The Hub

CS 4485.0W1 Group Project - Team 2

Aidan Turgut

Vijay Karthikeya Raja

Unaisa Aslam

Brandon Maweu

Aamir Mohammed

Supervisor: Professor Sridhar Alagar

Course Coordinator: Thennannamalai Malligarjunan

Erik Jonsson School of Engineering and Computer Science

University of Texas at Dallas

[Date]

Table of Contents

Chapter 1: Introduction

1.1 Background

Our web application is meant to solve the issue where many people must go to numerous sites when they want to get the latest in multiple topics such as music and sports. These topics are each in separate web pages that can be accessed from a main dashboard page. These pages can only be accessed by creating or logging into an account. The sports, music, and stocks pages provide real-time data for the user to view and keep updated on the latest in each of these topics. Not only that, but our web application also has AI functionality implemented in it that the user can use such as getting recommendations and information on music, sports, and stocks.

1.2 Objectives, Scope, and Goals Achieved

The primary goal of The Hub is to provide a centralized hub for a person's everyday needs on the latest in sports, music, and stocks. Our site offers sports, music, and stocks pages for the user to view and listen to anything related to those 3 things. Our application also allows and requires the user to make an account to access these pages. Our application also gives suggestions and recommendations with AI that the user might need or be curious about such as whether or not a specific stock is worth investing in.

We were able to finish all 3 pages for each of the following pages: stocks, music, and sports. All pages are functional as well with real time data being displayed. We also finished the main dashboard page where the user can access any of those 3 pages. User accounts are also functional in our web application where the user can make an account and log into it at any time to access The Hub. We also added a welcome page to anyone visiting The Hub where they can either log into the website or sign up for an account (separate web pages for signing up and logging in were also made as well). A dark mode feature was implemented in the Settings section of our application that changes the background and some of the styling of all the web pages to a darker theme.

1.3 Key Highlights

- **Feature 1:** Getting news and highlights from the latest sports games and news in stocks
- **Feature 2:** Getting the 10 currently most active stocks and viewing stock data on any stock with a line graph
- **Feature 3:** Listening to the latest music
- **Feature 4:** Getting recommendations for playlists on the music page
- **Feature 5:** Chatbot that the user can ask any question to and getting a response back from the chatbot.
- **Feature 6:** AI recommendation system that recommends the best music, stocks, etc.

1.4 Significance of the Project

This project helps address the problem of a user having to visit various sites to view different things they're interested in. This can make him or her feel lost or overwhelmed, which is why The Hub is a very convenient site with multiple topics (music, sports, and stocks) all in one site and with the latest things related to those topics that the user will most likely be interested in. The site can also be scaled up to include more topics that the user is interested in as well.

