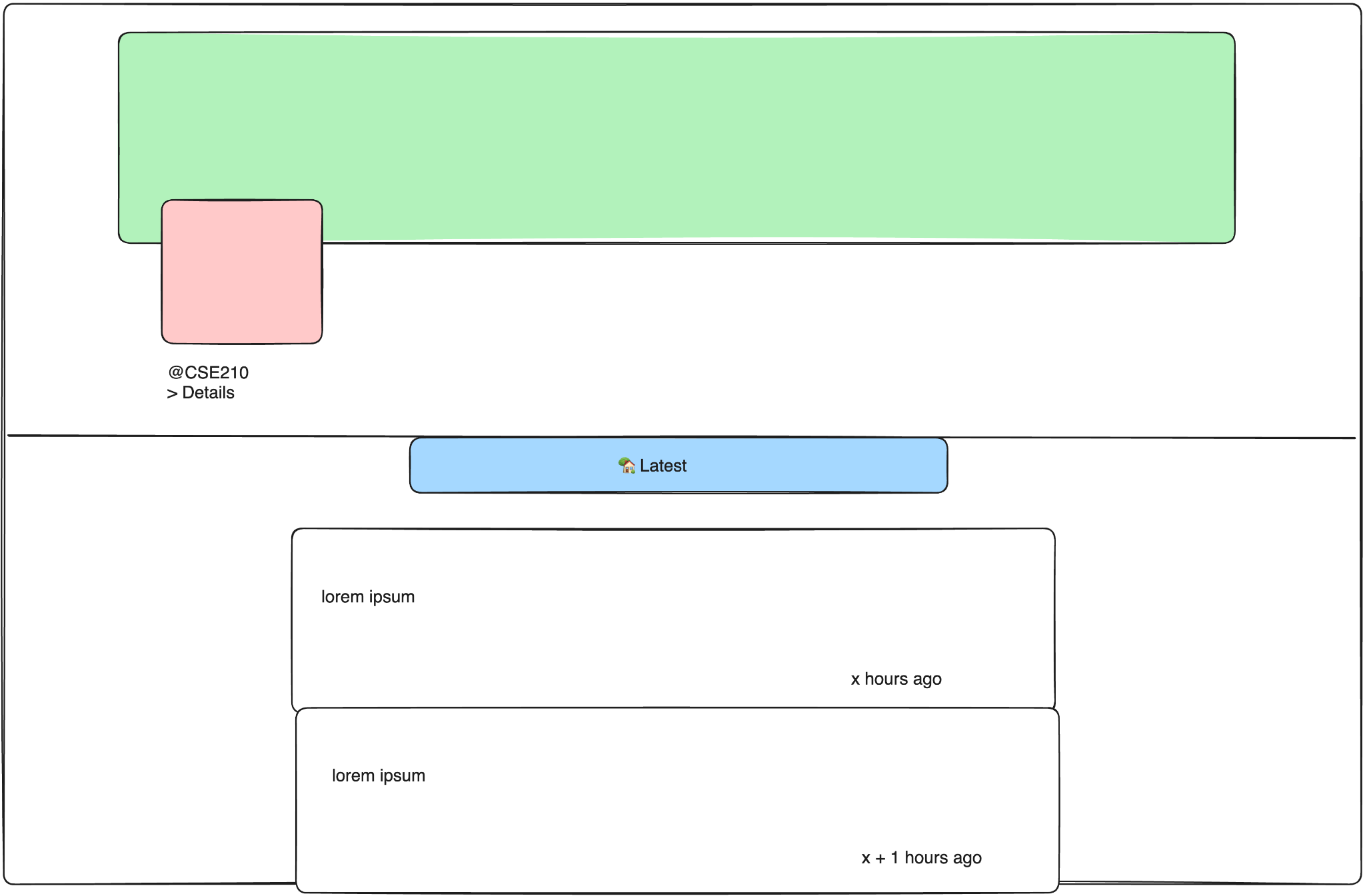
Architectural Decision Record

In the ever-evolving landscape of software development, the need for thoughtful and well-documented architectural decisions is paramount. This Architectural Decision Record (ADR) serves as a comprehensive repository, capturing the rationale, context, and implications of key choices made during the development process. By fostering transparency and providing a historical record, this document aims to empower the development team, stakeholders, and future contributors with a clear understanding of the architectural choices that shape our software systems.

