Development Project

SC4053/CE4153/CZ4153: Blockchain Technology

Project	**Development Project**	Week 6 to Week 12	35% of total marks

General Information

In the Development Project, you are expected to design and develop a **decentralized Application (dApp)** using tokens and smart contracts on Ethereum. The Development Project is a group activity with a component for peer-assessment in the group. You may work in groups of 2 to 3 students on this project. Single-member groups will need explicit permission.

Development Resources

There are several online resources (articles, videos, tutorials) to help you with development of dApps on Ethereum. Feel free to consult any online resource. We have put together a set of selected resources in the following GitHub repository.

BlockchainCourseNTU on GitHub: https://github.com/BlockchainCourseNTU/resource/

- Check the **Quick Access** links at the repo to find the main resources quickly.
- Check the **hello-dapp** page for a detailed introduction to dApp development.
- Make sure your system is ready with all the necessary prerequisites installed.
- Check the **basic** and **advanced Solidity Syntax** on the repo before you begin.

During the development, if you are confused about the structure and operation of Ethereum, you may feel free to refer back to the LAMS Sequences (lectures) on Ethereum or check out the following Medium article for a quick description.

How does Ethereum work, anyway? by Preethi Kasireddy
https://medium.com/@preethikasireddy/how-does-ethereum-work-anyway-22d1df506369

Needless to mention, you can always reach out to the Instructors and the TAs at the **Discussion Board on NTULearn**.

Evaluation

This is worth 35% of your total score. Instructors can't directly assess the contribution of each individual student to the completion of this team project. Hence, each student is required to rate the contribution of themselves and each of the other team members. All evaluations are held in confidence so no student will know how other teammates rate his/her contribution. You are to evaluate yourself and other team members fairly and objectively, bearing in mind the implications for the other members' marks (explained later). It is absolutely essential for you to submit your peer evaluation (for yourself and your teammates) to get marks for your team project. Details are at the end of this document.

Option 1: Decentralized Exchanges

Project in a Nutshell

Build a minimal viable Decentralized Exchange (DEX) on Ethereum and a simple (minimally styled) front-end website, which supports listing of available asset tokens on the marketplace, submission of trading order, matching and execution of orders (i.e., swapping/exchanging/trading assets), and most importantly, in our DEX, users have the ultimate control of his/her own digital assets.

Background & Problem Statement

Traditional trading sites and centralized exchanges allow users to purchase and trade different digital assets on their platforms. For example, your first time purchasing an Ethereum token or Bitcoin was probably through a centralized exchange like Coinbase, Binance, Bittrex, Huobi and alike. The central thesis for these exchanges is "key management is hard for average users", thereby they will manage cryptographic keys for you and provide more friendly interface as a service, and all you, the user, need to do is authenticate yourself to these platforms using traditional login username and passwords. The actual private keys that control and "own" the assets/cryptocurrencies are in the database of these companies. Evidentially, all these platforms claim that they have best security team in the world to secure your keys and therefore your assets from stolen or lost.

Unfortunately, there are no lack of examples of those platforms mishandling or losing users' assets, either by internal corruption, or by external hacks. The most infamous Mt.Gox, a centralized bitcoin exchange, which at one time is handling 70% of all bitcoin trading – the largest in the world, until they got hacked and lost majority of its bitcoin, or rather its customers' bitcoin and filed bankruptcy.

Fundamentally, the problem is that **your cryptocurrencies and digital assets are as secure as your key management**. Regardless of all the cryptographic protections and techniques we cover previously in our course, and how unhackable the underlying math is, **if you don't own your keys, then you don't really own your asset. And asking a centralized exchange to hold your keys for you is a huge trust assumption and liability.**

With this problem in mind, here comes a promising solution – decentralized exchanges or DEX for short.

What distinguish DEX is that, users would need to manage their keys in their digital wallet locally (e.g., MetaMask). Now, no one else would have control over your money – there isn't a honeypot of millions of bitcoin sitting in certain company's database waiting for hackers to get a shot, you don't have to trust any third party that your assets are safe.

