No.	Phase	Steps
1	Stock Trading app (Explained very briefly. Good playlist for stock trading, but can't be adopted for energy)	Started working on the app
		Encountered errors due to outdated dependencies
		Solved them
		Looked for energy market data on the web
		Realised that data is either behind paywalls or not publicly available for security reasons(of trade data)
		Decided to put dummy data for simulation
		As the playlist went into extensive stock related concepts, I realised it would not be possible to replace stock with energy items without major refactoring
		Dropped it
2	Research phase (Explained somewhat briefly)	Looked out for other sources to follow: Youtube playlists, Github repos, etc.
		Didn't find any playlists but found some GitHub repos.
		Read their READMEs, Found that commodity market does not work like financial market.
		Also realised that most of these apps used Auction based mechanism for trading and blockchain for running these apps.
		They called these "P2P Energy trading"
		I picked an app to understand: https://github.com/victorphoenix3/P2P-energy-trading
3	App from GitHub (Explained briefly, skipped out some details. I thought a project made by 4 IIT girls would be on another level)	Downloaded the app, tried to follow the steps from README to run (Hoping it would work out)
		Well, the app started and i was able to see the frontend.
		But nothing really was working... Even unable to connect wallet, etc.
		Then I spent days trying to understand the app, how it worked and how it was organized.
		Realised that I needed to install Metamask, I did that. Still it wont work.
		Learnt about testnets, got API keys from Infura. Still wont work.
		Finally realised that we needed to have metamask and the app running in same browser window. Now I was able to connect wallet.
		But then what? Still nothing.
		Saw a bunch of contracts, tried learning solidity and blockchain. Followed this video: https://youtu.be/gyMwXuJrbJQ?si=5Q2Zl2z5VaKdoi18
		Stopped watchin the video after 7 hours as the video was 30 hours long and i thought 7 was enough.
		Learnt that these contracts need to be deployed somewhere, etc. So I learnt Ganache, Remix, etc.
		But, you see, the funny thing was that these contracts weren't even being referenced by another file! Indicating that the project itself was incomplete.
		Let's try to contact the devs then. Sent requests on LinkedIn, emailed the developer, even created an issue on GitHub. No response.
		After losing hope with these people, I tried digging more on the internet. Found a couple of reseach papers, but these were locked behind paywalls.

		So, I dropped it for good. And started working on it from scratch.
4	Problems deciding a platform (Wasted a lot of time here, wont waste more explaining)	Choosing on technologies: I thought that if I am making this app, might as well make it well. Decided to make it with Flutter.
		Started working on flutter based app, but encountered an error: Not enough support for blockchain yet!
		I reverted back to making a web application using React. Setup an app using Create React App (CRA)
5	energytrader (Glad I started working on this)	Implemented a basic wallet connection page first.
		Then I decided a database should be used for storing personal data of user, instead of storing it on the chain. So i chose firestore.
		Then I created the all purpose Account page in which users will be able to register, edit their details after registration and even delete their account
		After that, I started working on the logout like functionality of disconnecting wallet.
		At this point, the user can register, login and logout. What next? Contracts!
6	energytrader2 (Explained briefly, a lot of shocking times)	I replicated the energytrader project into energytrader2 and started working on contracts.
		At this point, I didn't realise how poorly built those contracts from that GitHub repo were and I thought perhaps I could make the UI and utilise these contracts.
		I understood these contracts. And.. Wait, What? These contracts are silly! It's just not realistic. Here's how they worked: Suppose we have 3 buyers with 1,2,3 unit requirement respectively. And there are 2 sellers with 2,3 units to sell respectively. Then these contracts aim to simulate very lightly that 5 units to sell 6 to buy, so sell 5 to buyers and give one from the grid. So buyers happy, sellers happy, THE END.
		Okay, so frontend and the contracts, both need to be developed from scratch.
		Again, I replicated the working parts of previous apps and set out on a journey to develop contracts.
		I quickly realised that what we need is something like a community grid model.
		I made the Community contract for user management and managing some of the hyperparameters of the app.
		Then I developed the Auction contract: I decided that we will have the following structure: 1. There will be an auction which can be started and stopped 2. Sellers will publish their listings for selling energy 3. Buyers will bid on these listings. Just like it happens in real world auctions. I also ensured some other things like closing all the deals when auction ends, closing a deal prematurely, withdrawing from auction, fund checking and a lot more.
		Then I created a contract for the community's grid, and called it GridInteraction. This contract facilitated the two things: 1. Buying energy from those sellers who didn't get any bidders. 2. Allowing needy buyers to purchase directly from the grid. After all, we are making it look like a community, so we should be able to help people in need, right?
		What about payments? Since we are using an Ethereum blockchain, we use Ethers, right?

