

# CoinClock Documentation

## Introduction

“CoinClock is your decentralized solution for hassle-free attendance tracking. Say goodbye to outdated methods and manual record-keeping. With CoinClock, marking your attendance is as simple as a click of a button.”

### Overview

By leveraging the power of blockchain technology, CoinClock ensures secure and tamper-proof attendance tracking, providing a reliable and transparent system for both individuals and organizations. Users can easily mark their attendance with a single click, streamlining the process. CoinClock is designed to modernize attendance management, offering efficiency and accuracy in one comprehensive platform.

### Purpose

The purpose of CoinClock is to revolutionize attendance management by providing a decentralized solution that eliminates the inefficiencies and inaccuracies of traditional methods. Harnessing the potential of blockchain technology, CoinClock guarantees a secure, incorruptible, and transparent attendance tracking system for individuals and organizations. With its user-friendly interface and streamlined process, CoinClock aims to simplify attendance marking, ultimately improving efficiency and accuracy in attendance management.

### Technologies Used

* Blockchain - Solana
* Smart Contract - Anchor (Framework of Rust)
* Frontend - React and Tailwind CSS

## Getting Started

### Prerequisites

* Node v18.18.0 or higher.
* Rust v1.70.0 or higher.
* Anchor CLI 0.29.0 or higher.
* Solana CLI 1.17.0 or higher.

### Running the application

Follow the below steps to run the project-

Step 1 - Clone the repository using ‘*git clone <repo-url>*’ command.

Step 2 - Install Dependencies using ‘*npm install’* command.

Step 3 - Start the web app using *‘npm run dev*’ command.

## 

## Project Structure

### Anchor

This is a Solana program written in Rust using the Anchor framework.

Commands - You can use any normal anchor commands. Move to the anchor directory and run the anchor command or prefix the command with npm run, eg: *npm run anchor*.

Sync the program ID - Running this command will create a new keypair in the anchor/target/deploy directory, save the address to the Anchor config file, and update the declare\_id! macro in the ./src/lib.rs file of the program.

You will manually need to update the constant in anchor/lib/counter-exports.ts to match the new program ID using ‘*npm run anchor keys sync’* command

Build the program using ‘*npm run anchor-build*’

Start the test validator with the program deployed using ‘*npm run anchor-localnet*’

Run the tests using ‘*npm run anchor-test*’

Deploy to Devnet using ‘*npm run anchor deploy --provider.cluster devnet*’

### web

This is a React app that uses the Anchor-generated client to interact with the Solana program.

