Team 5: Final Report

On The Block: An Ethereum Smart Contract & Decentralized Application for Rental Contracts

Section 1: Team member names, and a description (a list) of who-did-what in the project.

1. Genevieve Flynn: structured the project backend/frontend, front end CSS, front end API calls, database routing for users/wallets/contracts
2. Katie Bramlett: front-end CSS / HCI (for all pages; specifically About, Contracts, New Contract, & Settings + navbar), settings API calls, database design, slide deck presentations
3. Claire Furtick: settings and contracts back-end, new contract page functionality and Solidity smart contract
4. Ada Kilic: back-end SQL database, database design, settings

Section 2: A brief non-technical overview of your project (2 paragraphs).

On the Block, our Ethereum-based decentralized application, is a streamlined payment portal which can be used to set up timely rental payments. Real-world rental contracts often go through a third party intermediary to exchange payment from the tenant to the landlord, which leaves room for error with safety, security, and fairness. Built on Ethereum smart contract technology, On The Block offers numerous benefits to landlords and tenants, including traceability, security, efficiency, fairness, and transparency. With On The Block, landlords and tenants will be guaranteed a quick, safe, and fair contract. As a user of the application, they can access user-friendly and efficient features, such as starting a new contract, checking existing rental contracts, terminating existing rental contracts, and getting helpful notifications. Furthermore, they can utilize cryptocurrency payment methods as they become more widely popular, rather than traditional payment systems.

Fueled by the heightened demand for blockchain technology, the Ethereum smart contracts market is rapidly growing. It's easier than ever before to integrate blockchain technology with user-friendly applications. This applies beyond just On The Block. There are endless possibilities of where our technology could be used beyond rental agreements. Get smart with your rental contracts, and visit On The Block today.

Section 3: The link/URL to your project website, with a list of items on the website. This can be your GitHub page for example.

Website: https://katiebramlett.github.io/on-the-block/

1. Link to GitHub Code
2. Project Description
3. Our Team

4. Final Project Documentation
5. Additional Documentation

Code: https://github.com/katiebramlett/on-the-block

Section 4: A description of your project "environment" - a list of libraries, packages and APIs that you used for the project. This is not inclusive of your own code.

1. Libraries & Packages in Project Environment
    a. React
    b. Truffle
    c. Ganache
    d. MySQL
    e. Axios
    f. Web3
2. Dependencies listed in package.json

```
"dependencies": {
  "@fortawesome/fontawesome-svg-core": "^1.2.36",
  "@fortawesome/free-solid-svg-icons": "^5.15.4",
  "@fortawesome/react-fontawesome": "^0.1.17",
  "@testing-library/jest-dom": "^4.2.4",
  "@testing-library/react": "^9.4.0",
  "@testing-library/user-event": "^7.2.1",
  "@types/react": "^17.0.38",
  "axios": "^0.25.0",
  "bootstrap": "^5.1.3",
  "history": "^5.1.0",
  "react": "^16.12.0",
  "react-bootstrap": "^2.1.1",
  "react-dom": "^16.12.0",
  "react-router-dom": "^6.0.2",
  "react-scripts": "3.3.1",
  "save": "^2.4.0",
  "web3": "1.2.2"
},
```

Section 5: A brief technical overview of your project (2 paragraphs).

This application was created with a Truffle framework to integrate with the Ethereum blockchain. This gives us an outline for the structure of our development environment, our Ganache testing framework, and how to interact with the blockchain. The front end uses React as a framework to break our design into reusable components. By using React and Truffle packaged together, it was easier to integrate with our smart contract back end. To store user information and settings, we used a MySQL database. The database stores all user information, account settings, and contracts information for pending contracts. The dApp is designed to run

in a web browser (currently, an incognito browser as not to confuse with any existing Metamask wallets or non-test Ethereum networks).

In terms of our project environment layout, our front-end decentralized application (written in React.js with CSS) interacts with our database and the blockchain through two main components. First, there is our database API, which we use to post and get information from our MySQL database. This API is written in Node.js. Second, we have our smart contract, written in Solidity. This allows us to interact with the Ethereum blockchain, where we have currently utilized our personal test blockchain in Ganache.