In a DEX system, the exchange only manages buy/sell orders from users – exactly like a marketplace where sellers put out the prices range and amount to exchange, and later on some buyers will fulfill that request. In simpler term, DEX are places where you bring your own key/wallet, to trade your digital assets (e.g., USD token) with other assets (e.g., Gold Token, ETH token) -- and this process is referred as token exchange, or token trading, or token swap. Here are some top tokens/digital assets on Ethereum blockchain.

Feature Requirements

A decentralized exchange on any one of Ethereum Testnets (i.e., Goerli, Ropsten, Rinkby, or Kovan) that supports the following features:

- For Asset Issuer:
    - issue new asset tokens (Only use ERC20 token standard)
- For Users:
    - Submit buy and sell orders (there are many types of order in finance world, for simplicity, try to support limit order first. If time permits, adding supports for other types of order will be bonus points)
    - Matched orders in the order marketplace will be executed, please noted that an order can be partially fulfilled (e.g., Alice want to sell 10 TokenA for 5 TokenB, whereas Bob only wants to sell 1 TokenB for 2 TokenA, then 20% of Alice's order will be fulfilled, 100% of Bob's order will be fully fulfilled)

- (bonus 1) support canceling order where users decide to cancel whatever remaining amount in his/her previously submitted buy/sell order
- (bonus 2) support batched execution of order matching (e.g., Alice wants to sell 10 TokenA for 5 TokenB, Bob wants to sell 5 TokenB for 3 TokenC, and Charlie wants to sell 3 TokenC for 10 TokenA, then batched execution will facilitate the token swaps among Alice, Bob and Charlie)
- (bonus 3) support some form of conditional order.

Tips

- try to buy and sell crypto assets at platforms listed below (see Leading DEX Projects) to get a feel of the process flow. (Don't spend too much, just purchase < 20 SGD for research purpose, you can invest on your own term off the course)
- Draw out the lifecycle of an order
- Draw out the architecture of your design and interaction between different Personas with your smart contract systems on-chain
- Start simple, make sure the whole data flow works, and then add more features iteratively
- Always have unit tests for every new functionality

Tricky points to ponder

- How to, how often and who should trigger an execution of a matched pair of orders?
- Should "older, unfulfilled" orders have a higher priority of being matched when other conditions being equal with other orders?

Leading DEX Project Examples

- Uniswap (https://uniswap.org/)
- Kyber Network (https://developer.kyber.network/docs/Start/)
- 0x Project (https://0x.org/docs)
- Gnosis Protocol (https://docs.gnosis.io/protocol/)
- Curve Finance (https://www.curve.fi/)

Option 2: Dutch Auction

Project in a Nutshell

Implement smart contracts on Ethereum and simple (minimally styled) front end webapp for a new token launch (STO/ICO) whose tokens are bid with Ether and distributed via a Dutch Auction over 20 minutes time.

Background & Problem Statement

"Auction" in the most canonical connotation, might depict a scene in your mind where rich people are trying to purchase an expensive art collection with ever increasing higher bid, resulting in some news-worthy finalized clearing price. -- this is usually referred as "forward first-price auction", ones that provide suppliers the opportunities to find the best price among interested buyers with ascending price tag.

In contrast with "first-price auction", "Dutch auction" is another type of auction we often seen in a STO (Security Token Offering, whose unregulated predecessor known as ICO, initial coin offering). In a Dutch auction, there are few key important features:

1. Everyone who buys the asset will get it at the same price per unit
2. The price per unit is determined by the market
3. The price starts high and descends over time

One of the most prominent blockchain projects, Algorand, uses Dutch Auction to distribute its network token (i.e., native cryptocurrency on their chain). Please watch their explainer video on how Dutch Auction works (auxiliary blog post).

