

**PRESOLVE**



# **THE NORMIE-FRIENDLY DAO CREATOR**

By Sam Holmes, Namdar Mesri & bapic

# Table of Contents

## Resources

- Github Repository
- Live App URL
- Video Presentation

## Description

- Abstract
- Description

## Infrastructure

- Nitty-Gritty Details
- Technologies Used
- Use of Infrastructure
- Notably Hacky Methods Used
- Impressive Methods Utilized

## Technology Stack

- Ethereum Tools
- Testing Tools
- Blockchain Networks
- Programming Languages
- Web Frameworks
- Databases
- Design Tools
- Other Technologies, Libraries, Frameworks & Tools

# Resources


## Github Repository

<https://github.com/presolve-xyz/presolve>

## Live App URL

<https://presolve.xyz>

## Video Presentation

 [presolve.xyz Demo](#)

# Description

## Abstract

DAOs are underutilized due to a steep learning curve for non-technical users. Presolve solves this with a normie-friendly DAO creator that uses Magic.link for seamless sign-up and Open Zeppelin's standards for governance and execution of proposals.

## Description

Why aren't DAOs widely used? The learning curve for normies getting into DAOs is a massive roadblock.

Current coordination platforms rely on a third party for treasury management and decision-making, which creates a single point of failure. This leads to participants being unable to coordinate without being under the control of a central leader.

Within organizations of all sizes and functions, coordinating people to make a decision is a long and friction-filled process. DAOs provide a solution for these coordination and decision-making issues. However, onboarding web2 users into DAOs requires significant context on governance, smart contracts, and the technical procedures involved.

**Our solution is not education, it's abstraction. We present Presolve, a single-purpose DAO creator for Web2 users. Normies can now benefit from the decentralized decision-making capabilities provided by DAOs.**

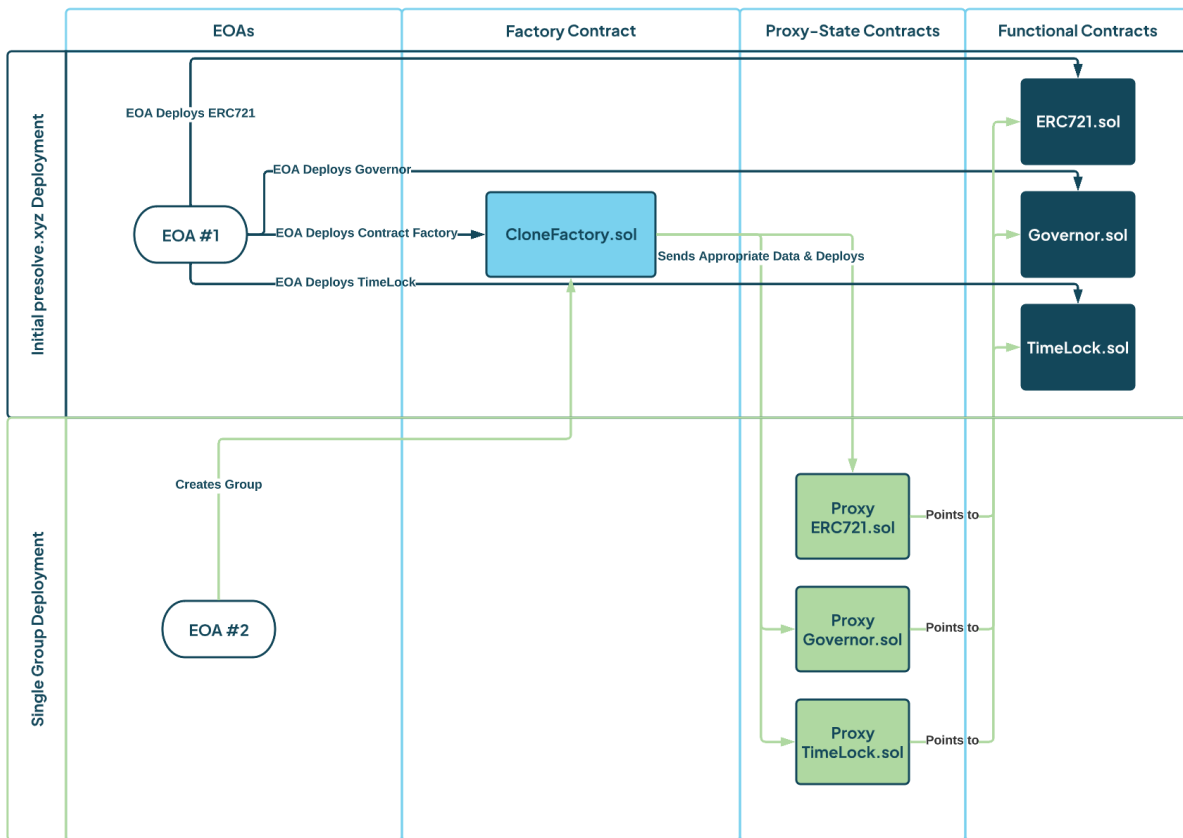
For example, your community can pool together funds to bring a music artist to your city for a one-time show. If the DAO is moving in the wrong direction, you can RageQuit to exit the DAO with your funds, relatively unscathed.

