Deadline team submit: 01/03/2024

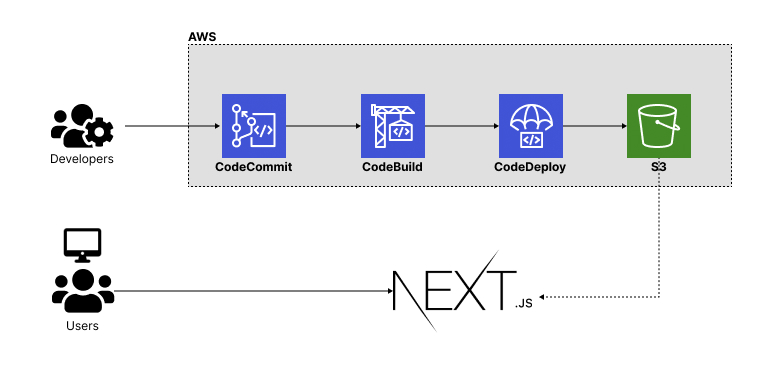
Deadline submit: 07/03/2024

Front-end của Dapp

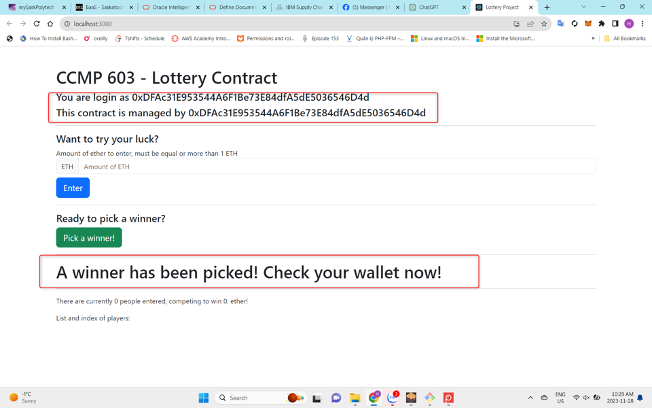
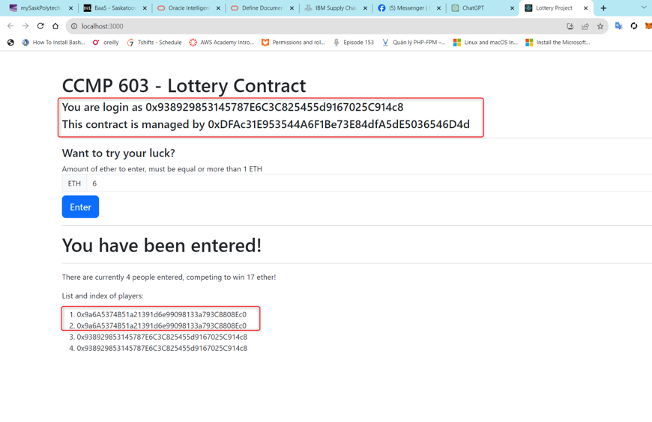
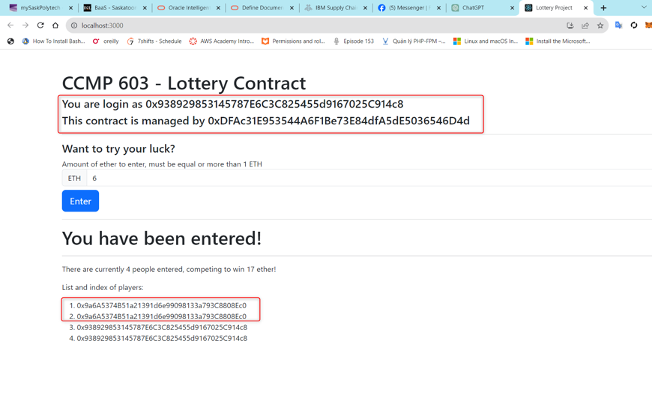
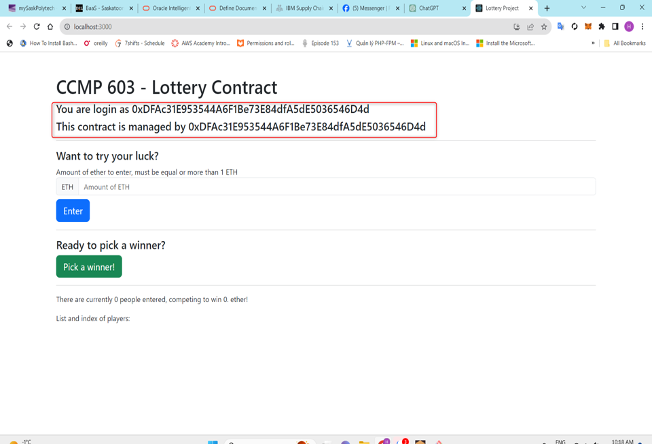
Proposal (5 points) explains your proposed solution. It should contain:

* **Problem Definition**: This project aims to solve some common problems that have been happening while developing our front-end:
  + **Versioning control**: it’s difficult to manage changes across teammates without a versioning control system. This thing will be a barrier for collaboration within our team in code discussion and development which needs to be centralized. In consequence, this can lead to conflicts, code override and difficult to track changes.
  + **Build and deployment process**: Development cycles can be slowed down by build and deployment procedures that are manual or inefficient. By using different compilers can be created different deployment packages leading to inconsistencies.
  + **Deployment environment:** using different configuration and installing manually to create environment for deployment, testing or production can lead to unexpected errors and failure deployment.
* **Nam:** Design and Architecture (2 points)

Our application will use modern technologies like Amazon Web Services (AWS) or DevOps, NextJS for the front-end. To buid it, the team will use AWS CodePipeline, CodeCommit, CodeBuild, and CodeDeploy to manage Continuous Integration and Continuous Delivery (CI/CD). This process is dedicated to the front-end application, which will be deployed to AWS Simple Storage Services (S3) and hosted as a static website.



The front-end will be developed in NextJS which use ReactJS as the core library, combining with CSS Tailwind Framework, the simple of it look like the images below:



* Tai: Plan of Implementation Stages (1 point) (AWS Codecommit, CodeBuild, CodeDeploy, S3 bucket, Route53)
  + Setting Up Version Control with AWS CodeCommit
    - Create a new AWS CodeCommit repository for the project.
    - Set up access controls and permissions for team members.
    - Migrate existing code repositories
  + Building and Testing with AWS CodeBuild
    - Configure AWS CodeBuild to trigger builds automatically upon code changes.
    - Define build specifications and scripts.
    - Integrate with testing frameworks for automated testing.
  + Deployment Automation with AWS CodeDeploy
    - Set up deployment groups and deployment configurations in AWS CodeDeploy.
    - Define deployment scripts and configurations.
    - Integrate CodeDeploy with CodeBuild for continuous deployment.
  + Configuring AWS S3 Bucket for Storage
    - Set up an S3 bucket for storing application.
    - Configure access policies and permissions for secure storage.
    - Integrate CodeBuild and CodeDeploy with the S3 bucket for retrieval during deployment.
  + DNS Configuration with AWS Route 53
    - Register domain names and configure hosted zones in Route 53.
    - Set up DNS records for the application, including aliases for load balancing.
* **In Conclusion**, applying Continuous Integration and Continuous Deployment (CI/CD) in software deployment can reduce human mistakes, inconsistencies, unexpected failures. CI/CD offers speed, efficiency, and high-quality delivery by automating tasks like code validation and deployment. It assures consistency and reliability in deployment environments, enabling collaboration among team members. For this reason, organizations are able to concentrate on business goals to reach customer satisfaction.

The proposal has a minimum requirement of 3 pages (12-point fonts and 1.5 line spacing). You can have pictures/diagrams/dataflows to help illustrate the technologies/tools that will be used in your solution.