Commands-

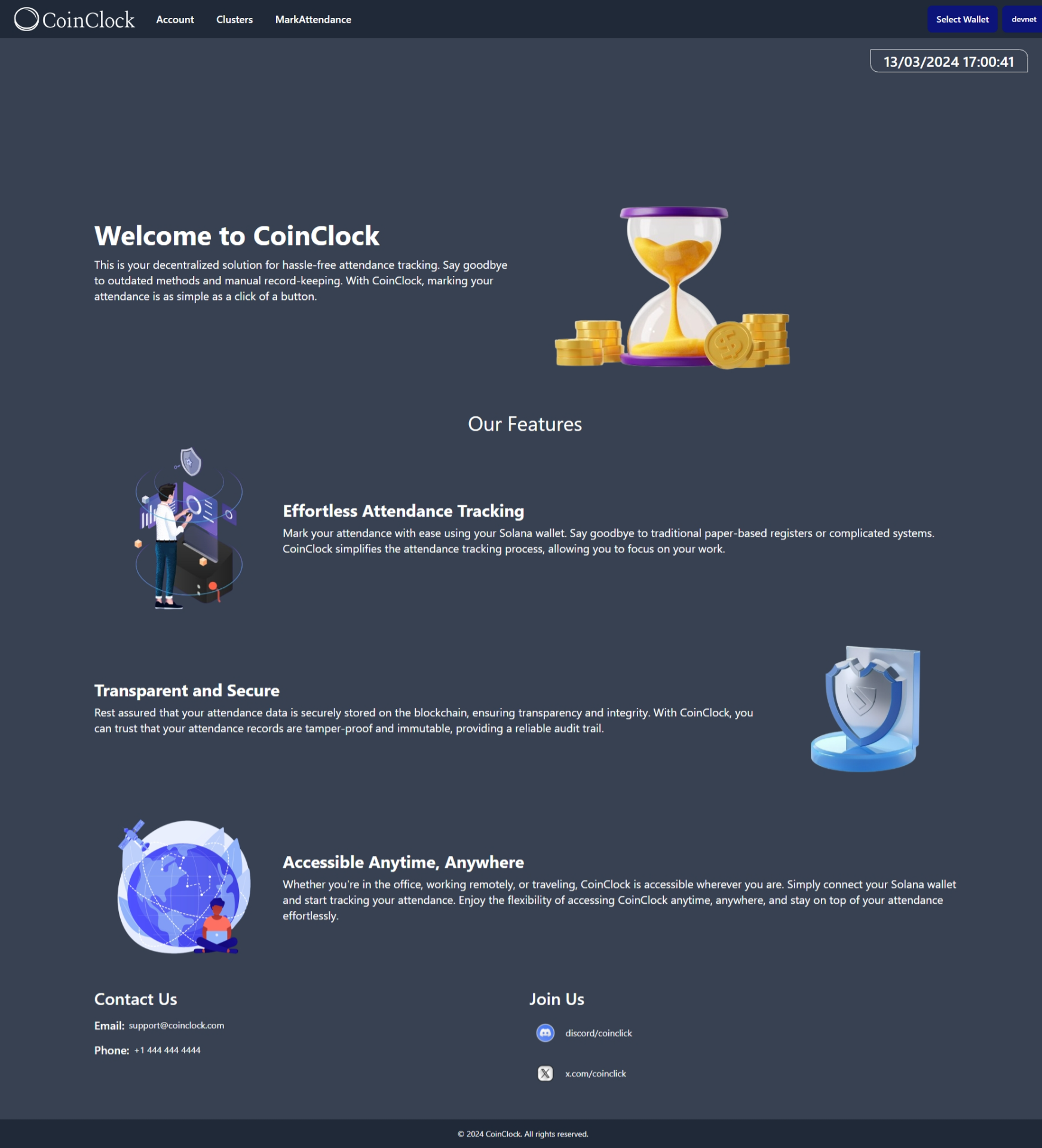
Start the web app using ‘*npm run dev’*

Build the web app using ‘npm run build’.