Feature Requirement

Dutch Auction smart contracts should:

- Firstly, define and implement your new token using the ERC20 standard
- Implement Dutch auction logic in another contract(s)
- Only elapse for 20 minutes, either all tokens get sold out at clearing price no lower than the reserved price, or only part of total token supply get sold with the remaining tokens burned by the auction contract
- Be able to distribute the token minted to legitimate bidders at the end of the auction
- (bonus) add tests to demonstrate the auction contract is resistant to reentry attack.
    - What is reentry attack,
    - hands-on practice on reentry.

Tricky Points to Ponder

- How to enforce auction duration/countdown clock in blockchain?
- How to link/wrap your token contract with your auction contract?
- How to "burn" the unsold tokens?
- How to enforce successful bidder to pay Ether for the new token, (I.e., they can't cancel the bid) and how to refund bids that are invalid?

Option 3: Decentralized Domain Registrar

Project in a Nutshell

Build a “xxx.ntu” domain name registrar service on Ethereum testnet, using “commit-and-reveal” bidding process and a simple (minimally styled) front end website to interact with the blockchain. Supports listing of registered domains, query the actual address behind a domain and bid for an unregistered domain etc.

Background & Problem Statement

Domain name registrar like Google Domain, or GoDaddy.com or Namecheap.com are commonly used and seen whenever you want to register your own domain on the web. The purpose of this, apparently, is replacing hard-to-remember IP address with human-readable domain names, and relying on a DNS (domain name server) to resolve the actual IP address behind domain like “spotify.com” or “ntu.edu.sg”. Usually, one registrar will take care of one major domain (e.g., “.edu”, “.com”, “.sg”), and some companies have multiple registrars that take care of multiple domains.

Some problems with our current domain registrar are

- fixed pricing – you can only purchase at the price they offered, instead of determined by market
- Pre-registered by the vendors – When google launched their “.dev” domain services, they always will secure the more popular ones like “google.dev”, “flutter.dev” for themselves excluding everyone else from bidding for those domains even before it was open to public.

A decentralized domain registrar, in contrast, is open, transparent, fair and decentralized. The price for registration of a domain is determined by a “sealed-auction” during which anyone can bid and counterbid for a domain. The process is all transparent as smart contracts on the blockchain, and no central authority or commercial company controls the selling and registration of new domains.

The perfect inspiration is ENS (Ethereum name service) which lets everyone register their “.eth” domain on Ethereum.

Feature Requirement

For “.ntu” Domain Registrar, it should

- Allows any Ethereum address to bid for an unregistered domain of the form “xxxx.ntu”
- The recommended default logic of the bidding process is a blind auction. Which consists of two phase -- “Commit-and-Reveal”

In the commit phase, bidders commit to the price he/she is willing to pay for the domain, this commitment has to be hiding such that no one else is able to learn exactly how much was committed, it should also be binding such that later on the bidder couldn’t open it with a different value that he didn’t commit to.

The commit phase usually gives users a few days/weeks to bid, in our demo, we can accelerate that by setting the commit phase duration to X minutes. (X<3 preferably).

In the reveal phase, the bidders “reveal” or open the commitments and the smart contract will determine the validity of the opening of the commitments, as well as the result of the auction based on all successful bids.

- Be able to resolve a registered domain to the underlying Ethereum address
- (bonus) be able to reversely resolve an Ethereum address to any associated registered “.ntu” domain(s).

The simple front-end website, needs to show

- List and query registered domains
- Good bid and higher bid will win the domain
- Able to send to registered “XXX.ntu” some ether

Option 4: Prediction Market

Project in a Nutshell

Build a platform for users to create and participate in prediction markets. A simple (minimally styled) front-end webapp should allow users to view the list of open & closed markets, current bets and outcome probability, resolution date and all relevant info on the page.

Background & Problem Statement

The core idea of prediction market is simple. In most cases, no single person or institution can be the ultimate dictator of unpredictable future events (e.g., next president of United States, or will Google's evaluation exceed Apple's in 3 years). Therefore, instead of trusting a proclaimed authority, we "outsource" the prediction to the "wisdom of the crowd" by asking everyone to voice their prediction and put money on their prediction. Some might have higher confidence on certain outcomes, maybe due to some insider information or any type of information asymmetry – for these people, they might avoid leaking the informational advantage he/she has on public, but when it comes to betting and having the opportunity to win a lot money by making the right bet, they are incentivized to put money on where their heart truly believes.

