**BlockCred**

by **The Misfits**

Arpit Masih | Simran Gogia | Utkarsh Sharma

**Theme: Blockchain**

**Simran [ To cover: Why? What? Features, and USP ]**

*[ Insert intro magic here ]*

Long gone are the days of issuing paper certificates, lining up outside the registrar's room for verification and putting a blindfold on while viewing the certificates of others.

While there exist several online certificate issuing platforms, such as those used by MOOC providers like Coursera, the existing systems are completely centralised; the issuer could revoke your certificate and no one would know.

On part of employers and academia, they have to trust on a closed system to believe that their candidate's certificates are legit.

We propose our solution, BlockCred, a completely transparent and decentralised credential issuing platform that solves all the problem plaguing the existing system.

Built on top of the Ethereum blockchain, BlockCred uses the Ethereum public ledger to keep track of credentials and publicly verify such credentials. Once issued, badges can't be revoked, even by the issuer, hence students and learners can focus on the things that matter and not worry about the legitimacy of their credentials.

And the good doesn’t stop here, BlockCred includes transactions and payment for certificates within the platform using cryptocurrencies. This means students can pay for the certificates they apply for using any ERC20 token of their choice, and institutions can target a truly global audience.

**Arpit [ To cover: User Interface, Scalability ]**

We started our bootstrapping with the basics. While centralised systems are plagued with security problems and a severe lack of transparency, they undoubtedly offer a seamless experience for the users. On the other hand, decentralised systems built on Blockchain, even with all their perks, haven’t been able to address the everyday user, especially in developing countries.

To counter this, we wanted to build a coherent user experience, where we take over all the technical jargon, and the user only has to focus on what matters the most for them; their credentials.

Using Adobe Illustrator, Adobe XD and core Human Computer Interaction guidelines, we have built a Web Application that puts the content at the centre and doesn’t overload users with the minor details.

In our React application, most of the components are natively coded with the minimal amount of static assets used. React’s component based architecture, our design methodology and Development pipeline allow for a truly scalable application, ready to take in multitudes of users from Day One. Even the badges we are using are not image assets imported into app, they’ve coded in a way that make them as easy as possible to change in the future.

**Utkarsh [ To cover: How? Architecture, Algorithms, No Bugs, Frameworks, Logic, Deployment, Development, Future Scope and Walkthrough ]**

I would be giving a live walkthrough of our application. The product we are using for this demo is NOT (emphasise) an Alpha or Beta build running on a localhost, it is a full-fledged Web Application **deployed** publicly using an automatic **DevOps** cycle with GitHub and Netlify, ready to take in users.

The key **frameworks** that we used are React for this application, Solidity programming language for the Smart Contracts, Truffle for Migration of the contracts to the blockchain and Ganache for testing on the localhost prior to this public deployment.

As my team mate mentioned, we start off with making it as simple to use. Users only need to have an Ethereum-based browser and a wallet to begin. Chrome and Metamask are two of most popular options, which are also completely free to use, which I would be using here, but users are open to using anything they want.

With a simple login, users are ready to issue, verify and claim credentials on BlockCred.

All edge cases are covered and accounted for in the **logic** of our Smart Contract design. In the development phase, we used the Remix IDE, Mocha test framework and the Chai Assertion Library to test our Smart Contracts to ensure that the users **don’t face any bugs**.

Starting with institutions, they have the ability to simply create a certificate with an associated crypto value for the badge and push it on to the public gallery for everyone to view. For the second option you can see here, if for example, they don’t want to include transactions or have a pre-set list of candidates that they directly want to issue the certificate, they can do use using the alternative creating method that we offer. I would be creating a demo certificate with my account here. After the transaction has been verified by the generous Ethereum miners and Proof of Work algorithms, the new certificate can be viewed by anyone.

Switching to a different account that I have here, I can see our newly created certificated in the public gallery, along with a list of the certificates that I already have in this account. The user can explore the list and apply for a Credential of their choice, sending a request to the issuer, and paying the associated amount. Notably, this amount that the user just payed isn’t transferred directly to the institution until the request is verified. For the meantime, it is associated with the certificate itself, and refunded in full if the request is declined.

Switching back to the issuing account, I can see the list of requests for my certificates. Only I, as the issuing party can take action on these requests, thanks to our Smart Contract design. I can approve the request, redeeming the associated crypto value and issuing the certificate to the user or decline the request and refund the amount to the student. After approval, if I switch back to the user account, I can see my new certificate in the list.

Going over to the verification section, I can enter the ID of the certificate I just earned and my public address to claim a digital badge that I can download and share on my socials. Notably, this digital badge has all the important information that someone might need to verify my credential.

If I take a look at the badge I just downloaded, I can observe the ID of the certificate and my public address. Switching to a third account, that has had no knowledge of this process till now, I can use these values to verify the certificate that was just issued. This emulates potential employers, teachers and people alike checking the verification of badges using BlockCred.

In the **architecture** of our application, we’ve configured the endpoints of the applications and the Smart Contract methods in a manner that ensures for an authenticated set of users, charges gas fee only in actions that require exchange of money.

We plan on building our own HD wallet to remove the single external dependency we have, Metamask.

**Simran [ To cover: Outro ]**

*[ Insert magic here ]*