## Snapshots of features

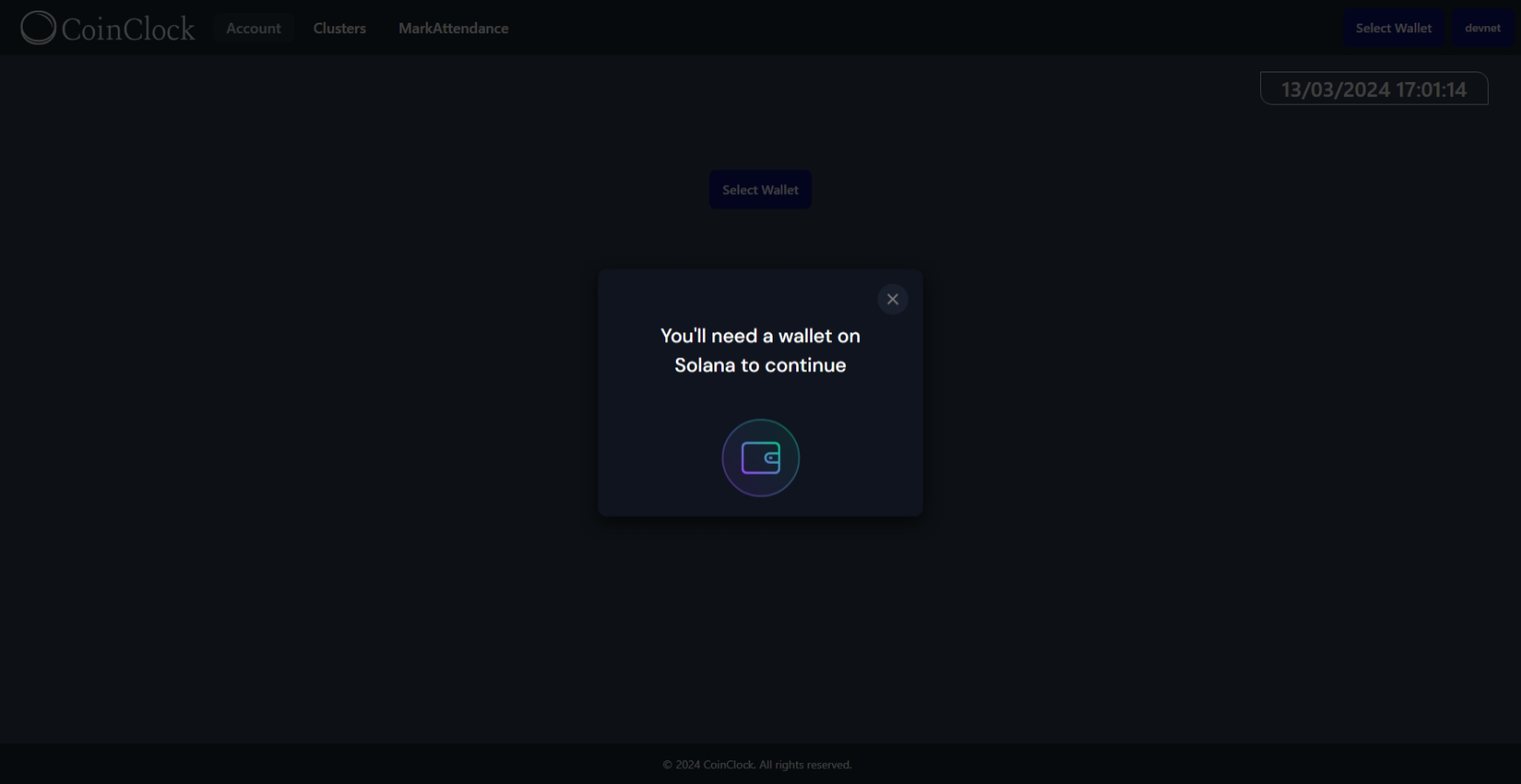
The following are the snapshots of the entire project-

### Home Page



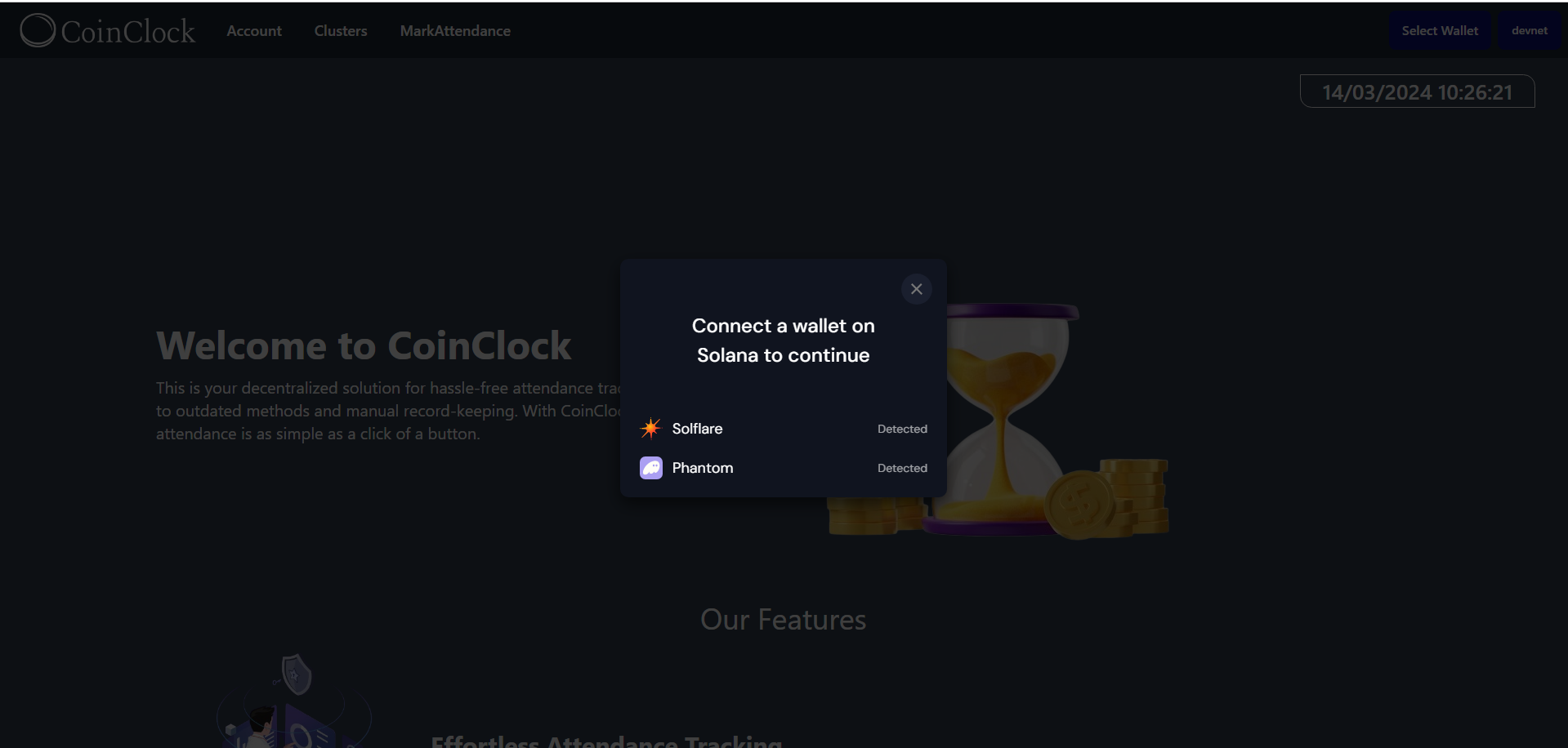
Here, on the homepage, we got some basic information about our application. And some functionality in the top header. To proceed, you will have to click on ‘select wallet’ present on the right side in the top header.