If you read some studies in the Wikipedia page, there are numerous academic studies on the effectiveness (and also potential deficits) of prediction market, how it is usually more accurate than some brainy, elite research institutes.

Feature Requirement

Firstly, make sure you at least visit Augur and Omen website to see how their prediction market look like and roughly the flow. If you try to read Augur's FAQ or blog post, or Omen's blog post, they tend to be harder to understand, filled with terminologies and novel designs, as their inceptions are more than 3 years ago, and a lot have changed, updated, or eliminated since then.

The only hard requirements for this project are

- Users can open new topic (a new prediction market) with descriptions, outcome options to bet on, variable amount on each bet, resolution date & time, arbitrator identity (the trusted judge after the event occurs)

You can add on many more of your own design based on your understanding of how this prediction market would work better and provider prediction closer to truth. Some examples:

- (optional, bonus) ability to sell your bets and making it a financial derivative plus a marketplace where people can buy and sell bets on future events (business students are surely thinking about the definition of "Futures" right now, it's similar but not quite the same)
- (optional, bonus) allow betting using crypto assets other than Ether, for example, using Tether, or any other tokens here.
- (optional, bonus) count everyone's vote in weightage based on their reputation – the probability of an outcome is the sum of weighted bets. In another word, if a thinktank has a much better track record of predicting "political" related events, then his bets might have higher weightage even if his liquidity is the same as everyone else betting on other options. In order to do this, you will have to come up with a reputation system to assign everyone a reputation score (no black mirror style, thank you ...)

Leading Projects as Examples

- Augur (https://www.augur.net/)
- Omen from Gnosis Protocol (https://omen.eth.link/)

Option 5: Bring Your Own Projects

Note: You MUST get an approval from course coordinator about the project ideas on their difficulties, scopes, and other factors before proceeding with the development project.

- You will be required to submit a one-page write up on your project idea, similar to the other topics described in this document.
- The write-up should contain the background and problem statement, plus what your MVP will be in terms of feature requirements.
- The write-up may also contain your choice of bonuses/nice to have features.

Here are some ways to formulate your idea if you don't know where to start:

1. Read the DeFi SoK paper (first 3 chapters) and go to top DeFi projects' GitHub repo to find a feature request in their backlog that involves a reasonable amount of contract development.
   - Good example: Order-book based DEX, AMM based DEX type of tasks/features.
   - Bad example: Simple "increase testing coverage" or "fix a certain bug" tasks.

2. Watch Vitalik's "Things that matter outside DeFi" talk for more ideas like the following ones.
   - Censorship resistant, high-quality engagement social media – mostly mechanism design (read quadratic voting)
   - NFTs
   - Authenticate/attestation with Eth identity – self-sovereign & social recovery (Log in with Eth, see EAuth EIP)
   - Retroactive public good funding, (optimism's blog post)

3. Check Gitcoin bounties or hackathon to pick a topic with reasonable contract development.

Assessment Criteria for Development

You are expected to design and develop a decentralized application using tokens and smart contracts (most likely on Ethereum). This is an essential tool of the trade, and you will be exposed to the most practical hands-on aspects of blockchain development. This is worth 35% of your total score.