This solution applies to the potential of any coordinated effort, say a music concert. Fans that purchase tickets for an event can now drive the direction of how that concert is coordinated and can direct funds to where their interests lie, rather than the event organizers needing to guess what their attendees have a preference on.

Using Magic.link, web2 users can sign up seamlessly with email or Google Sign-in, linking them with a public address. This public address can then Deploy Group Governance, Timelock, and Token Proxy Contract sets using Open Zeppelin's Clone and Initializable Standards. Users can agree or disagree with proposals, termed 'Ideas' that are automatically executed once the majority has decided the outcome of the Idea.

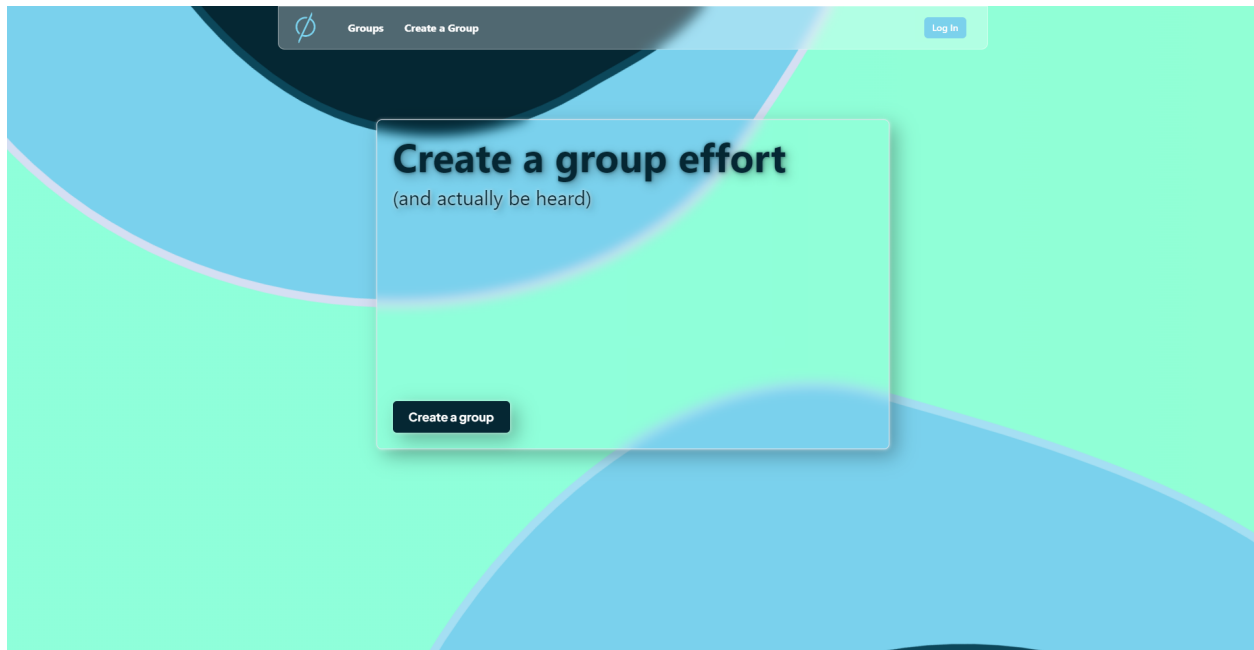
**With Presolve, we finally have Normie Friendly DAOs.**