### Select wallet



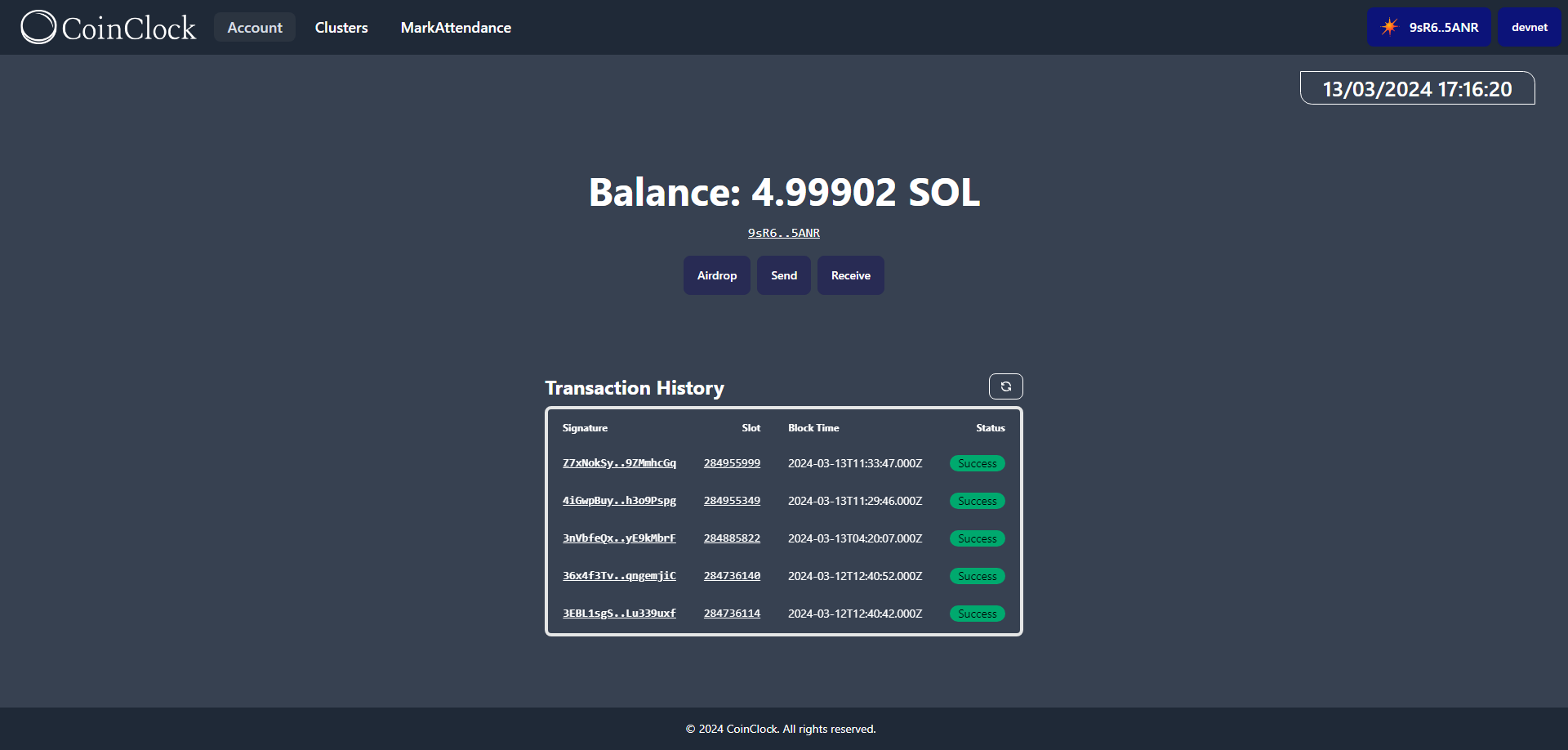
If you have any wallet that supports Solana installed in your browser, you will get a popup like the above. You will have to install a web wallet.