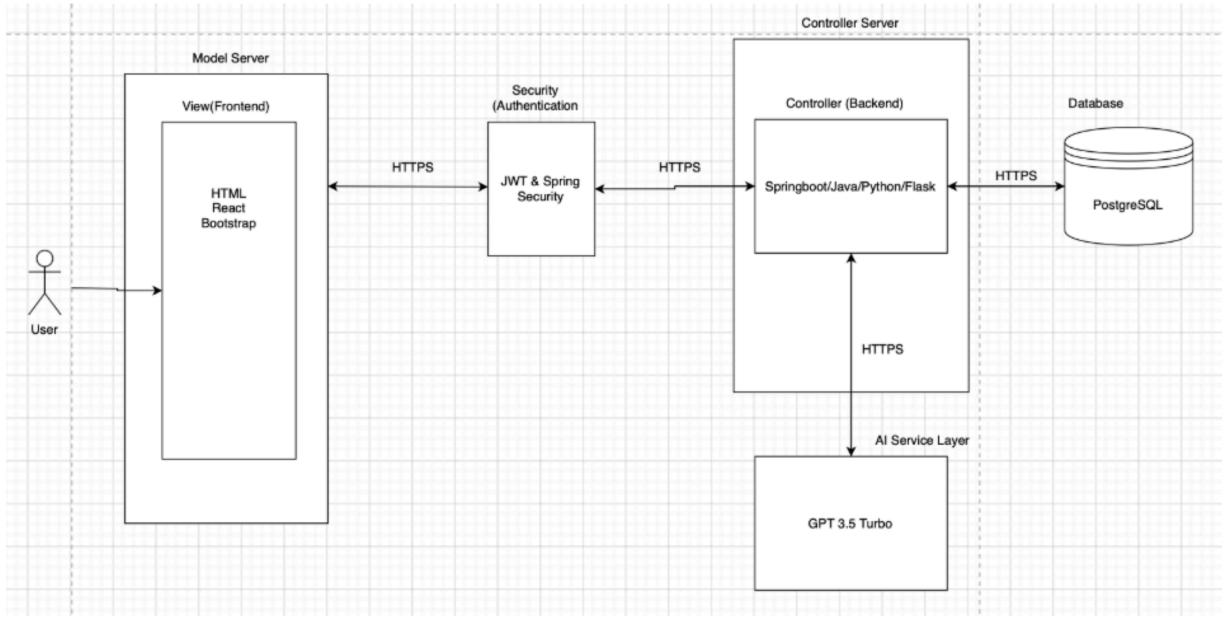
Chapter 2: High-Level Design

2.1 System Overview & Proposed Solution

We used Axios to call multiple APIs that deliver different real time data to our site every time the user visits the pages in them. The APIs that we used were meant for obtaining current news and other real-time data from the internet to display in various pages in The Hub. The OpenAI API was also used to create the AI chatbot in the SportsHub page and the StocksHub page.

2.2 Design and Architecture Diagrams

[Include system architecture, flow diagrams, and database schemas, ER diagrams.]



2.3 Overview of Tools Used

The web pages were developed using the VS Code IDE while the back end was mostly done in Java with the IntelliJ IDE. Visual Studio Code allowed the front end developers to write the code for the web pages of our web application in jsx source code files and share those files to Brandon's GitHub repository using Git and Git bash. The database was created using PostgreSQL and the APIs were tested with Postman.

Chapter 3: Implementation Details

3.1 Technologies Used

Frontend:

- HTML, CSS, Javascript: Core web technologies for markup, styling, and interactivity.
- React: Single-page application framework.
- Vite: Provided fast local builds and hot-module reloading.
- Axios: Promise-based HTTP client for communicating with our backend APIs.
- Chart.js: rendered interactive line charts for our time-series data.
- Bootstrap: Provided responsive grid and prebuilt components for our UI layout.

Backend:

- Java: used as the primary language for our backend services, provided strong typing, and support and integration with Spring Boot.
- Spring Boot & Spring Security: rapidly bootstrapped and configured RESTful APIs, dependency injection, and application properties. Spring Security secured our endpoints with role-based access control and integrated JWT authentication filters.
- Maven: manages all of our Java dependencies, and builds lifecycles.
- Python & Flask: built a lightweight microservice for our recommendation tasks and exposed it via REST endpoints.

Database:

- PostgreSQL: our relational database for persisting users and historical data.
- pgAdmin: GUI tool for schema management for our database.

Version Control & Project Management:

- Git: For source control
- GitHub: Hosting the central repo, supported branching workflows and pull requests.
- GitHub Projects: Kanban boards and issue tracking to organize tasks and coordinate sprints.

External APIs & Cloud Services:

- Youtube v3 API: Fetched and displayed relevant video content.
- AlphaVantage API: Pulled real-time stock price data and basic company news for individual tickers.
- Financial Modeling Prep API: Retrieved the top 10 most active stocks.
- Last.fm API: Pulled trending tracks and artist-specific playlists.
- NewsAPI: Aggregated headlines and full-text articles across customizable news sources.
- Open AI GPT - 3.5 Turbo: Integrated via API to power AI-driven features (e.g content summarization).

