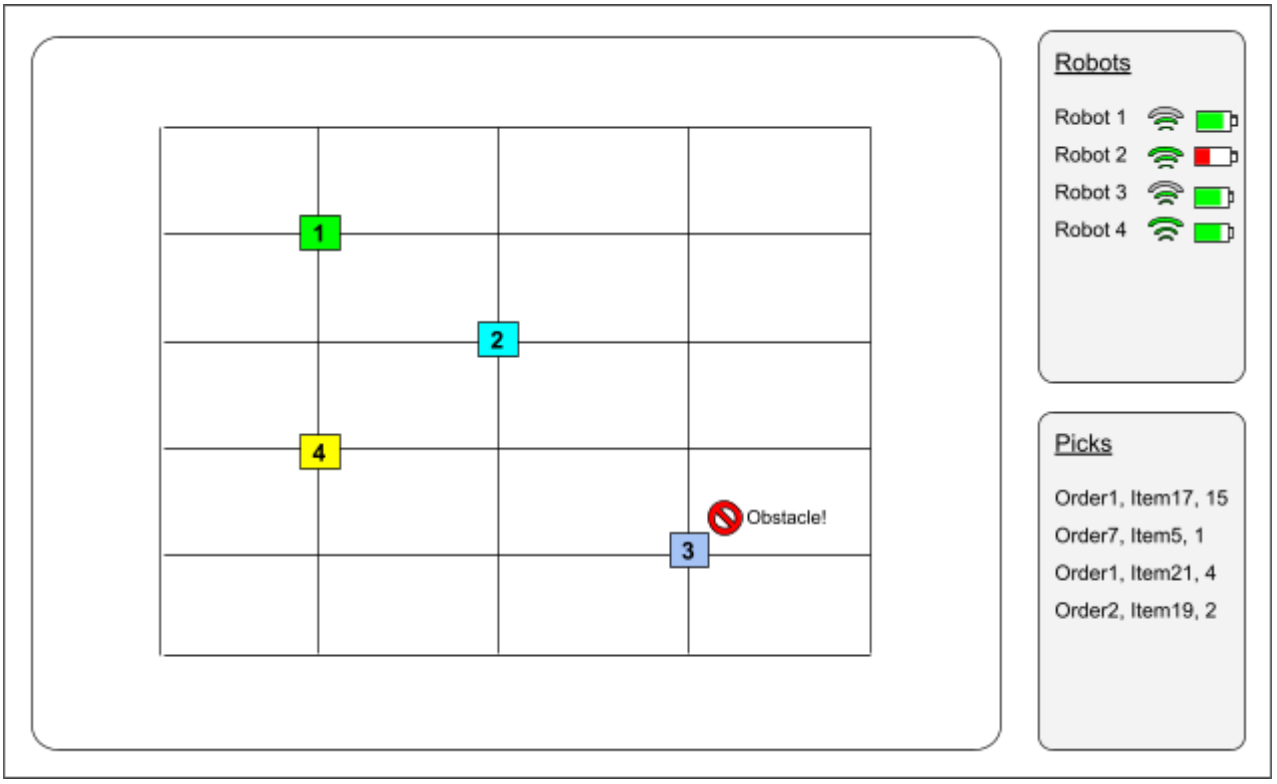
Task:

A locally hosted WebApp needs to be created, which serves as the User Interface for the Accio Server. This App should read a **Json File** which contains deployment parameters (**N, L** and **W**) and **display a grid structure**. The App should also have a **List View of robots**, with their **WiFi strength, battery percentage displays**. According to the

received status messages, **visual markers for robots** should be displayed on the grid in their **current node locations** on the grid structure. If an obstacle is reported by any robot, **appropriate visual indication** should be displayed along with the visual markers of robots. After every API calls for getting the pick information, the same should be **appended and displayed** in a **List View of picks**. Once pick is completed, and the acknowledgement to the server is done, the corresponding pick information has to be **removed** from the List as well. Action messages to the Robot should be **sent using MQTT protocol** promptly after receiving the pick information from the server.

Helpers/ Pointers:

- The diagram below can be used as a guide to the frontend of the app. Remember, this is just an example. Candidates have the complete freedom (and encouraged) to choose innovative layouts, components, widgets etc. as long as all functionalities are serviced.



- The python code **robot_emulator.py** provided with the task creates MQTT clients and generates MQTT status messages for all robots. Remember to run MQTT broker (of your choice) in the background.
- The **parameters.json** file with necessary deployment parameters is also given with the task.
- The test server emulating Accio Server for API calls can be started by running the python script **server_emulator.py**

Note: The submission should include all the files, assets and dependencies or clear instructions or scripts to install dependencies. A small **documentation** is also expected detailing the assumptions made, app functionalities, limitations and scope of future work. In case of any minor errors or lack of compliance in the provided emulator codes, candidates are **encouraged to rectify** those themselves and report those in the documentation. If more clarity is needed, you can reach out to Accio via email. The task is expected to be completed **within 2 to 3 days** and time taken to finish the task after receiving the task will be noted.