Once you have installed the web wallet, click again on the ‘Select Wallet button’ and you will get a popup informing you about the available wallets.



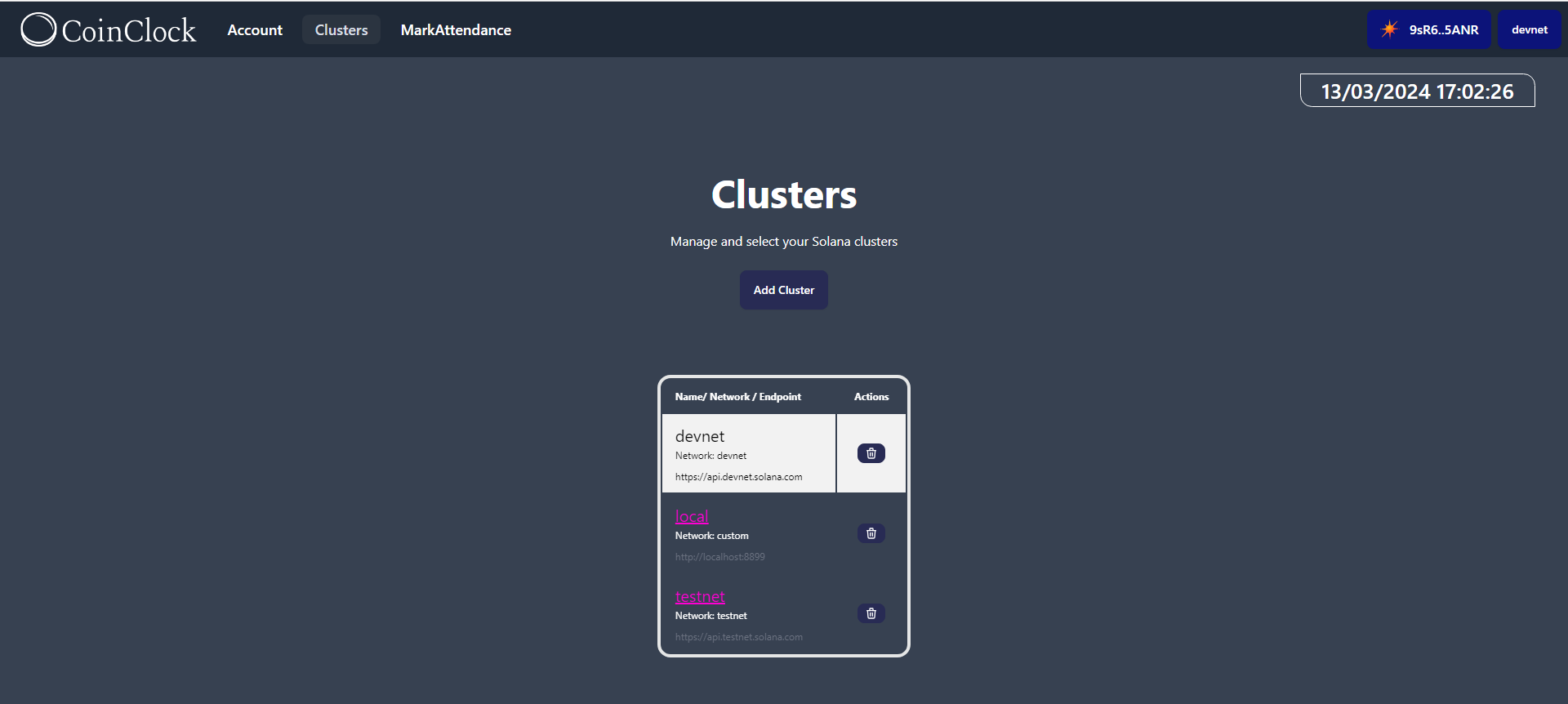
Select your desired wallet to continue.