Tools:

- IntelliJ Idea: Java/Spring Development with integrated debugging, refactoring, and Maven support.
- Visual Studio Code: Frontend, Python, and miscellaneous editing, with extensions for ESLint and REST clients.
- Discord: Communication between team-members.

3.2 Code Documentation

GitHub link: https://github.com/BrandonMDallas/CS-4485-Project

The frontend is organized around a React/JavaScript frontend with a thin HTTP abstraction layer and context-based authentication. The root is a main.jsx which bootstraps the app by wrapping App.jsx in an AuthProvider and renders to the DOM. App.jsx defines our client-side routing using React Router: public routes (e.g Welcome, Login, Register) and protected routes (e.g SportsHub, StockHub, MusicHub) which are guarded by a RequireAuth component. All API interactions funnel through a single apiClient.js class that wraps Axios calls against a configurable base URL. Once authenticated the users land on Dashboard.jsx which has three primary tabs Main, Profile, and Settings. In addition each Hub encapsulates its own data fetching, state management, and UI logic. SportsHub manages sport selection, team following list, and a simple chatbot UI that responds to users queries about scores, players and news. MusicHub integrates with our backend to fetch top tracks, search, and personalized recommendation along with playlist CRUD operations. StockHub fetches time-series data from our backend, renders charts via Chart.js, and surface financial news.

The backend is a Spring Boot application structured around clear separation of concerns. At startup, we have a DotenvInitializer which loads all third-party API keys into Spring's environment, while CorsConfig and SecurityCorsConfig open CORS to our react frontend. Security is centralized in SecurityConfig, which wires a stateless JWT filter(JWTAuthenticationFilter & JWTAuthEntryPoint) tokens are minted and validated by JWTGenerator using HS512 with expiration defined in our SecurityUtils class.
Controllers map HTTP routes to service calls via simple DTOs: AuthController handles registration and login, UserController and WidgetController expose standard CRUD operations through UserService and WidgetService, and we have multiple controllers for the external APIs we use (e.g AlphaVantage, LastFM, NewsAPI) which just relays query parameters to its service layers for external API calls. The recommendation_service.py is a module that implements a lightweight Flask microservices that has two capabilities, music recommendation and stock recommendation. The music recommendation fetches tracks via LastFM and applies our algorithm for determining what songs to recommend. The stock recommendation fetches the ticker and does a recommendation based on sector and volatility of the stock.

3.3 Development Process

We adopted an agile workflow organized into weekly sprints, breaking our team into frontend and backend pods. The frontend pod (Aidan, Unaisa, and Aamir) each had owned a hub page, Aidan built StocksHub, Unaisa built MusicHub, and Aamir built SportsHub. The backend pod (Brandon and Vijay) jointly developed our server API and the AI driven recommendation engine that powers the pages and lastly handled the integration for features on the frontend to the backend. We tracked features, chores, and bugs in GitHub projects and used a Kanban board to manage priorities and sprint planning. We had weekly meetings on discord to keep ourselves aligned where we did reviews at the end of each meeting to showcase progress and gather feedback. Code was version controlled in GitHub, with pull request reviews and facilitating collaborative debugging sessions whenever issues arose.

Chapter 4: Performance

4.1 Testing and Validation

We ran our React pages from Visual Studio Code with Vite and Node.js. Before adding APIs to our source code files, we made sure that they worked and tested them to see what data (in the form of JSON files) they would give back to us (based on different requests to them) with Postman. Our AI recommendation system was tested by Vijay and Brandon by them inputting sample data to see what output the system would return (and improving them if needed afterwards). We didn't really use any testing strategies and just ran and compiled our code to see if it worked and did what we wanted it to do.

4.2 Metrics Collected and Analysis

Some of the metrics we were able to obtain was initially based on the loading time of the pages and the quality of access for each of the pages, which was primarily focused on the responsiveness and usability of the features. Within our pages, our metrics were mostly qualitative and focused more on having low-latency responses.

