BC-HUB System Design

Faraz Shaikh
Ryan Laporte
Hongting Li
Kesavar Kabilar
Gan Hao Cheng, Keith
Sahil Malek
Nathan Ralph

CRC Card:

Card1

Component: NavBar
Style: \frontend\src\components\NavBar\NavBar.css

Responsibilities: UI for the webpage, link the 4 major sections of Market, Learn, Community and News. Corresponding in the routing of the index.js	**Collaborator**:

Card2

Page: Education
Style: \frontend\src\components\Pages\Educational\EducationPage.css

Responsibilities: Display relevant information for users to learn digital currency at beginner, intermediate and advanced level.	**Collaborator**: EducationPage NavBar

Card3

Page: Community
Main Component: \frontend\src\components\Pages\Community\Community.js **Child Components:** ● Feed ○ FeedCard ● CommunitySidebar **Style**: \frontend\src\components\Pages\Community\community.css **Utilities:** ShowMoreText **Hooks:** useFetch **HTTP Requests to:** backend\routes\community.js, backend\routes\auth.js

Responsibilities: Presentation of the Community feeds of our app. Requests Middle tier server for data through GET requests pointed to our HTTP Endpoints.	**Collaborators**: ● Feed ● FeedCard ● CommunitySidebar. ● NavBar

	• Routes\community.js SearchBar

Card4

Component: CommunitySidebar
Style: \frontend\src\components\Pages\Community\CommunitySidebar.css

Responsibilities: Sidebar component that is displayed along with the feed. The sidebar has buttons to navigate through the different feeds easily, as well as create a new post.	**Collaborators**: • Community • MakePost

Card5

Component: GrabNews
HTTP Requests to: backend\routes\newsfeed.js

Responsibilities: GET request to backend for news articles that are then sent to the Newsfeed component	**Collaborators**: Newsfeed

Card6

Component: Newsfeed

Responsibilities: Utitlizes data received from the GrabNews component to display articles using the News component. Implements pagination for articles. Requests for other pages of articles are made from this component via the GrabNews component. Displays the search bar and filters results based on the user's search query received from NewsSearchBar	**Collaborator**s: News GrabNews NewsSearchBar NavBar

Card7

Component: NewsSearchBar	
Responsibilities: Component for a search bar on the Newsfeed. The search query made by the user is sent to the Newsfeed component to be processed.	**Collaborators**: ● Newsfeed

Card8

Component: News	
Responsibilities: Component for an individual article. Displays title, preview, image, data, and publisher of an article and links to the external website where the article is posted.	**Collaborators**: ● Newsfeed

Card9

HTTP Endpoint: backend\routes\community.js	
HTTP Responds to: ● Community.js ● CommunityComment.js ● CommunityDetailedView.js ● DetailedViewCard.js ● FeedCard.js ● ReplyCommentModal.js **Queries and Manipulates:** ● community_post ● communitycomments	
Responsibilities: Query the database for documents in the community_post and communitycomments collections. Modifies the collection by creating new posts, deleting posts, updating posts and comments. Sends	**Collaborators**: ● community_post.js ● communityComment.js

HTTP responses to the client.	

Card10

Database: MongoDB
Collections: ● community_post ● communitycomments **Answers to:** community.js

Responsibilities: Stores data. Answers queries and executes manipulations from backend\routes\community.js	**Collaborator**: community.js

Card11

Page: Market
Child Components: ● CryptoPage **Style**: \frontend\src\components\Pages\Market\MarketPage.css **Hooks:** useFetch **HTTP Requests to:** ● Backend\routes\market.js ● backend\routes\cryptopage.js

Responsibilities: Presentation of the Market section of our application. Request to backend for cryptocurrency data. Each crypto has a follow/unfollow option. Followed currencies are displayed in a separate table which they are removed from when unfollowed. Crypto data is filtered by the crypto symbol when using the SearchBar. If a valid symbol is entered, the SearBar button will open the corresponding CryptoPage. CryptoPage displays historical crypto data on a chart by 24 hours, 12 days, 2.5 months, 9 months.	**Collaborator:** ● CryptoPage ● SearchBar ● NavBar

Card12

HTTP Endpoint: backend\routes\market.js
HTTP Responds to: ● MarketPage

Responsibilities: Retrieves cryptocurrency data from API and sends to frontend.	**Collaborators:** ● MarketPage ● CryptoPage

Card13

HTTP Endpoint: backend\routes\newsfeed.js
HTTP Responds to: ● Newsfeed

Responsibilities: Retrieves newsfeed data from API and sends to frontend.	**Collaborator:** GrabNews

Card14

Component: MakePost

Responsibilities: GUI for user experience to create a post in the community.	**Collaborator**: backend\routes\community.js

Card15

Component: HttpHandler

Responsibilities: A handler used to send REST requests to the database to achieve various actions in the database.	**Collaborator**: ● (backend\models\community_post.js) ● (backend\models\communityComment.js)

Card16

Component: SearchBar
Style:\frontend\src\components\components\SearchBar\SearchBar.css