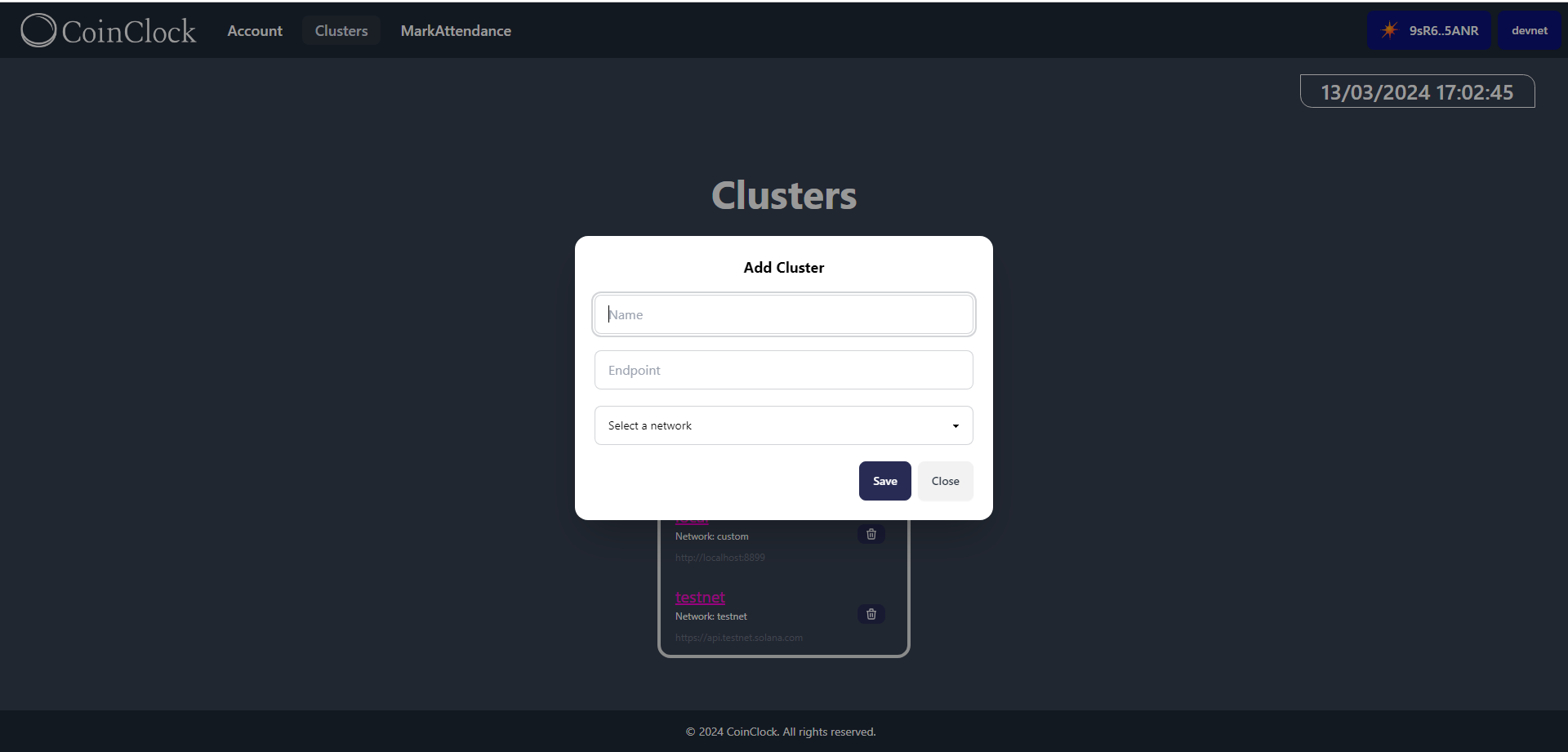
### Account



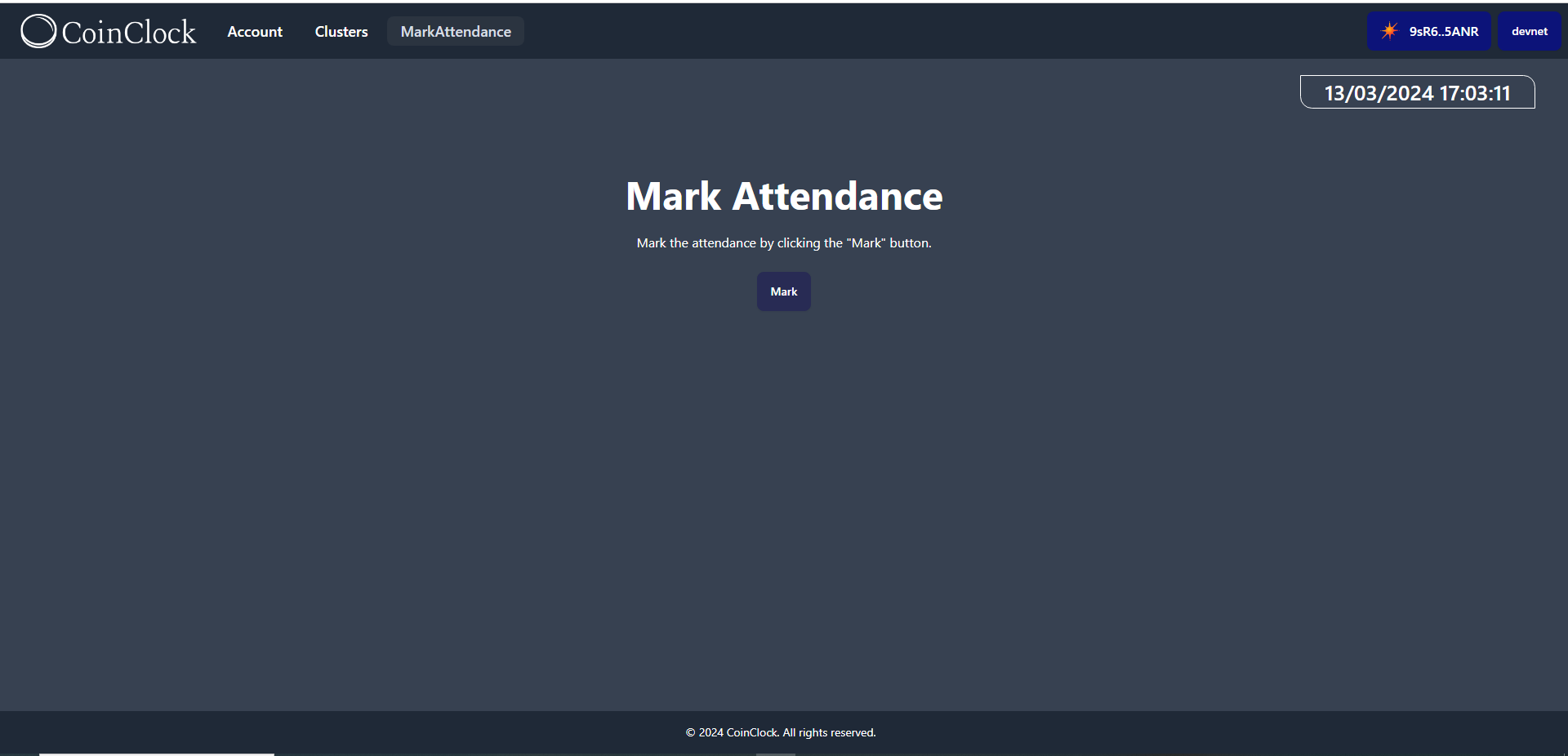
After connecting the wallet, click on the Account option on the top header, you will have a new page like the above, which shows the balance in your account, your account address (below balance), and transaction history for your account.