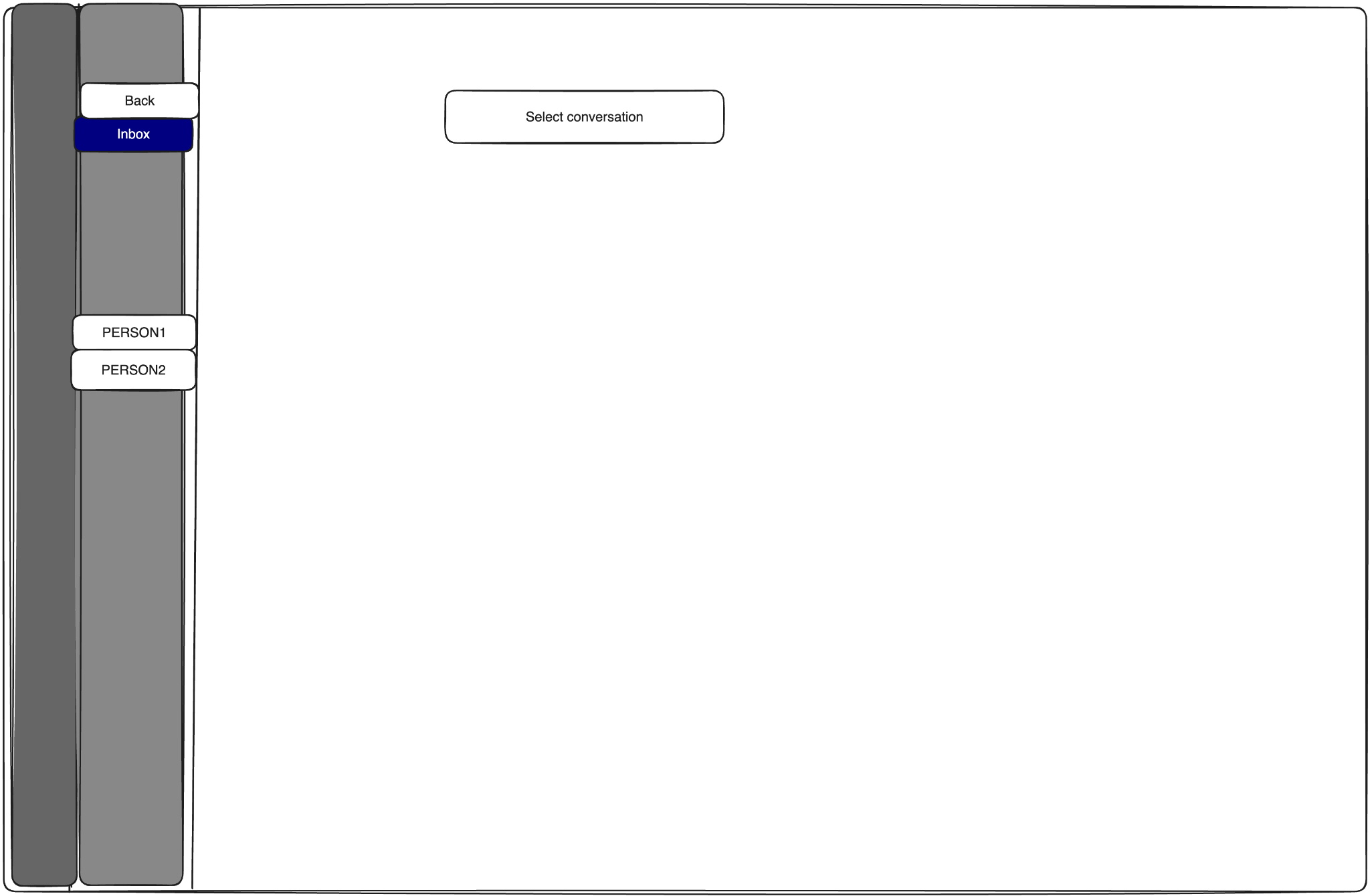
### UI/UX decisions

We decided to overhaul the social platform's user interface to boldly align with the sleek, contemporary aesthetics epitomized by Twitter. The visual changes, including adopting Twitter's signature white and blue color scheme, discontinuing niche customization features, and optimizing styles for mobile devices reflect careful consideration of current design principles and user preferences. Centering content applies principles of responsive web design to improve the browsing experience on small screens. Modularizing CSS adheres to best practices for maintainable code. Applying Helvetica font and updating components like buttons with on-trend styling create a fresh consistent look across the platform, while preserving functional behavior of all features. The following are a few of the wireframes we made while making the decision.

