# Vectre Labs

System Design Document

Prachyo Sarker
Henry Wong
Peter Chow
Zhao Ji Wang
John Pham
Nikhil Lakhwani

# Table Of Contents

# Frontend CRC Cards

| Landing Page | |
|---|---|
| Parent/Subclass: none | |
| Responsibilities: | Collaborators: |
| ● Provides all users with information regarding Vectre, a general overview and what users can expect when they decide to login and register. | ● Navbar |
| | ● User |

| Home Page | |
|---|---|
| Parent/Subclass: none | |
| Responsibilities: | Collaborators: |
| ● For the registered user, it allows the user to see all posts from users they follow or communities they join. | ● Navbar |
| | ● User |
| | ● Post |

| Login Page | |
|---|---|
| Parent: none | |
| Subclass: Register User Component | |
| Responsibilities: | Collaborators: |
| ● For the unregistered user, the login page allows for a streamlined way for the user to connect their Metamask Wallet and register with name, username and bio. | ● User |
| ● For the registered user, the login page allows for the user to connect their Metamask Wallet and return to their account. | ● Moderator |

| | |
|---|---|
| | |

| Register User Component | |
|---|---|
| Parent: Login Page<br>Subclass: none | |
| Responsibilities:<br><br>● For the unregistered user, they are able to register with name, username and bio. | Collaborators:<br><br>● User |

| User Settings Page | |
|---|---|
| Parent/Subclass: None | |
| Responsibilities:<br><br>● For the logged in user, they are able to change any user profile settings, customize notifications and privacy control. | Collaborators:<br><br>● Navbar<br>● User |

| Trending Communities Page | |
|---|---|
| Parent/Subclass: None | |
| Responsibilities:<br><br>● For the logged in user, they are able to explore different communities to be a part of from ones they have not seen / joined before. | Collaborators:<br><br>● Navbar<br>● User<br>● Moderator<br>● Community |

| User Profile Page | |
|---|---|
| Parent: None<br>Subclass: User NFT Dashboard | |
| Responsibilities:<br><br>● For a registered user to view and edit his/her profile information such as username and bio. | Collaborators:<br><br>● Navbar<br>● User |

| Community Profile Page | |
|---|---|
| Parent/Subclass: None | |
| Responsibilities:<br><br>● For the moderator, the community profile page acts as the hub for their community, as well as a gateway to create proposals for their community members to vote on.<br><br>● For the community member, they are able to vote on proposals and create posts.<br><br>● For the user who has not joined the community, they are able to see posts from community members. | Collaborators:<br><br>● Navbar<br>● User<br>● Community |

| Navbar | |
|---|---|
| Parent/Subclass: None | |
| Responsibilities:<br><br>● It allows the registered user to | Collaborators:<br><br>● Community Profile Page |

| | |
|---|---|
| navigate between different pages and perform actions such as logout and | ● User Profile Page<br>● Trending Communities Page<br>● User Settings Page<br>● Login Page<br>● Home Page |

| User NFT Dashboard | |
|---|---|
| Parent Class: User Profile Page<br>Subclass: none | |
| Responsibilities:<br><br>● For the user whose dashboard it is, they are able to showcase their NFTs on their profile effectively.<br><br>● For other users, they are able to interact on the user's dashboard. | Collaborators:<br><br>● User |

## Backend CRC Cards

| User | |
|---|---|
| Subclass: Moderator | |
| Responsibilities:<br><br>● Provides a model for the users of the app and defines how each User node will be formatted in the Neo4J DB<br><br>● Provides functionalities for creating/updating/retrieving/deleting User nodes in the Neo4J DB | Collaborators:<br><br>● Neo4J Database |

| Post | |
|---|---|
| Parent/Subclass: none | |
| Responsibilities:<br><br>● Provides a model for post content users will create and interact with. Defines how the Post node will be formatted in the Neo4J DB<br><br>● Provides functionalities for creating/updating/retrieving/deleting Post nodes in the Neo4J DB<br><br>● Provides functionality for retrieving specific posts to display on user's personal following feed | Collaborators:<br><br>● Neo4J Database |

| Neo4J Database | |
|---|---|
| Parent/Subclass: none | |
| Responsibilities:<br><br>● Provides utilities for interacting with the graph database that stores all information relating to the application (i.e. all users, posts, etc…) | Collaborators:<br><br>● None |

| Router |
|---|
| Parent/Subclass: none |

| | |
|---|---|
| Responsibilities:<br><br>● Redirects API endpoints to the correct functions so that they can be executed | Collaborators:<br><br>● User<br>● Post<br>● Community<br>● Moderator<br>● Comment |

