Final Project Report

ARGO UX Master Component Library

Authors

Kevin Roa, Aliah De Guzman, Saud Baig,

Noah Turrin, Te'a Washington, Samuel Anozie,

Co-Authors

Mark Bentsen

Has every member of your group contributed to this document and participated in the project meetings?

Yes

EXECUTIVE SUMMARY

This report comprises the Project Management Plan, Requirements Specification, Architecture, Design, and Testing plan pertaining to the Proof-of-Concept UX Master Component Library, completed as part of a collaboration between ARGO and UTD.

The project was managed with a Scrum methodology, with Notion as our technology of choice for task and sprint tracking. Potential risks, reduction strategies for each risk, and professional standards were all agreed upon at the beginning of the project.

Requirements for this project were documented via use cases, which are summarized as follows:

1. Importing a component into a project.
2. Modifying a component within a project.
3. Modifying the base component.
4. Creating new components.
5. Viewing existing components.
6. Removing components.

Architecturally, this project relies on a bundled deployment strategy for the component library, and a presentation and data layer for the user interface deliverables. Various software libraries interact within their own layers. Material UI, HighCharts, FlexLayout, and React all serve as the backbone of our front-end architecture. NPM & Chromatic handle the deployment of the component library, while Storybook is the technology of choice for creating and maintaining the library.

Our designs for the components themselves rely heavily on ARGO's own documentation and Material UI guidelines, while the designs for the Stock Analytics, Social Media Analytics, and Fresh Oranges user interfaces were created by iterating on reference designs. The component architecture is designed by identifying required props that the React components would need in order to function properly. Each of these designs are traced back to the original requirements to ensure consistency and diligence in design.

Finally, the testing plan ensures that each and every component created works as desired, with test cases unique to each component's use case. These cases are generated based on the design and intended usage for each of the components.

We extend a special thanks to the class leaders and ARGO representatives that worked with us throughout the completion of this project.

TABLE OF CONTENTS

LIST OF FIGURES

List Of Images

LIST OF TABLES

List Of Tables

INTRODUCTION

PURPOSE AND SCOPE

The purpose of this document is to provide an all-inclusive overview of the entire ARGO UX Master Component Library project. This document goes over every piece of written documentation provided to ARGO throughout the duration of the project.

PRODUCT OVERVIEW

The ARGO UX Master Component Library is a two directional UX master component library (MCL) which intersects development with design. It provides scalability and enforces standards documented by the UX program. The functional React components can be edited in the MCL and consumed across all ARGO projects.

The library is built using Storybook, distributed using NPM, and used in various React projects.

DOCUMENT STRUCTURE

The document contains the major points from each course deliverable. The following deliverables are covered: Project management plan, requirements documentation, architecture documentation, detailed design documentation, and testing plan. Headings and subheadings are used to organize each section.

TERMS/ACRONYMS/ABBREVIATIONS

Term/Acronym/Abbreviation	Description
Actor	A user of the system
CD	Course Deliverable
CI/CD	Continuous Integration, Continuous Deployment
FR	Functional Requirement
JS	Javascript
Material UI	A component library.
MCL	Master Component Library
NFR	Non-Functional Requirement
NPM	Node Package Manager
PD	Project Deliverable
React	A frontend javascript framework used to build UIs.
Storybook	A javascript library used to build UI components in isolation
TS	Typescript
UC	Use Case
UI	User Interface
UX	User Experience

Table 1. Terms & Descriptions

PROJECT MANAGEMENT PLAN

PROJECT ORGANIZATION

Group Members

Name	Contact	Role	Rationale
Kevin Roa	kar180005@utdallas.edu	Group Leader, Developer	Leadership role decided via group decision.
Samuel Anozie	sca180003@utdallas.edu	Developer	Individual experience level
Saud Baig	ssb190008@utdallas.edu	Developer	Individual experience level
Aliah De Guzman	and180008@utdallas.edu	Developer	Individual experience level
Noah Turrin	nmt170002@utdallas.edu	Developer	Individual experience level
Te'a Washington	tvw190000@utdallas.edu	Developer	Individual experience level

