BC-HUB System Design Sprint1

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

CRC Card: **3**
Component: NavBar 3
Page: Education 3
Page: Community 3
Component: GrabNews 4
Component: Newsfeed 4
Component: News 4
HTTP Endpoint: backend\routes\community.js 5
Database: MongoDB 5
Page: Market 5
HTTP Endpoint: backend\routes\market.js 6
HTTP Endpoint: backend\routes\newsfeed.js 6
System Interaction **6**
Abstract View of System Architecture **7**
System Decomposition **8**

CRC Card:

Card1

Component: NavBar
Main:\frontend\src\components\NavBar\NavBar.js **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**: Hongting Li

Card2

Page: Education
Main: \frontend\src\components\Pages\Educational\EducationPage.js **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**: Hongting Li

Card3

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

Responsibilities: Presentation of the Community feeds of our app. Requests Middle tier server for data through GET requests pointed to our HTTP Endpoints.	**Collaborator**: Sahil Malek, Kesavar Kabilar

Card4

Component: GrabNews
Main: \frontend\src\components\GrabNews.js **HTTP Requests to:** backend\routes\newsfeed.js

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

Card5

Component: Newsfeed
Main: \frontend\src\components\Newsfeed.js

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.	**Collaborator**: Faraz Shaikh

Card6

Component: News
Main: \frontend\src\components\News.js

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. The Newsfeed component uses this component in a loop to render many articles and create a newsfeed.	**Collaborator**: Faraz Shaikh

Card7

HTTP Endpoint: backend\routes\community.js
HTTP Responds to: \frontend\src\components\Pages\Community\Community.js **Queries and Manipulates:** community_post collection (backend\models\community_post.js)

Responsibilities: Query the database for documents in the community_post collection. Modifies the collection by creating new posts. Sends HTTP responses to the client.	**Collaborator**: Sahil Malek

Card8

Database: MongoDB
Collections: ● community_post (backend\models\community_post.js) **Answers to:** backend\routes\community.js

Responsibilities: Stores data. Answers queries and executes manipulations from backend\routes\community.js	**Collaborator**: Sahil Malek, Ryan Laporte

Card 9

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

Responsibilities: Presentation of the Market section of our application. Request to backend for cryptocurrency data.	**Collaborator:** Nathan Ralph

Card 10

HTTP Endpoint: backend\routes\market.js
HTTP Responds to: ● \frontend\src\components\Pages\Market\MarketPage.js ● \frontend\src\components\Pages\Market\CryptoPage.js

Card 11

Responsibilities: Retrieves cryptocurrency data from API and sends to frontend.	**Collaborator:** Nathan Ralph

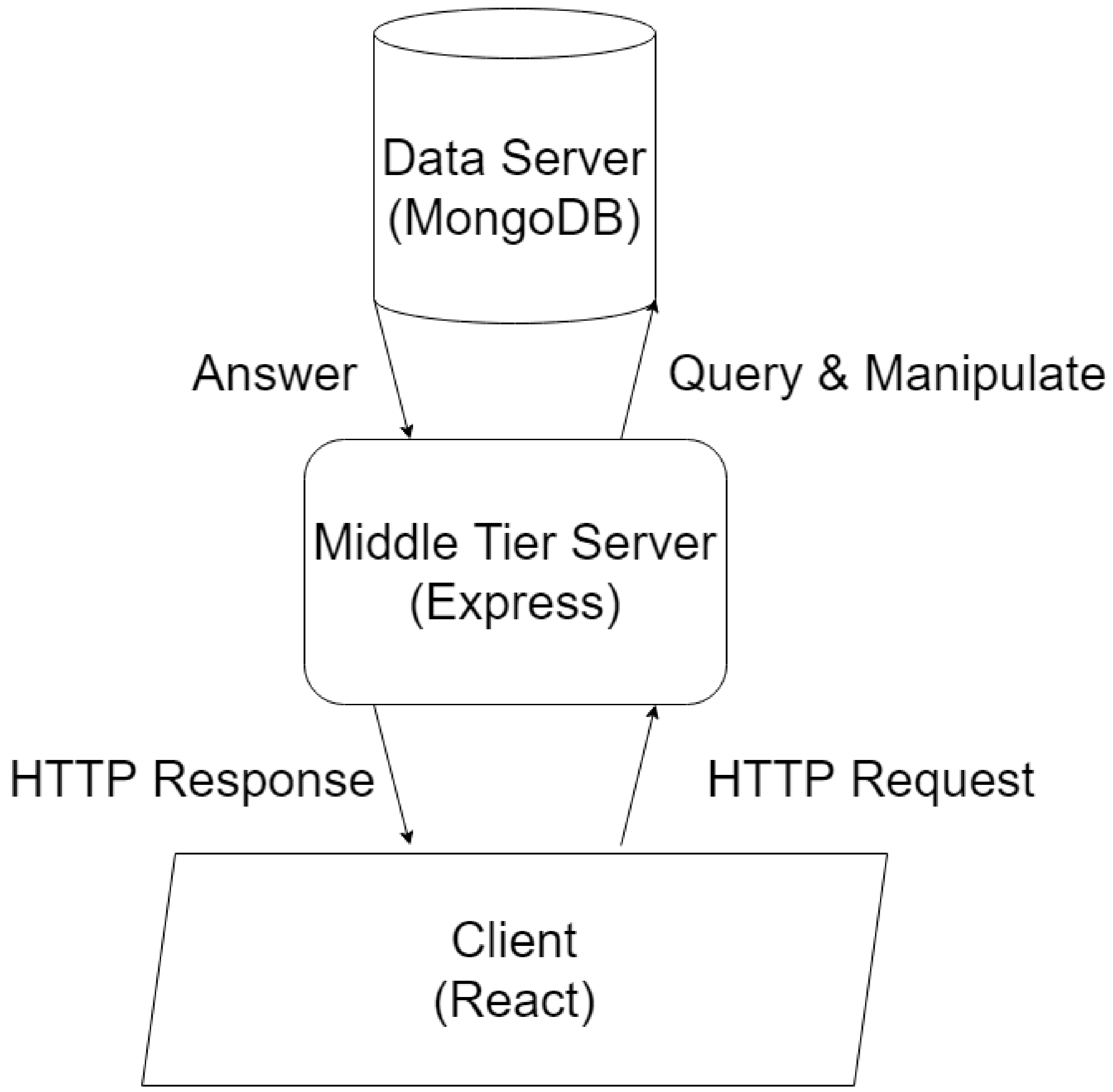
HTTP Endpoint: backend\routes\newsfeed.js
HTTP Responds to: ● \frontend\src\components\Newsfeed.js

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

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 ensure 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).
    - 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.
- 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.