Section 6: One "if I had to do this again" paragraph from each team member that contains technical lessons learned, and what would you have done differently if you know what you know now with the same project). Examples could include using a different API, some other type of restructuring etc.

Claire: If I had to do this again, I would have wanted to gain a better foundational understanding of React first before jumping into the project. I had no experience with React before the project and when determining how to set up and organize the components of our project and how they interact with each other, as well as when trying to solve errors we encountered, my lack of a comprehensive understanding of the structures and capabilities of React made these more difficult to troubleshoot. Also, instead of Ganache, I think it would have been cool to use Metamask and then get test Ethereum through the test nets that Metamask provides because then this would be easier to translate over to using a real Etherem wallet in Metamask.

Genny: If I were to do this again, I would spend a lot more time on the structure of the repository because I believe as we added new technologies, it made some folders bloated. Our front end used a mix of styling methodologies and in the future I would have only used one to avoid confusion. I would have focused more on integrating with Metamask in the beginning instead of relying on Ganache for testing. I learned from this project, not just about new technologies including React, Web3, Node, etc. but also about good conventions to follow and how to structure code in a scalable way for multiple people to interact with. Doing it again, it would be interesting to focus more on some other use cases as well, not just rental contracts.

Katie: If I were to do this project again, I would spend more time upfront laying out the design, scope, and implementation of the project. As we went along, we came up with new ideas, which sometimes led us to have to go back and redo our previous work to change direction (e.g., database design and schema, pages, front-end design, application features, etc.). If we had a more clear view of the whole picture in the beginning, I think it would have paid off in the long run, in time savings, logistics, and marching to one direction. This could also have helped us to better understand the interactions between each piece of our project, and maybe we could have gone further with our implementation. However, with a heavy workload of researching a new and

emerging technology and teaching ourselves as we went, I think we did as best as we could to be adaptable, flexible, and willing to learn; overall, I think we designed a great foundation for future rental contract applications.

Ada: If I were to do this project again, I would want to have a better knowledge of all of the elements we would need for the project. There were many parts such as learning React etc. that took a lot of time to learn and use. Additionally, having a better idea at the beginning of how all elements of the project would interact would have been useful since this took a lot of back and forth and error checking. Also it would be nice to explore smart contracts with other cases and situations as well. Although, we had to learn a lot considering we had to teach ourselves manys new technologies.

Section 7: Advice/Instructions for follow-on projects. Write your words of advice or instructions for future students who may choose to build on your project, including any ideas for follow up projects.

First of all, it is important to understand the advantages of blockchain technology and what kind of projects it makes sense to use blockchain technology for vs projects where it isn't necessary. Also, make sure you understand the limitations of blockchain technology and more specifically solidiy functionalities. For example, we originally planned on doing recurring payments and have the user submit one contract that dealt with the recurring payment call aspect within solidity, but we found that there was no timer functionality in solidity. Finally, it would be very helpful to have a good understanding of React to make using web3 within a React app easier.

Section 8: How to download and get the project working. If there's equipment, a description of where to purchase (what model # etc).

See the on-the-block repository README for Install How-To instructions.

Section 9: What works, what doesn't, what to be aware of (pitfalls, issues).

One thing that doesn't work is recurring payments. We first thought this was something that could be built into the smart contract in solidity, but it turns out we could not do that. Getting the contracts to resubmit a payment every month, maybe still on user click of a button that reminds them to pay this month's rent, is something we thought about for the future to solve this issue but never got to. Another thing that doesn't work is the notifications. There is only a notification for a new contract being created and the notification is not as nice as we were hoping for it to look.Also, a user is only able to have one wallet address connected to their account even though in reality a user may have multiple wallets. Finally, there are error checks missing here and there when it comes to making sure a user is inputting the correct type of input to a field and such.

Appendix I: Any other relevant documents such as licenses.

Appendix II: All homework assignments. No need to rework these. These are needed just for administrative reasons by the department and SEAS. Just reattach/re-put in the google folder, and list them here. These will not be graded again etc.

https://drive.google.com/drive/folders/136Ch4Trv9kPDD3Sc3z864l6bozBO2rY_?usp=sharing (shared with Amrinder, Elyse, & Chris)