In our program, we have a pool of strategies for collecting cargos and going to the destination. Depending on the situation at hand, the most appropriate strategy will be chosen at each step of making decision. Below are some of the available strategies:

a. **Greedy:** At every step, our program chooses to collect the lightest package at the position that we have not visited

b. **Smart Greedy algorithm:** In case the packages chosen by the Greedy strategy are far apart from each other, we will cluster packages close to each other into groups. In other words, packages belonging to the same group are close to each other. The weight of each group is the sum of the member packages. We will then visit each group in the increasing order of their weights and use the Greedy algorithm for the strategy of picking up all packages in one group.

c. **Consider other teams' moves:** Collecting the lightest or nearest package is a good move but when all teams are heading towards that same position, it might not be that good anymore. We consider this case and will pick another package, which is not a possible target for other teams.

The match data and updated information at each step will be saved to a file from which our program reads and processes. Based on this information and the past behavior of other teams, we will pick the best-at-the-time strategy for the next moves.

We use Java to build a graphical user interface (GUI) for displaying the updated information of the map. This helps us to visualize the strategies of our teams and others' in order to make decision for the next moves. We use C++ to write the strategies and Java to write supporting tools such as the GUI display.