# Infrastructure

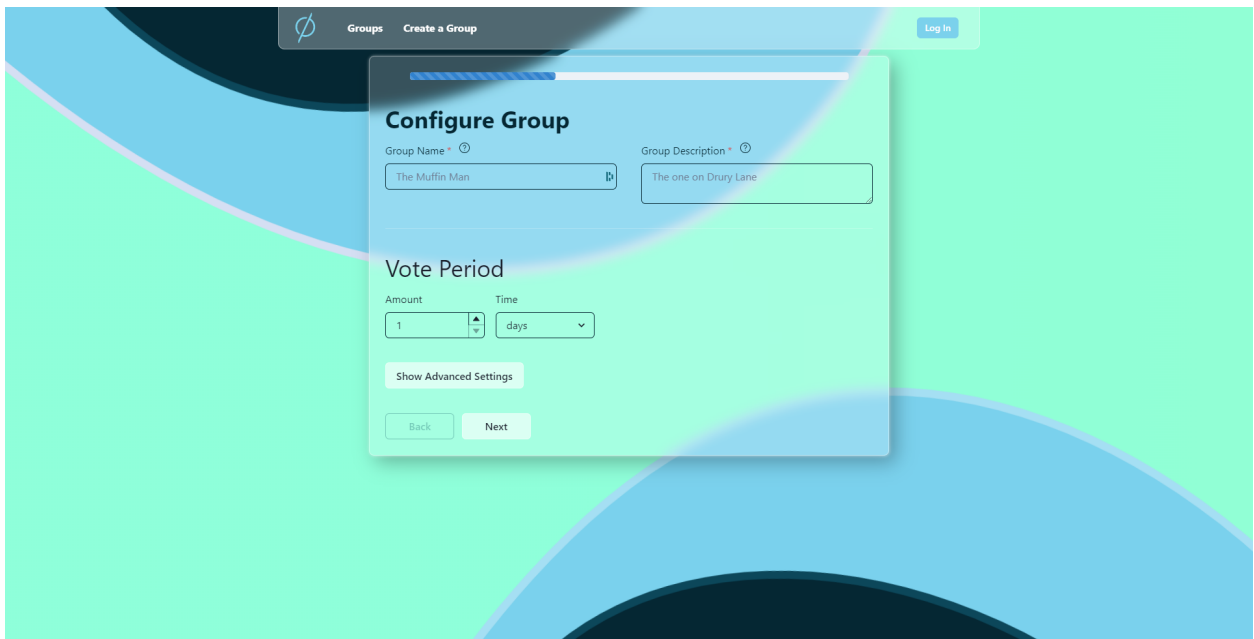


Smart Contract Infrastructure

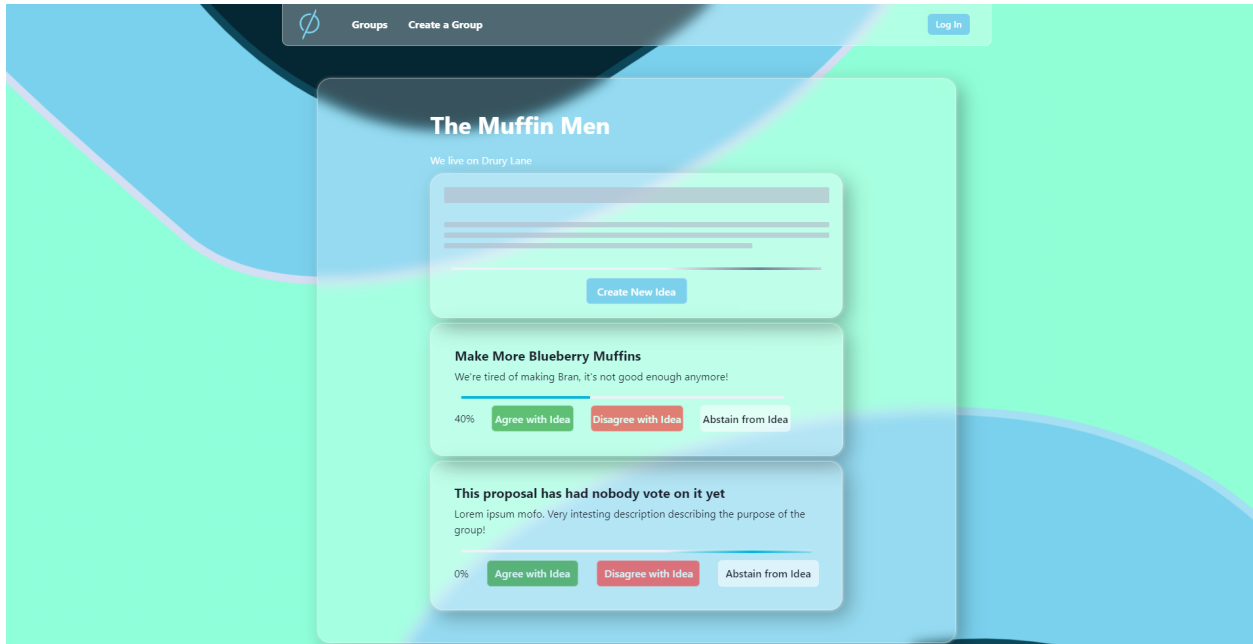
## Interface Screenshots



Home Page



Group Configuration



Group view populated with Proposals

## Nitty-Gritty Details

Our infrastructure serves as a minimal proxy factory, with the entry point starting at `OurCloneFactory.sol`, the minimal proxy contract. Within `OurCloneFactory`, we designed four primary functions, including `DeployGovernor`, `DeployTimelock`, `DeployERC721`, and `DeployDAO`. Each time a DAO is created, its information (such as Governor, Timelock, ERC-721, & Deployer Addresses) is wrapped in a struct and stored in an array. To understand `OurCloneFactory`'s functionality better, it's crucial to comprehend DAOs.