Criteria	Standards		
	Fail standard **(0-40 %)**	**Pass standard** **(41-74 %)**	**High standard** **(75-100 %)**
Identify the core problem, and design practical blockchain based solution. (LO 1, 3, 4)	Identifying completely wrong problem, and designing blockchain solutions that are somewhat related but are not the actual solutions expected for the problems.	Identifying the correct and relevant problem in line with the course materials, designing blockchain solution reasonably in line with solution expected for the problem, and trying to relate the course materials to the planned solution. Overall design can be further improved.	Identifying the correct and relevant problem in line with the course materials, designing technically accurate blockchain solution that is expected for the problem, and clearly connecting the course materials to the planned solution. Complete and practical design of solution.
Develop the blockchain based solution as per design in the relevant programming environment. (LO 3, 4)	Ad hoc development of the solution, with little or no connection with the core problem or the actual design. Little or no evidence of core competence in terms of programming such blockchain solutions. No or little evidence of critical evaluation of the proposed solution.	Logical development of the solution, in connection to the core problem and the actual design. Reasonable level of programming expertise demonstrated through the blockchain solution. Reasonable evidence of critical thinking related to the proposed solution, and some attempt at critical evaluation of the proposed solution. Attempt at ensuring efficiency, security, privacy, scalability of solution may be missing.	Logical development of the solution, in connection to the core problem and the actual design, connecting the fundamental principles of blockchain systems to the distributed application. Expertise in programming demonstrated through the blockchain solution, with clear evidence of critical thinking related to the proposed solution and its critical evaluation in terms of efficiency, security, privacy, and scalability.
Overall standard of the codebase, documentation and the user experience of the application. (LO 4)	Disorganised format and arrangement of the code, without any comment and little or no mention of relevant references/resources. Non-intuitive and unpleasant user experience with the final application.	Clear logical flow and well-formatted arrangement of the code, with all essential components. Reasonable comments and reasonable documentation of relevant references/resources. Decent user experience with the final application, with scope for improvement.	Clear logical flow and well-formatted arrangement of the code, with all essential components. Detailed set of comments in documentation to illustrate programming choices made towards the solution, to refer to relevant resources, and to complete the code professionally. Excellent user experience with a professionally built blockchain application.

Peer Evaluation within the Team (moderating factor)

The instructor cannot directly assess the contribution of each individual student to the completion of this team project. Hence, each student is required to rate the contribution of themselves and each of the other team members. All evaluations are held in confidence so no student will know how other teammates rate his/her contribution. You are to evaluate yourself and other team members fairly and objectively, bearing in mind the implications for the other members' marks (explained below). It is absolutely essential for you to submit your peer evaluation (both for yourself and your teammates) to get marks for your team project.

Scale of Ratings for Peer Evaluation

10-9	**8-7**	**6-4**	**3-1**	**0**
Demonstrated outstanding contributions and efforts during teamwork. Took a lead role and coordinated most of the work.	Exhibited equal and appropriate contributions and efforts during teamwork. Took some part in management and coordination.	Made sufficient contributions but a greater effort could have been made during teamwork. Little or no support in coordination.	Did not contribute much effort during teamwork. Did not contribute much towards the work. Little or no support in coordination.	Made no effort to contribute during the teamwork.

To factor your peer evaluations within the team (both for yourself and your teammates) into the final consolidated marks of this team-based learning assignment, the following computation will be used:

1. If a student receives a Mean rating of 8 or more, that student receives *100% of the team's marks*, as obtained based on the criteria and standards mentioned above for the development project.
2. If a student receives a Mean rating < 8 but >= 2, that student receives a specific percentage of the team's marks, as per the formula *20% of team's marks + Mean rating * 10% of team's marks*.
3. If a student receives a Mean rating less than 2, that individual *case will be further investigated* by your instructor and that specific student may finally end up receiving 0% of the team's marks.

Example: Assume the total marks for the team assignment is 100, and your team scores 70 out of 100.

- A student with an average rating of 8.5 gets 100% of 70, i.e., 70 out of 100.
- A student with an average rating of 6.5 gets (20% + 65%) = 85% of 70, i.e., 59.5 out of 100.
- A student with an average rating of 3.0 gets (20% + 30%) = 50% of 70, i.e., 35 out of 100.
- A student with an average rating of 1.8 gets the case investigated further by the instructor.

Your instructor reserves the right to review/revise each peer evaluation rating for your team in case of questionable circumstances, which include, but are not limited to, acts of bias, discrimination or malice.