Table 2. Group Members

Course Organizers

Name	Contact	Role
Eric Wong	ewong@utdallas.edu	Course Instructor
Li Dongcheng	dxl170030@utdallas.edu	Course TA
Chen Zizhao	zizhao.chen2@utdallas.edu	Course TA

Table 3. Course Organizers

Project Sponsors

Name	Contact	Role
Mark Bentsen	mark.bentsen@argodata.com	QA Manager (Main Contact)
Ponchai Reainthong	ponchai.reainthong@argodata.com	UX Director
Raisa Gonzalez	raisa.gonzalez@argodata.com	UX Program Manager

Table 4. Project Sponsors

LIFECYCLE MODEL USED

- Agile (Scrum)
    - Every week is a new sprint
    - Based on the previous sprint:
        - Make necessary changes
        - Review sprint backlog to decide next course of action
        - Allocate tasks to group members
    - Meet 3 times weekly:
        - Monday: Meeting with ARGO
            - Discuss progress/issues/future/etc
        - Wednesday/Friday: Group meeting
            - Discuss current sprint
    - Provide weekly progress reports/demos
        - Get feedback, implement in next sprint
    - Deliver incrementally
- Rationale:We chose scrum since it allows for flexibility when it comes to requirements changes and is an involved process that makes sure we are able to adapt to client needs. By getting consistent feedback, we are able to create useful features.

RISK ANALYSIS

Risk	Severity	Likelihood	Reduction Strategy
Errors with infrastructure technology (e.g. Azure complications, Storybook.js complications)	High	Low, Azure will be set up by client company	Investigate up-to-date requirements for deploying Storybook.js to cloud services, especially Azure
Errors with integrating technology (e.g. Material UI style overrides have bugs)	Medium	Medium, is known to be an issue with framework technology	Create proof-of-concept for custom classes, or refrain from using frameworks all together.
CI/CD pipeline desyncs between deployments	High	Low	Reducing the amount of steps in deployment should reduce this risk, as well as implementing rollback capabilities.
Cached versions of deployed projects not updating on browser	High	Very Low, there should be internal code within Azure to stop this from happening	N/A
Restrictive customizability of components	Medium	Medium, depends on the implementation details and the requirements from the client	Flush out the implementation details from the client and develop the components accordingly
Team member sickness	Medium	High due to the flu and Covid season	Have clear documentation in case a team member has to take over a deliverable of the sick team member
Team member has heightened external workload (e.g Project in another course)	Medium	Medium	Communicate situation to group members. Adjust project schedule to fit reduced capabilities.

Table 5. Risk Analysis Matrix

HARDWARE RESOURCE REQUIREMENTS

- Desktop/laptop computer
- Mobile phone

SOFTWARE RESOURCE REQUIREMENTS

- StorybookJS
- TypeScript/JavaScript
- React
- HTML
- CSS
- MaterialUI for React
- Highcharts
- Golden-layout

DELIVERABLES AND SCHEDULES

Key	
CD	Course Deliverable
PD	Project Deliverable

Table 6. Deliverable Key

ID	Deliverable	Dependencies	Time Est.	People Est.	Rationale
CD1	Project Management Plan		3 days	6	Can complete with information from first sponsor meeting
CD2	Requirements Documentation	ARGO Project Proposal, CD1	10 days	6	Time-frame allows each member to ensure that the requirements encapsulate the scope of the project effectively

CD3	Architecture Documentation	CD2	10 days	6	Time-frame fits within the project timeline and allows for it to be used in a timely manner by design phase
CD4	Detailed Design Documentation	CD3	10 days	3	After the initial architecture documentation is complete, expanding on it should be doable within this time-frame
CD5	Testing Plan	Development Schedule	5 days	2	Will use automated testing tools
CD6	Final Project Report	CD1, CD2, CD3, CD4, CD5, PD1, PD2, PD3, PD4	7 days	6	Most of the content is from previous deliverables
CD7	Final Project Demonstration	PD1, PD2, PD3, PD4	5 days	6	Team has to include all project functionality in the presentation
PD1	Component Library (Simple Components) & Storybook JS Initialization	CD3	7 days	6	Implementing simple components will allow for early validation of the product.
PD2	Component Library (Advanced Components)	PD1, CD4	21 days	6	Will need a longer period of time to create advanced components compared to the simple components due to complexity