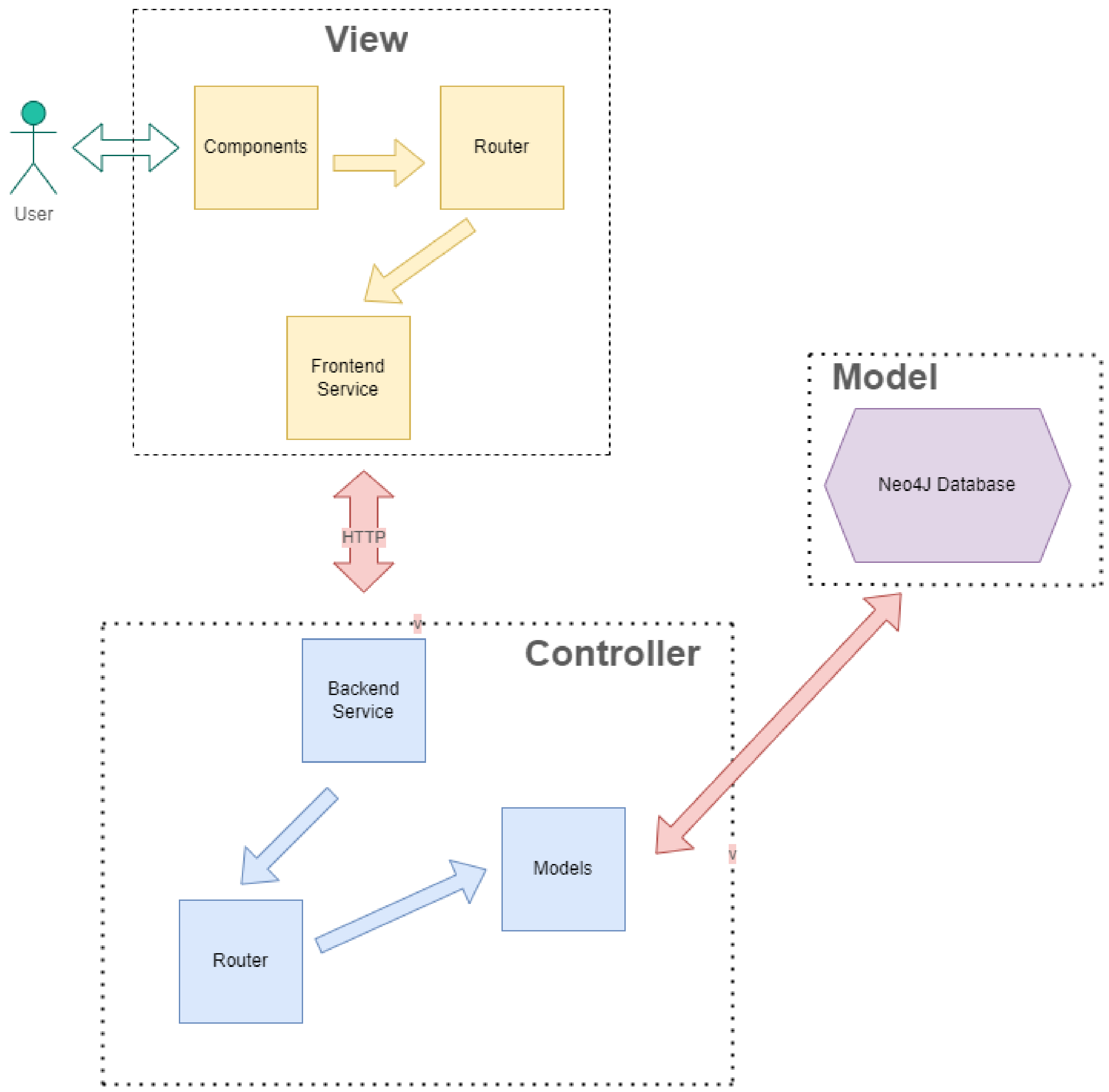
| Moderator | |
|---|---|
| Parent: User | |
| Responsibilities:<br><br>● Provides a model for moderators of communities on the app. Defines how the Moderator node will be formatted in the Neo4J database.<br><br>● Provides utilities for moderator-specific actions such as deleting posts, kicking members, etc.. in communities | Collaborators:<br><br>● Neo4J Database<br>● Community |

| Comment | |
|---|---|
| Subclass: Post | |
| Responsibilities:<br><br>● Provides a model for comments that users can leave under posts. Defines how the Comment node will be formatted in the Neo4J database.<br><br>● Provides utility functions for interacting with the Comment node in the Neo4J database (creating/deleting/retrieving/updating nodes and their properties) | Collaborators:<br><br>● Neo4J Database |

| Community | |
|---|---|
| Parent/Subclass: none | |
| Responsibilities:<br>● Provides a model for the communities users will be able to join based off their MetaMask wallet contents. Defines how the Community node will be formatted in the Neo4J database<br>● Provides utilities for interacting with the Community node in the Neo4J database (such as creating/deleting/retrieving/updating communities and their properties) | Collaborators:<br>● Neo4J Database |

| Backend App | |
|---|---|
| Parent/Subclass: none | |
| Responsibilities:<br>● Is the backend server that hosts the APIs exposed to the frontend. | Collaborators:<br>● Router |

## Software Architecture



## System Decomposition

The entire system is decoupled between the frontend and the backend. The frontend is the application that the user will be interacting with. The frontend heavily uses React to display the pages users navigate to (which are implemented as React components). Chakra UI was utilized to help with styling and for creating the components used for the frontend. Redux was also utilized as a state management tool for transitioning between different states, and Redux Saga was used to create API calls to the backend.

For a lot of the user interactions within the app, data is required to be transferred in between the frontend and the backend. For example, when a user edits their profile, the changes to the profile are sent to the backend so the database can be updated. This is handled through the use of HTTP endpoints exposed by the backend. The frontend is able to create/update models in the backend by using POST/PUT requests and are able to retrieve information from the backend using GET requests.

The backend app is a web server running on Node.js that uses the Express framework to host the APIs that the frontend will be calling and interacting with. The backend uses Express' Router object to call individual methods and execute the requests. These methods come from individual classes in the backend (i.e. POST requests to register a new user calls the constructor method for the User object). These classes interact with the Neo4J Database object that interacts with the Neo4J database and performs read/write operations to make changes to the data.