Troops Marching

Step1 : clicking a marching ground:

 if(IsGroundLayer(ClickedObject)||IsEnemyLayer(ClickedObject)||IsMineLayer(ClickedObject)){

            Debug.Log("March target clicked");

            troopsExpeditionManager.PotentialTargetForMarchClicked(ClickedObject,hit);

        }

In TroopsExpeditionManager:

 public void PotentialTargetForMarchClicked(GameObject Target,RaycastHit hit){//clickedObject

        //this will be called by global

        target=Target;  //storing target

        position=hit.point;

        //trigger march manager stage 1 -----

        expeditionUI.TriggerConfirmationUI();

    }

In ExpeditionUI:

 public void TriggerConfirmationUI(){

        //by expedition manager

        InitialConfirmPanel.SetActive(true);//the one will with tick and cross panel    }

Now after confirming Ground click to march (UI buttons), clicking tick button. It triggers a function ExpeditionUI:

public void Stage2ConfirmationUI(){//triggered by stage 1 confirmation ui

        //the one with exiting army or creating another army option

        ConfirmPanel2.SetActive(true);

        Armys=troopsExpeditionManager.GetAllThePresentUnits();

            int armyCount = Mathf.Min(Armys.Length, 5); // Ensure we handle only up to 5 armies

            if(armyCount>=ArmyLimit){

                CreateButton.SetActive(false);

            }

            else{

                CreateButton.SetActive(true);

            }

            for (int i = 0; i < armyCount; i++) {

                ArmyButtonGO[i].SetActive(true);              // Activate the button

                armyId[i].text = Armys[i].ArmyId.ToString();  // Set the army ID on the button

                int index = i; // Capture 'i' locally to prevent closure issue in the listener

                ArmyButton[i].onClick.AddListener(() => ArmyIsChosen(Armys[index])); // Assign the listener

            }

// Optionally hide extra buttons if fewer than 5 armies

            for (int i = armyCount; i < ArmyButtonGO.Length; i++) {

                ArmyButtonGO[i].SetActive(false); // Hide buttons for unused army slots

            }

    }

In TroopsExpeditionManager:

public TheUnit[] GetAllThePresentUnits(){//this will be called by ui indirectly for showing info of

    //current present army

        //by ui stage1 confirmation

        theUnits = FindObjectsOfType<TheUnit>();

        return theUnits;

    }

expeditionUI

 void ArmyIsChosen(TheUnit ChoosedOne){

        ChoosenUnit=ChoosedOne;

        // Debug.Log("clicked"+ChoosedOne.ArmyId);

        troopsExpeditionManager.ExistingArmyIsChoosen(ChoosenUnit);

        EndStageTriggered();

    }

In TroopsExpeditionManager

public void ExistingArmyIsChoosen(TheUnit choosedOne){

        //By expedition ui

        ArmyIsChoosed(choosedOne);

    }

    void ArmyIsChoosed(TheUnit choosedOne){

        ChoosenUnit=choosedOne;

        //march it.

        march();

    }

    void march(){

        // Debug.Log(ChoosenUnit.ArmyId);

        ChoosenUnit.SetTroopsTarget(position,target,SpawnPoint);

        EndStage();

    }

    public void EndStage(){

        globalUIManager.RefreshPermission();

    }

ExpeditionUI

public void EndStageTriggered(){

        //this will be triggered by ui cancel buttons

        // refresh all ui;

        InitialConfirmPanel.SetActive(false);

        ConfirmPanel2.SetActive(false);

        troopsExpeditionManager.EndStage();

    }

TheUnit:

public void SetTroopsTarget(Vector3 position,GameObject Target,GameObject SpawnPoint){

        spawnpoint=SpawnPoint;

        StopAllAction();

        target=Target;

        if(target.layer==6){

            SetTargetPosition(position);

        }

        else{

            SetTargetPosition(target.transform.position);

        }

    }

void StopAllAction(){

        if(isMining){

            isMining=false;

            mining.StopMining();

        }

    }

The indirectly trigger ui give two ways. It lets us choose whether to march existing army or create another.