PD3	Component Library (Niche / Placeholder)	PD2	5 days	6	
PD4	Stocks Analysis UI	PD2	7 days	2	Team can split up to design all interfaces in a smaller timeframe
PD5	Social Media Analytics UI	PD2	7 days	2	Team can split up to design all interfaces in a smaller timeframe
PD6	Rotten Tomatoes Clone UI	PD1	7 days	2	Team can split up to design all interfaces in a smaller timeframe
PD7	MCL Update System	PD1	10 days	3	Team can split up to work on update system

Table 7. Deliverables

MONITORING/REPORTING/CONTROLLING MECHANISMS

Monitoring and Control Deliverables

- Requirements Traceability Matrix (RTM)
    - The use of RTM ensures that all requirements are accounted for, verifies that the requirements are met and align with project objectives, and tracks changes over the course of the project
- Review and Status Meetings
    - Regularly assess progress with sponsors, and identify any obstacles to a project or task.
    - Provide a platform for team members to communicate updates, address concerns, and ensure alignment towards the common goal.
- Project management tool (Notion)
    - Track the progress of deliverables, as well as who is responsible for each deliverable.
    - The developers can focus on the implementation of the deliverables.

PROFESSIONAL STANDARDS

The below standards are being set to ensure professionalism is upheld throughout the project's duration. It is imperative that the team adheres to these standards in order to ensure that the project is a success. Communication is heavily emphasized since it is the solution to a lot of problems.

- Team members should communicate on a regular basis with the team regarding any issues or blocks they are encountering
- The team should communicate with the stakeholders at Argo regarding any questions they may have
- Team members should come to meetings on time unless previously communicated
- Team members should not utilize unapproved materials for deliverables
- Team members should complete deliverables on time, however, if they are facing issues that could change the deliverable date these should be communicated ASAP

CONFIGURATION MANAGEMENT

Document	Version	Changes	Reviewed By
Project Management Plan	1.1	Expand description of deliverables, other minor modifications	All Group Members; Mark Bentsen
Requirements Documentation	1.0	Initial Document Creation	All Group Members; Mark Bentsen
Architecture Documentation	1.0	Initial Document Creation	All Group Members; Mark Bentsen
Detailed Design Documentation	1.0	Initial Document Creation	All Group Members; Mark Bentsen
Testing Plan	1.0	Initial Document Creation	All Group Members; Mark Bentsen
Final Report	1.0	Initial Document Creation	All Group Members; Mark Bentsen

Table 8. Configuration Management

PROJECT IMPACT

INDIVIDUAL

The technologies we used in this project provided us with valuable experience and skills. We got the chance to work with in-demand technologies such as StorybookJS, TypeScript/JavaScript, React, and MaterialUI that are widely used in the industry. The experience gained from working with these technologies can make us more marketable to potential employers and better prepared for future projects. Additionally, working on a project in a team setting allowed us to develop our collaboration and communication skills. We understand that in the workplace, projects are usually completed in a team setting, and the experience gained from working on this project can help us be more effective and efficient in future team projects.

Completing a project like this gave us a sense of accomplishment and confidence in our abilities. We can use this experience to showcase our skills to potential employers and peers. In conclusion, the impact of this project can be significant for each of us. The experience gained from working with in-demand technologies, developing collaboration and communication skills, and the sense of accomplishment can all positively impact our professional and personal lives.

ORGANIZATIONAL

The project to create a UX Master Component Library is expected to have a significant impact on the company, particularly in the management and centralization of the design system. Currently, the company's functional UI components are managed in the local project, resulting in a lack of consistency and reusability. With the implementation of a centralized design system, the company can now easily manage and update shared components across multiple products. This can result in significant time and cost savings for the company, as developers will no longer need to recreate similar components across different projects.

The implementation of a UX Master Component Library is central and critical to the ARGO design system. The UX Master Component Library is expected to have a significant positive impact on ARGO's design system, ultimately leading to more efficient and consistent development processes and an improved user experience.