DAOs consist of three separate contracts that require each other's addresses to operate cohesively: the Governor, the Timelock, and either an ERC-20 or ERC-721 (in our case, we chose the ERC-721). The Governor is the contract most people interact with when they work with a DAO. It enables users to join, create a proposal, and establish the requirements for a proposal to pass. The Timelock contract holds the Governor's funds and gives users a specific amount of time to withdraw their investment if they're unhappy with the DAO's direction. The Timelock contract has two roles: Proposer and Executor roles.

The Proposer cues successful proposals for the Executor to execute. Generally, the Governor is the Proposer, and the Executor can be anyone, including the `0x0` address, allowing anyone to execute it. We made the Governor a Proposer, and the Executor the `0x0` address. The `0x0` address is a default value in OpenZeppelin's Governor Contract that enables anybody to execute on behalf of the DAO and is useful since we need this functionality. This would allow the owner to RageQuit, which is more user-friendly for members not happy with the DAOs trajectory, and since there is a period when proposals can't be executed, we don't have to worry about people trying to maliciously execute proposals faster, akin to a vote-race. Finally, the ERC-721 contract mints an NFT for each user who joins the DAO, and instantly delegates that NFT to themselves, giving them one vote per proposal.

These contracts don't have a Constructor; instead, they are utilizable and create contract objects that are individual contracts on the blockchain. We refer to these three contracts as Skeleton contracts. To make our infrastructure function correctly, we must deploy the Skeleton contracts first and send their addresses to `OurCloneFactory`.

In `OurCloneFactory`, each function (`createGov`, `createTimeLock`, `createERC721`) uses the addresses to generate a single instance of the Skeleton contract that it inherits. `CreateDAO()` wraps all of this into one function, enabling the user to only need to sign a single transaction.

For our front-end, we used Magic.link to help users onboard with a fiat ramp and wallet marker, which the user can utilize just using an email or Google Sign-On (or existing wallets like MetaMask, WalletConnect & Coinbase Wallet). Magic.link has an SDK that we used with TypeScript.



## Technologies Used

Figma, Adobe Illustrator, React, ChakraUI, Next.js, Vercel, TypeScript, ESLint, Prettier, Husky, Magic.link, Solidity, Polygon, OpenZeppelin's Governor, Timelock, ERC-721, Clone & Initializable Standards, Hardhat & Mocha/Chai

## Use of Infrastructure

The front-end consists of React TypeScript, using ChakraUI as a library, and Next.js as an SSR Framework, hosted on Vercel. ESLint & Prettier commanded by Husky helped keep our commits clean to ensure we did not have any compilation issues on Vercel or any commit issues while working as a team. Some of the UX direction and assets were made in Adobe Illustrator & Figma.

The data is hosted on the Polygon Layer 2, on the Ethereum blockchain, using multiple smart contracts.

## Notably Hacky Methods Used

- We used `patch-package` NPM module to ad-hoc patch the `@openzeppelin/contracts-upgradeable` contracts to allow for a custom owner address that isn't the message sender because it is used behind an initializer proxy contract.
- The dynamic background for the front-end was a keyframe animated .svg that was converted from native HTML into React using `dangerouslySetInnerHTML`. Maybe not the best method, but it worked.

## Impressive Methods Utilized

We deployed a minimal proxy factory to lower the memory size of our contract and reduce gas costs significantly for the user. Our entry point contract would point to our minimal proxy factory, `OurCloneFactory`.

# Technology Stack

## Ethereum Tools

Ethers.js, Hardhat, TypeChain,

## Testing Tools

Mocha/Chai

## Blockchain Networks

Polygon Layer 2, on top of the Ethereum Blockchain

## Programming Languages

HTML, CSS, JavaScript, TypeScript, JSX/TSX, Solidity

## Web Frameworks

React, ChakraUI, Next.js

## Databases

Aside from the Contract Factory Address Array on Polygon, we did not need a database! All data stored on-chain, w00t! 🎉

## Design Tools

Figma & Adobe Illustrator

## Other Technologies, Libraries, Frameworks & Tools

Magic SDK for interaction with Magic.link Connect, Ethereum Blockies for unique block avatar creation & Alchemy SDK for interaction with their libraries.