const DecentralizedFederatedLearning = artifacts.require('DecentralizedFederatedLearning');

contract('DecentralizedFederatedLearning', (accounts) => {

let decentralizedLearning;

before(async () => {

decentralizedLearning = await DecentralizedFederatedLearning.deployed();

});

it('should publish a task', async () => {

await decentralizedLearning.publishTask('Test task');

const task = await decentralizedLearning.currentTask();

assert.equal(task.description, 'Test task', 'Task not published');

});

it('should authorize a node', async () => {

await decentralizedLearning.authorizeNode(accounts[1]);

const isAuthorized = await decentralizedLearning.authorizedNodes(accounts[1]);

assert.equal(isAuthorized, true, 'Node not authorized');

});

it('should participate in a task', async () => {

await decentralizedLearning.participateInTask({ from: accounts[1] });

const isParticipant = await decentralizedLearning.taskParticipants(accounts[1]);

assert.equal(isParticipant, true, 'Node not participating');

});

it('should complete a task and trigger federated learning', async () => {

await decentralizedLearning.completeTask({ from: accounts[1] });

const isTaskCompleted = await decentralizedLearning.currentTask.completed();

assert.equal(isTaskCompleted, true, 'Task not completed');

// Additional assertions for federated learning can be added here

});

});

}