Responsibilities: A search bar component. Has parameters for the data to search, the search button function, and the onChange input function. Used in the MarketPage	**Collaborators**: ● MarketPage Feed

Card17

Page: CommunityDetailedView
Main Component: CommunityDetailedView.js **Child Components:** ● DetailedViewCard ○ CommunityComment ■ ReplyCommentModal ● NotFoundPage **Style**: community.css **Utilities:** ● ShowMoreText ● HttpHandler **Hooks:** useFetch **HTTP Requests to:** community.js, auth.js

Responsibilities: A detailed view of a social media post. This page provides more info and functionality, such as viewing and posting comments and replies, as well as a button for deleting posts.	**Collaborators**: ● DetailedViewCard ● NavBar ● NotFoundPage ● community.js

Card18

Page: Landing Page
Child Components: ● SpeechBubbles ● Feature1 ● Footer ● Hero **Style**: ● \frontend\src\components\LandingPage\cryptoIllustration.jpg ● \frontend\src\components\LandingPage\heorIllustration.png ● \frontend\src\components\LandingPage\BCLogo.png **Utilities:** ● \frontend\src\App.js ● \frontend\src\Index.js

Responsibilities: A pleasant start to the website. A colorful enjoyable website for which they can access all of the main functionalities to traverse through our website.	**Collaborators**: ● App ● index.

Card 19

HTTP Endpoint: backend\routes\users.js
HTTP Responds to: ● UserProfile and its sub components **Queries:** ● user

Responsibilities: Query the database for a user given their id	**Collaborators**: ● User.js

Card 20

Page: Login
Main Component:LogIn **Style**: ● \frontend\components\Pages\LogIn\LoginIn.css

Responsibilities: A pleasant view for users to be able to log in to the website.	**Collaborators**: ● App ● Index

Card 21

Page: Registration
Main Component: Register **Style**: ● \frontend\components\Pages\Register\Register.css

Responsibilities: A pleasant view for users to be able to sign up to the website.	**Collaborators**: ● App ● Index

Card 22

Page: UserProfile
Main Component: UserProfile **Child Components:** ● ProfileMain. ● LearnProgress. ● ProfileSidebar ● UserPosts ● Currency **Style**: ● \User.css ● \ProfileSidebar.css **Utilities:** ● App ● Index **HTTP requests to** ● **Community.js** ● **Auth.js** ● **Market.js** ● **Users.js** ● **cryptopage.js**

Responsibilities: A main website to display user info such as posts and followers	**Collaborators**: ● useFetch ● ProfileMain. ● LearnProgress. ● ProfileSidebar

	● UserPosts ● Currency

Card 23

HTTP Endpoint: backend\routes\cryptopage.js
HTTP Responds to: ● MarketPage

Responsibilities: Retrieves cryptocurrency historical cryptocurrency data from API and sends to frontend.	**Collaborators:** ● CryptoPage

Card 24

HTTP Endpoint: backend\routes\auth.js
HTTP Responds to: ● Community ● UserProfile ● DetailedViewCard ● CommunitySidebar ● Navbar ● Login ● Register ● Logout Queries and manipulates ● User collection ● deadToken Utilizes ● isLoggedIn, isPostOwner, isCommentOwner, getUser