7	energytrader3 : Contracts (Omitted most of the stuff, otherwise it would be too complex to understand)	Well, no. Why not create custom tokens? A separate currency for our platform. This sounds appealing. Users will want to hold on to our currency and the value of currency and the platform will be in sync, leading to mutual valuations.
		So, I created another contract called ElectricityToken which was our currency with the symbol of ELTK. This was developed using ERC20.
		And then, a question popped in my head. We are saying that the trade occurs after the auction, but how do we actually simulate a trade in the application?
		Then I decided to create certificates. I created a EnergyCertificate contract: So now, Sellers will first need to make a certificate for the amount of energy they wish to sell. The certificate can't be made for free though. It requires a 10% collateral. This ensures that sellers will most likely have the energy for which they are creating a listing, reducing chances of fraud. Also, the buyers will have something as a proof that the trade went through incase seller refuses to supply energy after the trade. For legal purposes, that is.
		Good progress, we made 5 contracts, a lot of work. Shall we deploy them now? NO! Because now we have a circular dependency problem. Several contracts refer to one another and require them to be deployed before deploying themselves. For example, contract A and B both reference each other. So A's constructor requires the address of B but B also requires the address of A. These address can only be retrieved after they are deployed. It's basically a deadlock.
		So we created an Admin contract which will be deployed first and other contracts will refer to this for internal uses and the community Admin account.
		Next, we made a script to deploy these contracts in order. We used hardhat for development and easy deployment of these contracts.
		Before running the script, we needed to carefully decide on the price of our ELTK, price of energy of the grid, etc.
		We did some calculations as below: We assumed that any trade will be of hundreds of units of energy, so we decided that 1 ELTK should be able to purchase roughly 100 units. Then we averaged a direct selling price of electricity to 5 INR. Then we calculated that 1 ELTK should be worth around 500 INR Then we referenced the price of 1 Ether and divided it by the 500 to get the worth of 1 Ether in terms of ELTK. We got around 588. Based on this data, we calculated that 1 ELTK should be around 0.0017 ETH. Likewise, the grid's energy price should be 0.01 ELTK as 1 ELTK represents roughly 100 units. And based on these values, we decided to keep 10,000 ELTK in circulation at the start.
8	energytrader3 : Servers (Servers are good, but took a hell lot of time generalising contract calls, which i did because i believe it is a part of the framework)	As need arised, we created the firebase-auth server for authentication with firebase. We then created the transaction-server from which all transactions must go through. Both these servers are built using Node.js
		The transaction-server's key role is to abstract contract function execution and to ensure that the running costs are not imposed on the user.
		To make the transaction-server work, we had to do major refactoring in our contracts.
9	energytrader3 : Frontend (I felt this is only important as a proof of concept implementation, so I spend most of my time refining	Coming back to the frontend, we utilised Redux for state management and also utilised session storage to manage stored data
		Then we also built a reusable notifier component, which can be used easily to show notifications in the app.
		Earlier, we implemented all the navigation controls in the header. We improved it to be more practical.
		Then we started creating pages. First, we created a page for token exchange. A place where users can acquire ELTK by paying in Ethers.
		Then, we developed the page where users can buy energy from the grid.
		Then, we worked on building a page for admin to control the auctions

contracts and the server.)	Then, we created a page for auctions itself.
	Later, we fixed the footer to stick to the bottom if content is less and be less annoying
	And still the development continues...