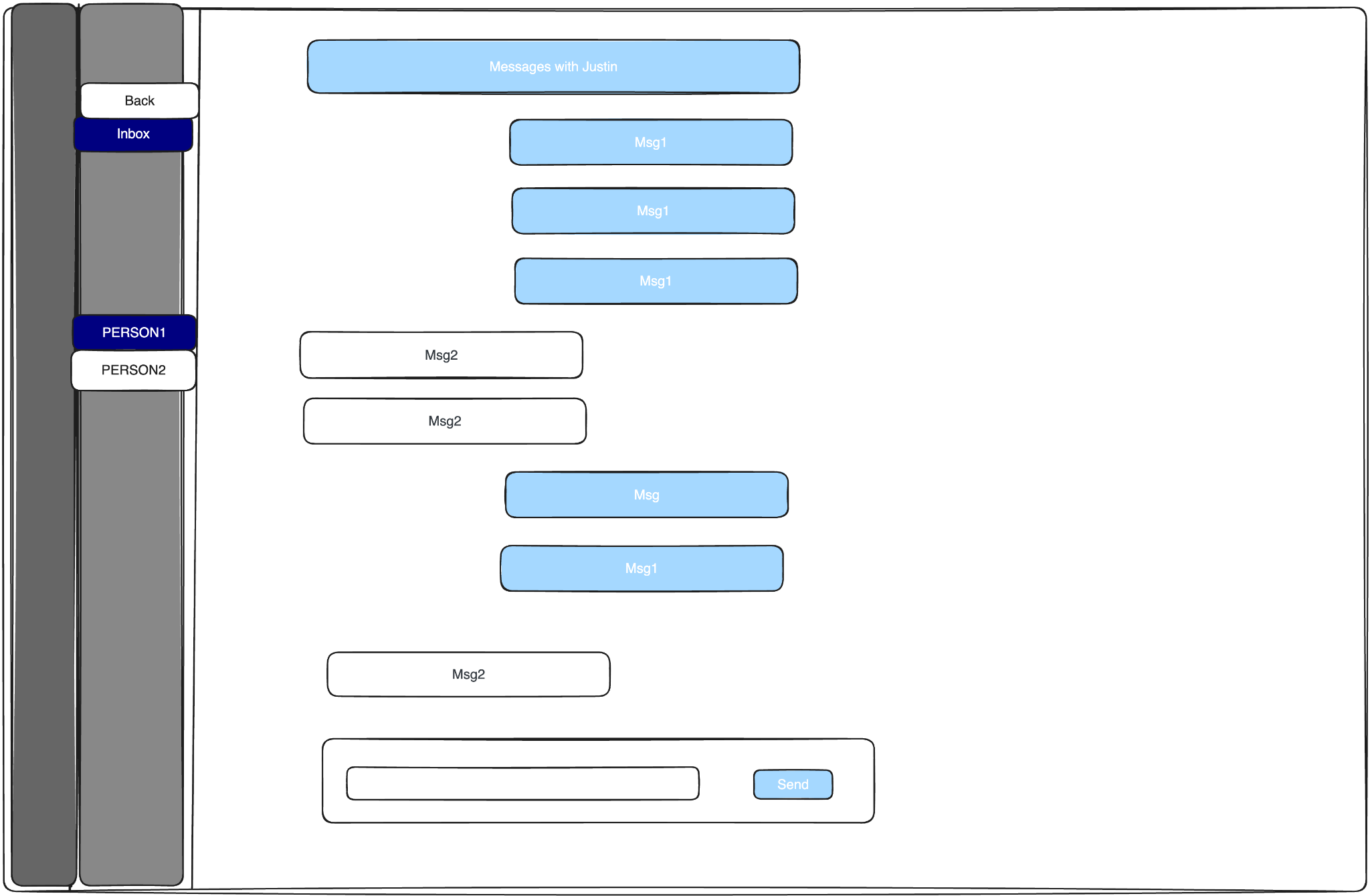
Index



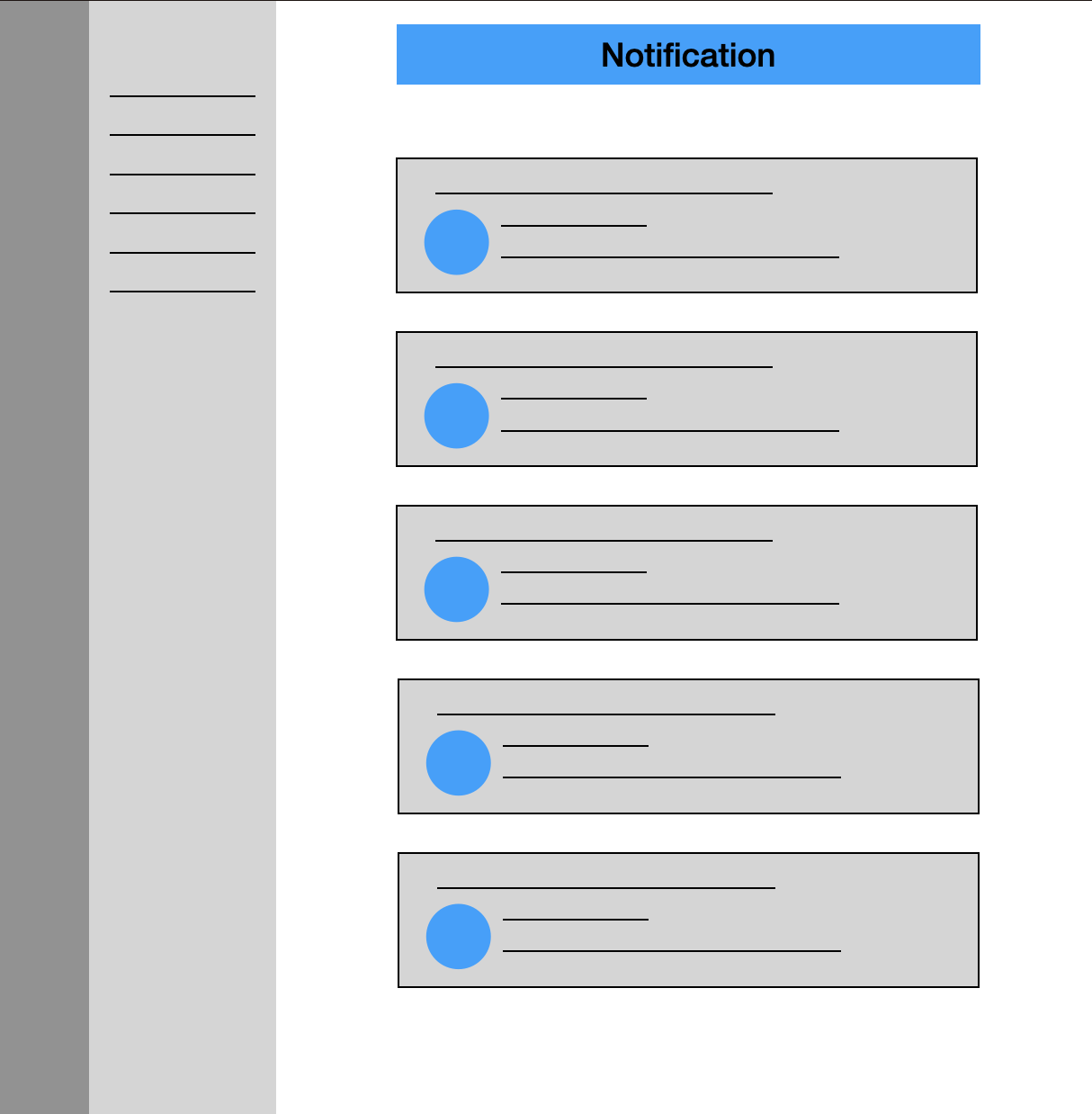
Message



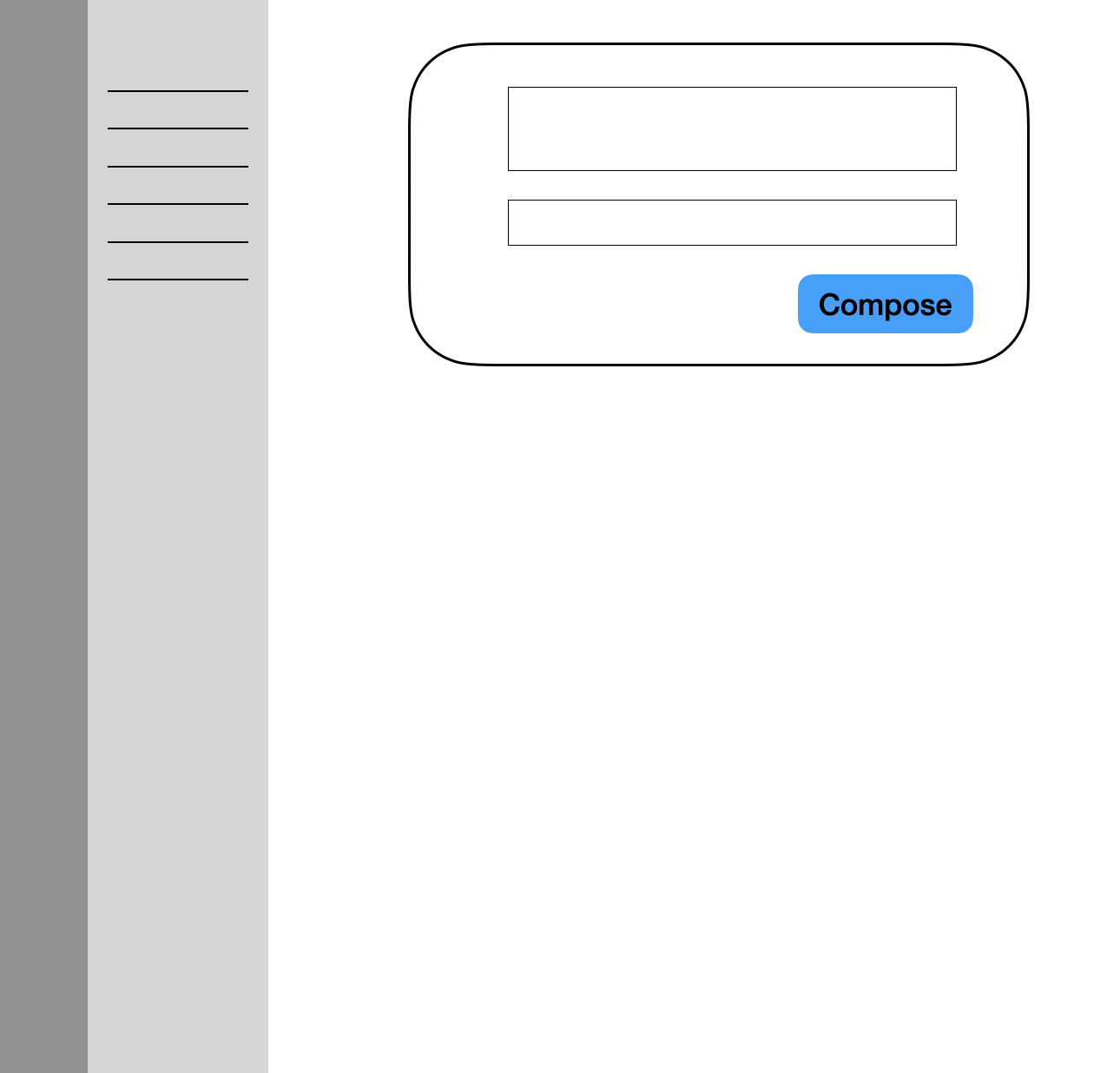
Send Message



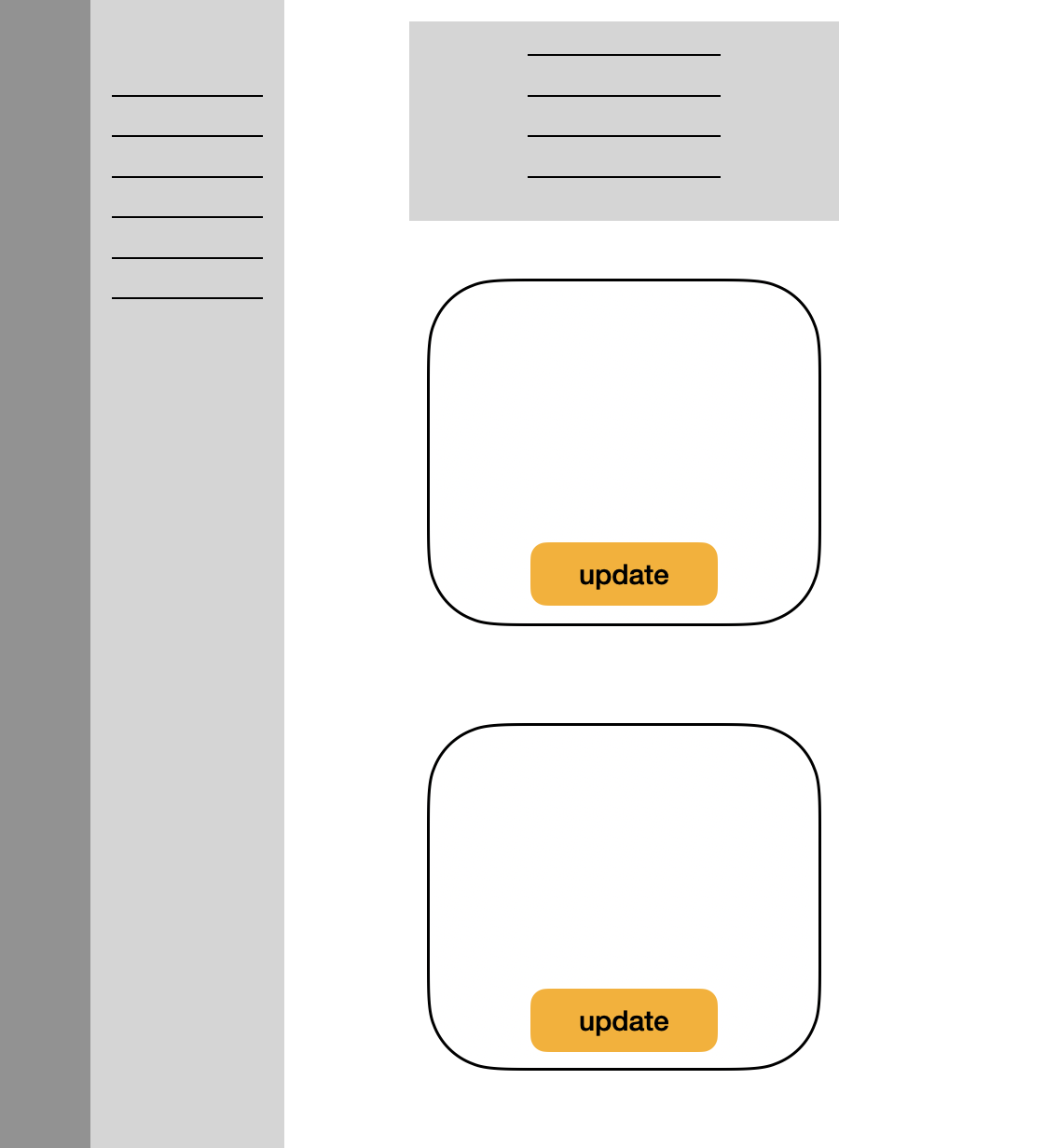
Notification



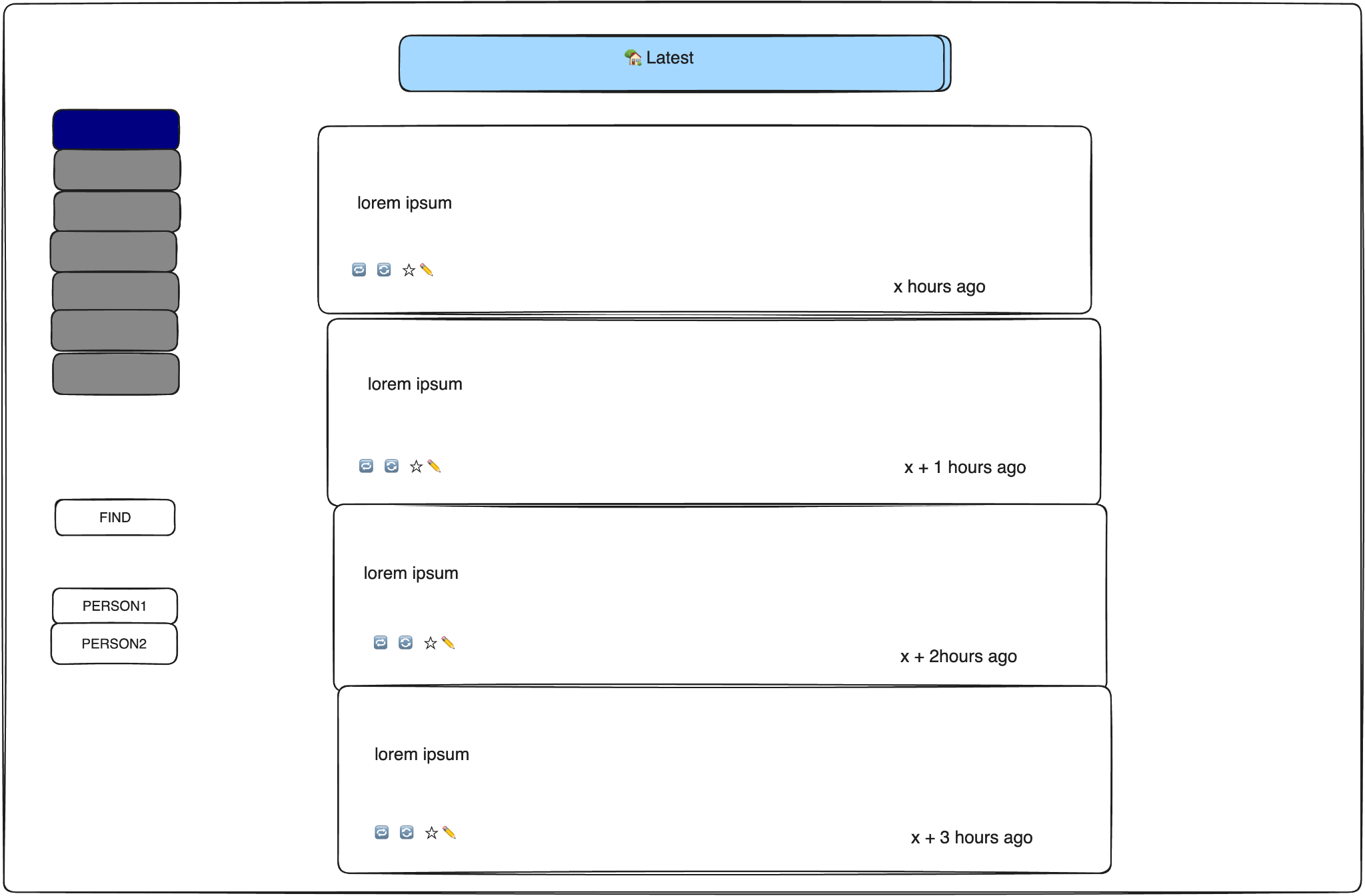
Compose



Preference

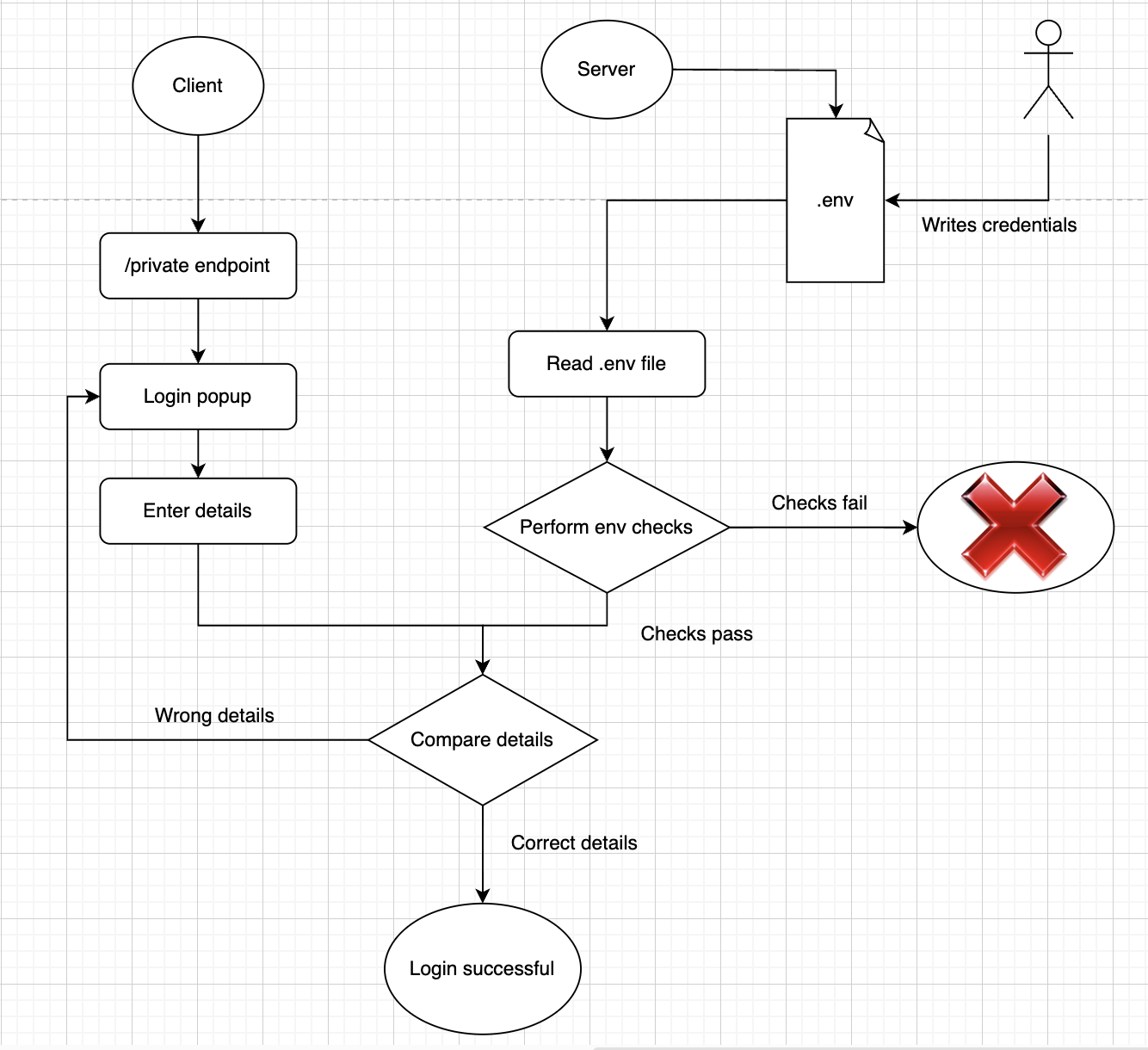


Private



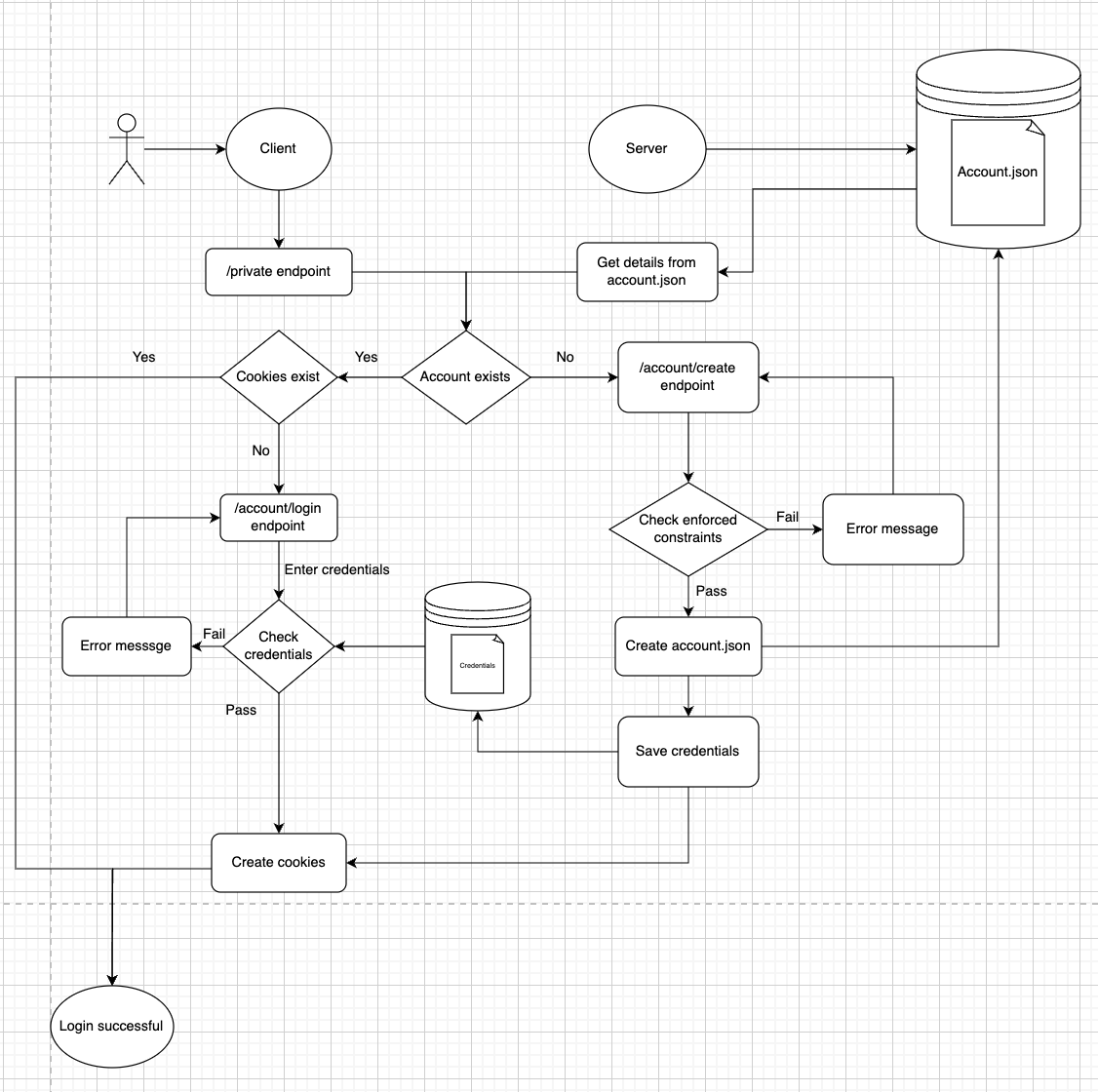
### Create account and login redesign

#### Original system design



The initial system relied on a 3rd party middleware “express-basic-auth” causing multiple code failures and decreasing flexibility. The create account and server booting module were also highly coupled. This was hampering our ability to handle the create user and login issue.

#### New System Design



We hence decided to completely remove the middleware and go back to making our code simpler and local. We