### Cluster





### Mark attendance



On this page, employees can mark their attendance for the day. Here, on marking the attendance, a transaction is made on the Solana for that date and time, which works as same as you clock in using some other CRM.

## Features of CoinClock

* **Solana Wallet Integration:** Users can securely log in to CoinClock using their Solana wallet, ensuring seamless authentication and data protection.
* **Attendance Marking:** With a simple click, users can mark their attendance, initiating a transaction that is securely recorded on the Solana blockchain.
* **Admin Panel (upcoming):** Administrators have access to an intuitive admin panel where they can view detailed attendance data. This includes user account addresses and timestamps for each attendance marking.
* **Transparent Attendance Tracking (upcoming):** CoinClock ensures transparency in attendance tracking through the use of blockchain technology, providing tamper-proof records that can be easily audited by administrators.

## 

## API Used

The following APIs are used in CoinClock:-

### Solana RPC Endpoint

In our project, we utilize the Solana Devnet API (<https://api.devnet.solana.com>) for interacting with the Solana blockchain during the development and testing phases. This API serves as a gateway to the Solana Devnet, a test network for developers to experiment with Solana blockchain functionality without using real SOL tokens. By leveraging this API, we can deploy smart contracts, interact with the blockchain, and test our application's functionality in a simulated environment.

### Solana Explorer

In our project, we utilize the Solana Explorer API (<https://explorer.solana.com/>) for accessing blockchain data and retrieving information related to transactions, smart contracts, and account addresses. This API serves as a crucial tool for interacting with the Solana blockchain, allowing us to securely retrieve and display real-time data within our application. By leveraging the Solana Explorer API, we ensure accurate and up-to-date information for our users' attendance tracking needs.

## Third-Party Frameworks/Libraries

The following third-party frameworks /libraries are used-

### Tailwind CSS

Tailwind CSS is a utility-first CSS framework that streamlines the process of styling web applications. Unlike traditional CSS frameworks that provide pre-designed components, Tailwind CSS focuses on providing low-level utility classes, allowing developers to create custom designs with ease. By utilizing utility classes for common CSS properties, Tailwind CSS offers a flexible and efficient approach to styling, resulting in cleaner code and faster development times.

### Create-solana-dapp package

[create-solana-app](https://github.com/solana-developers/create-solana-dapp) is a third-party framework that streamlines the development process for Solana-based decentralized applications (dApps). By providing a set of pre-configured templates and tooling, it simplifies the setup and deployment of Solana dApps, reducing development time and complexity. This framework abstracts away many of the low-level details of Solana development, allowing developers to focus on building and deploying their applications efficiently.

## Future Enhancements

The following are the enhancements that can be made to the project -

* **Bonus and leaves as NFTs:** Non-fungible tokens (NFTs) can be introduced to represent bonuses and leaves, providing a unique and secure way to manage rewards and time off within the system.
* **Integration with CRM:** CoinClock can be integrated into our existing CRM systems to incentivize and reward employees based on their attendance and project contributions. Also to generate detailed attendance reports, manage user roles and permissions, and export data for payroll or analytics purposes.
* **Utility Token Integration:** CoinClock can be integrated into our wallet application, enabling users to utilize tokens earned through attendance for various utility purposes within the ecosystem.

## License Information

CoinClock is licensed under the MIT License.

Permission is hereby granted, free of charge, to any person obtaining a copy

of this software and associated documentation files (the "Software"), to deal

in the Software without restriction, including without limitation the rights

to use, copy, modify, merge, publish, distribute, sublicense, and/or sell

copies of the Software, and to permit persons to whom the Software is

furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all

copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## Our Team

* Punar Dutt Rajput
* Rinki