For the Music Hub, the main focus was being able to make the API calls swiftly and accurately. Fortunately, we were able to satisfy both of them as when the user opens the page, the trending songs gets loaded instantly so the user would be able to interact with the page almost instantly and even the Music Recommendations were instant. Additionally, outside the API calls, creating the playlists and any other functionality were as instant as could be.

For the Sports Hub, one of the main focuses was how information would be loaded in when an API call is made when the user clicks on something and how quickly the API would respond to environment changes, both accurately and swiftly. This would allow people to quickly access information on the page and we made sure to keep that in mind when developing the sports page since people want to look at information at a glance and not spend too much time going through it.

For the Stocks Hub, one of the biggest problems was with the API usage as a lot of the APIs that we planned on using were limited to a few calls a day, so it was hard for us to test it. Additionally, the API calls would vary depending on what we would want to display, whether it be the stock graph, the recommendations, or just a basic search of the stocks available. With that though, it was responsive and gave low latency outputs.

Chapter 5: Lessons Learned

5.1 Insights Gained

A key takeaway that we got from this project was how important planning is before doing the actual implementation of any project. We realized that we should've spent a bit more time at the beginning planning out all the features our website would have, the layout and organization of each web page, and listing all the functionalities and requirements that our website would have (thus allowing programming of them to be a lot easier and straightforward).

Having the main idea planned out more thoroughly would allow us as a team to prepare for challenges we faced on the technical side of the project as well.

5.2 Lessons from Challenges

We as a group faced a lot of challenges in integrating the separate work that we each did with Git and GitHub. We learned that planning and organizing how you work on a project as a team is important to do first before splitting work to separate team members. Not only that, but making sure that any work that is done by a team should be able to run or be accessed properly by any teammate (such as a computer program on each member's laptop/desktop). This issue was mainly present near the end of the semester where some of us couldn't run code on VS Code properly without getting help with working with Git and Git bash in the IDE.

Communication challenges also happened sometimes and when that happened, especially later on in the project cycle, we made sure to talk it through and get on the same page to finish the project strongly.

5.3 Skills Developed and Improved

We mainly learned how to develop web pages in React and connect them to a program that runs a server in the back-end. We also learned and utilized APIs a lot (including parsing JSON files to extract data from other sources) primarily with GET requests using Axios. Our teamwork skills developed significantly throughout the semester as we learned how to deal with conflicts that can arise with any team working on a project (primarily with integrating work or ideas together into the final product).

Chapter 6: Future Work

6.1 Proposed Enhancements

One improvement that can be made to our web application is enhancing our AI recommendation system to deliver the user more information and features/services such as whether a specific stock in the StocksHub page is a good investment based on how that stock is doing recently. Another thing we could improve is making the styling of the web pages more consistent with each other while keeping functionality simple and consistent between pages to add a good flow between pages.

6.2 Recommendations for Development

Future teams should spend at least 1-2 weeks and many hours solely planning out how the development process for their project will be throughout the semester and all the requirements, design, tech stack, and integration process will be for their project. Every team should also check on a regular basis from time to time the progress that each team member has made so far and see whether or not their work can be properly integrated to the entire project. This ensures that as progress is made throughout the semester with the project, the team doesn't have to go back and make significant and difficult changes or fixes. Make sure to help out other people as well when someone feels another person may be struggling or not doing as much as they can be doing for the current week.

Chapter 7: Conclusion

7.1 Summary of Key Accomplishments

We accomplished building a working website with a back-end server running and handling data from the website. We also succeeded in allowing the website to extract real time data and valuable information from internet sources with the use of various APIs and Axios. Not only that, but we also managed to build a working AI recommendation system that helps provide the user information on recommended music, stocks, etc.