designed a completely new architecture which gave us the ability to decouple the create user and server boot methods. We were also able to add new functionalities more with authentication.

#### Logout functionality

The new architecture helped us create a logout functionality to our system.

### 

### Testing framework

There are no tests for anything in the [original shuttlecraft project](https://github.com/benbrown/shuttlecraft). We decided that we needed to add tests to improve the reliability and maintainability of the project. The majority of the codebase is JavaScript, so we need to choose a JavaScript testing framework that would make it easier to incorporate tests into the codebase.

#### Jest

Jest is a very popular JavaScript testing framework developed by Facebook. One of its goals is to keep the test environment user-friendly. A main reason that we decided to choose Jest for our testing framework is that it is easy to set up and use. In addition, we had used it for our warmup assignment, so there is already less of a learning curve compared to the other testing frameworks; this was important because the timeframe we have for the project is very short. Some pros of Jest include built-in mocking capabilities, snapshot testing, parallel test execution, extensive documentation, and a large community. Some cons of Jest include its large size and integration issues.

#### Alternatives considered

One alternative JavaScript testing framework that we considered was Mocha. However, Mocha requires more manual configuration to set up compared to Jest. In addition, Mocha does not have a built-in assertion library.

Another alternative JavaScript testing framework that we considered was Jasmine. However, Jasmine also requires more manual configuration to set up compared to Jest. In addition, Jest has native support for snapshot testing while Jasmine does not.

### More Features We Add

Change Username

The old implementation does not allow you to change your username once you create an instance. Since the account information is stored in our server as the account.json file, we can simply add a new form in our preference page to set our new username and write the value in our json file. After doing that, the new user information is updated, and therefore the updated username can be seen in the public page and searched in the Fediverse by others.

Change Avatar

The old implementation does not allow you to change your default avatar. We find that the image is stored in an external website and loaded using links stored in account.json. Since we store all the information in json files in our server, for simplicity, we only allow the user to choose the avatar among several images we choose in the preference page. Uploading their own images will need extra storage format in our static storage folder or need us to set up a database. Instead, we pre-uploaded several avatars for the users to select. Once the avatar is changed, it can be seen in the public pages and the post feed by others.

Add Bio

The old implementation does not allow you to add your bio so that everyone can see and know about you better. Similarly as we did in changing username, we simply add a new form in our preference page to upload the bio. We add a new attribute in our account.json and rewrite it when we update our bio. This allows other users from the Fediverse to see our bio of the account created by using our Shuttlecraft instance.

Turn on/off Features

In many social media platforms, we have the choice to send a private post. Or sometimes we do not want other users to repost or comment on our post. We add this new feature in our shuttlecraft to let the users select whether others can reply/boost/like our posts. The post is created in our .data/posts folder using json files. We can add three attributes to each post: canReply, canBoost and canFave. Everytime we want to send a new post, we can choose the attribute to be true/false in the frontend using a checkbox implemented by html/handlebars. Then post it in the Fediverse so that no others can interact with it if you do not want them to. You can also edit your choice by clicking on the edit button below you post. With that, we can flexibly control the interactions with others via post.