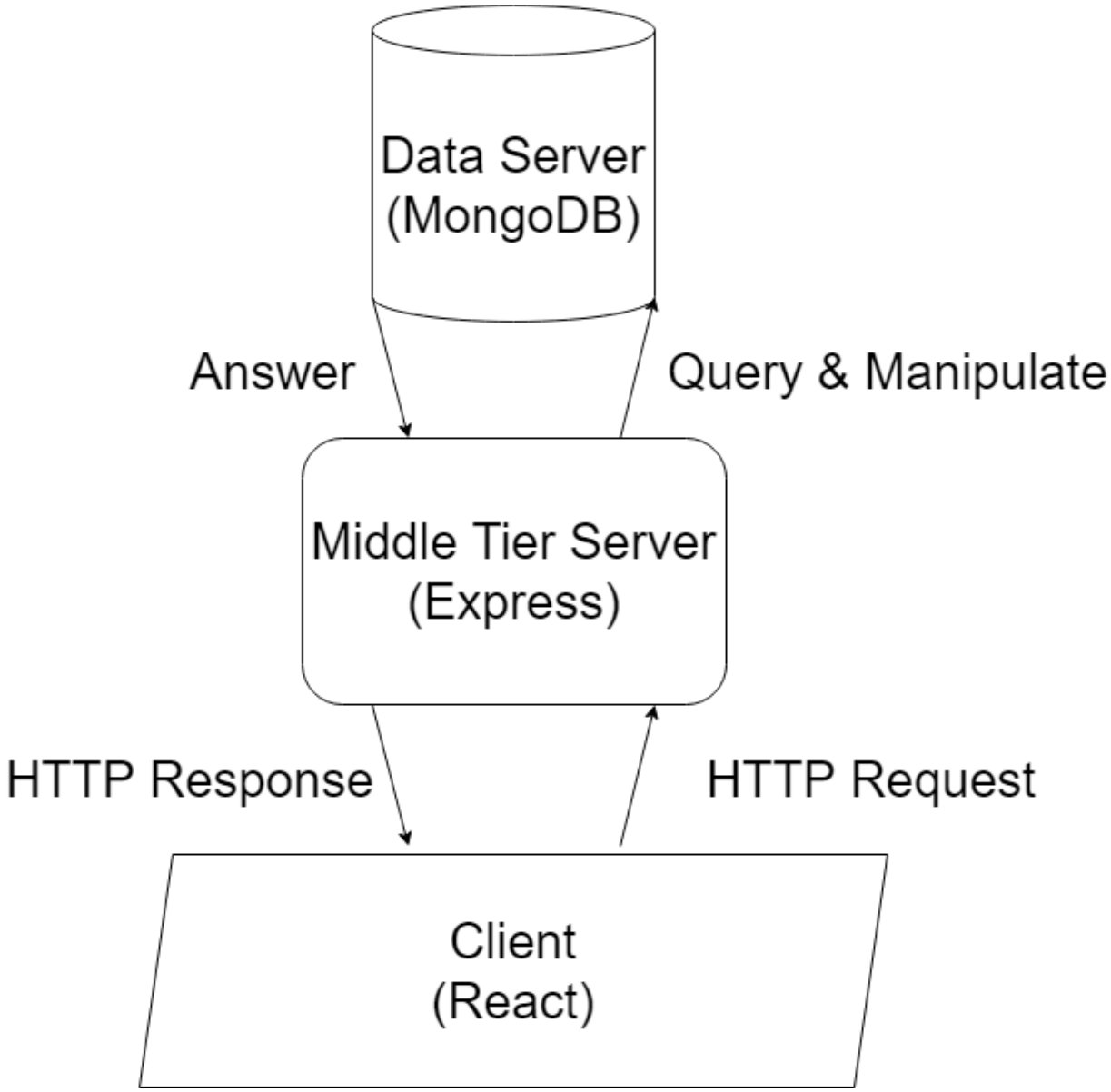
Responsibilities: Handles registering, logging in and logging out users. Also determines if a user is authenticated and if they are authorized	**Collaborators:** ● isLoggedIn, isPostOwner, isCommentOwner, getUser ● User collection

	• Community • UserProfile • DetailedViewCard • CommunitySidebar • Navbar • Login • Register • Logout

System Interaction

There are no dependencies on the OS. No language compilation is needed. No virtual machines are necessary. You will need a MongoDB connection string to connect to our cluster, but this is provided in the repository for now. An internet connection will be required due to the Database access. Please note that the port numbers are statically defined, so if those ports are unavailable, you would have to redefine them to ports that are available on your machine.

Abstract View of System Architecture



Our system follows the three-tier architecture https://www.linuxjournal.com/article/3508

System Decomposition

- Client
    - The react components and the Pages containing several components will execute HTTP requests to the middle tier server and/or to various API endpoints.
    - They make these requests so that they can receive or post data to some database, whether it be to our MongoDB cluster, or whatever database is used by the APIs that we used (to retrieve news articles and crypto-currency market info).
    - They also make these requests to authenticate themselves, as well as authorizing themselves for various tasks such as deleting social media posts or verifying that a profile page belongs to them
    - Our middle tier server and the news/market API send data back through HTTP responses
    - This data can then be presented to the client
    - For example
        - A user visits community/trending-feed
        - Community component executes a GET request to the middle tier server for the trending social media posts
        - Middle tier server sends back a HTTP response containing a list of the most trending social media posts
    - Stylesheets (local and from bootstrap), utilities, and hooks are used to present the data to the client, as well as retrieve it via HTTP requests
- Middle Tier Server/API endpoints
    - Our middle tier server and the API endpoints we used serve as the middlemen between a database and our client-side
    - Our express server (which has its routes split into separate files for cohesion) and API endpoints serve as HTTP endpoints for our client
        - This is where clients request data (they don't directly request data from the database)
    - They also query and manipulate the database (express queries and manipulates our MongoDB cluster using mongoose, and the news and market APIs query and manipulate whatever database they use)
    - They also provide extra security and some business logic (for example sorting social media posts by like counts at the trending-feed endpoint)
- Database
    - MongoDB and its collections we have set up, as well as the databases used by the news and market API make up the database tier in this architecture
    - Answers to queries made by the middle tier server
    - Executes upon manipulations made by the middle tier server (such as the creation of a social media post)

- Strategy for dealing with errors and exceptional cases
    - Client Side HTTP Requests

        - When an error occurs during an HTTP request, the error is logged to the console and will be displayed to the client. Note that the application won't crash.
        - In the case of an HTTP response arriving slowly, through the useFetch hook, we can display a loading message to the client.
            - Furthermore, if data such as a user's profile picture is loading or failed to load, then we display a default profile picture
    - Network error
        - Internet connection is required since the client and middle tier server interact with HTTP requests and responses, and the middle tier server needs to connect to the database. Software will handle this by displaying "failed to fetch" to the user.
    - Invalid Input
        - When a user enters an invalid path into the url, they'll be greeted with a page telling them that the page they are looking for does not exist.