7.2 Acknowledgements

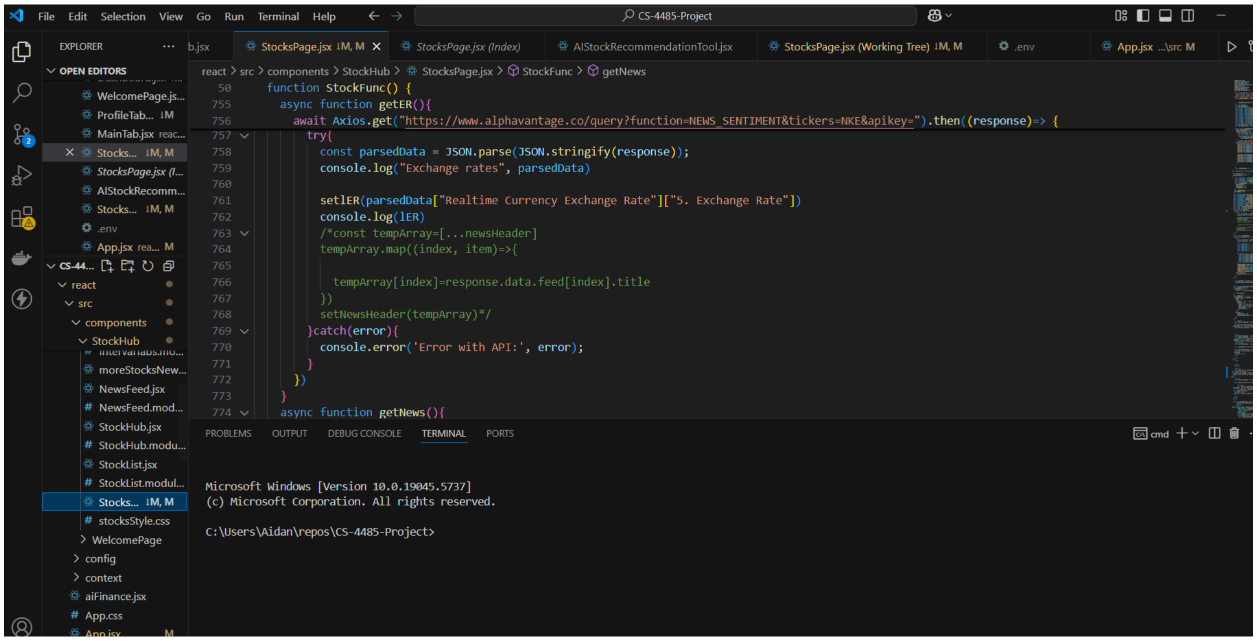
Thank you professor Alagar and Thenn, for meeting with us and giving us honest feedback on the progress that we made with our project throughout the Spring 2025 semester. Your feedback helped our web application look nice, be fully functional, and have AI functionality in it.

References

- https://www.geeksforgeeks.org/
- https://www.youtube.com/
- https://www.w3schools.com/
- https://www.alphavantage.co/
- https://site.financialmodelingprep.com/developer/docs

Appendices

Here’s a snippet of some of our code in Visual Studio Code.



```
function StockFunc() {
  async function getER(){
    await Axios.get("https://www.alphavantage.co/query?function=NEWS_SENTIMENT&tickers=NKE&apikey=").then((response)=> {
      try{
        const parsedData = JSON.parse(JSON.stringify(response));
        console.log("Exchange rates", parsedData)

        setlER(parsedData["Realtime Currency Exchange Rate"]["5. Exchange Rate"])
        console.log(lER)
        /*const tempArray=[...newsHeader]
        tempArray.map((index, item)=>{

          tempArray[index]=response.data.feed[index].title
        })
        setNewsHeader(tempArray)*/
      }catch(error){
        console.error('Error with API:', error);
      }
    })
  }
  async function getNews(){
```

```
Microsoft Windows [Version 10.0.19045.5737]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Aidan\repos\CS-4485-Project>
```