REQUIREMENT SPECIFICATIONS

STAKEHOLDERS

- ARGO UI/UX developers
- ARGO system developers
- Affiliated banks
- Affiliated medical facilities

USE CASE MODEL

UC1: Use Component	
Participating Actors	User, Storybook System
Entry Conditions	- Need reusable front-end component for an application - Component exists
Normal Flow of Events	1. User imports a component from the library 2. User includes the component in their usage 3. User customizes the component to suit their needs
Exit Conditions	- The component is fully functional and operates correctly within the application.
Exceptions	a. Component does not fit users' needs b. Certain aspect of component can't be customized
Extensions	a.1: See if another component handles the problem a.2: Have developer actor create new component for the library b.1: Have developer actor implement the customization capability to the library

Table 9. UC1

UC2: Customize Component	
Participating Actors	User, Storybook System
Entry Conditions	- Has specific requirements for application front-end design that requires customizable components - Component exists
Normal Flow of Events	1. User imports a component 2. User adheres to implemented design guidelines to customize component
Exit Conditions	- Component is fully customized
Exceptions	a. There may be a time when a user may want to customize a component beyond the implemented design guidelines. b. The customization may not be possible with the current storybook component implementation
Extensions	b.1: See UC3

Table 10. UC2

UC3: Refactoring Component	
Participating Actors	Developer, Storybook System
Entry Conditions	- Change in requirements from the overall Argo organization related to front-end design - Developer has access to component design library codebase
Normal Flow of Events	1. Developer modifies files related to component that needs update 1. Developer tests changes 2. Developer requests code change on main component design library 3. Administrator reviews and approves code changes 4. Changes are pushed to live storybook library 5. Projects using the component update to latest version
Exit Conditions	- Component changes are consumed by existing and new users of said component
Exceptions	a. Component design library code may not compile for developer b. Changes introduced by developer can break component design library c. Administrator may not approve code change
Extensions	a.1: Likely caused by developer, diagnose issues with refactor. b.1: Ensure that any changes made to a component still function properly on any project it is used within before making it live. c.1: Various reasons; Work with the administrator to understand why the component was not approved.
Special Requirements	- Changes introduced to component library must adhere to usability and ensure that updated component is easy to use - Changes must also adhere to maintainability and extensibility in case future changes are needed

Table 11. UC3

UC4: Create Component	
Participating Actors	Developer, Storybook System, Development team
Entry Conditions	Need for new component to be stored
Normal Flow of Events	1. Developer creates a component for use in products 2. Developer tests changes 3. Developer requests code change on main component design library 4. Development team reviews and approves code changes 5. Changes are pushed to live storybook library 6. Projects using the component update to latest version
Exit Conditions	- Component is consumed by existing and new users of said component
Exceptions	a. Tests fail b. Review is denied c. Projects use old version of the library
Extensions	a.1: Fix bugs in component code and try again b.1: Conform to project standards and try again c.1: If projects need the new version, update dependency version and use latest
Special Requirements	- Changes introduced to component library must adhere to common design standards

Table 12. UC4

UC5: View Component	
Participating Actors	User, Storybook System
Entry Conditions	- User should have access to the component library system - User has chosen a specific component to view
Normal Flow of Events	1. User enters the component library 2. User selects a component to view 3. The Component Library System displays the component along with information about it
Exit Conditions	- User is able to see the displayed component
Exceptions	a. If the user is not authorized to view the Component Library System, they will be denied access
Special Requirements	- The Component Library System needs to be able to verify developer identity - The Component Library System should contain 1 or more user interface components

Table 13. UC5

UC6: Delete Component	
Participating Actors	Developer, Storybook System
Entry Conditions	- Developer has ensured the component that they will delete is no longer required for any project - Component to be deleted is not a part of other reusable components
Normal Flow of Events	1. Developer opens the master component library 2. Developer selects the component they want to delete 3. Developer selects the option to delete the component 4. Developer confirms component deletion
Exit Conditions	- The component is removed from the master component library
Exceptions	a. The component is being used in a project
Extensions	a.1: Refactor the project to use an alternative component a.1.1: Remove the current component dependency a.2: Deprecate the component but keep it within the library for some time to give other projects time to migrate
Special Requirements	- The developer is required to get administrator(s) approval to remove a component - Once the developer selects the option to delete a component, it is removed from the library within 5 seconds

Table 14. UC6

GRAPHIC USE CASE MODEL

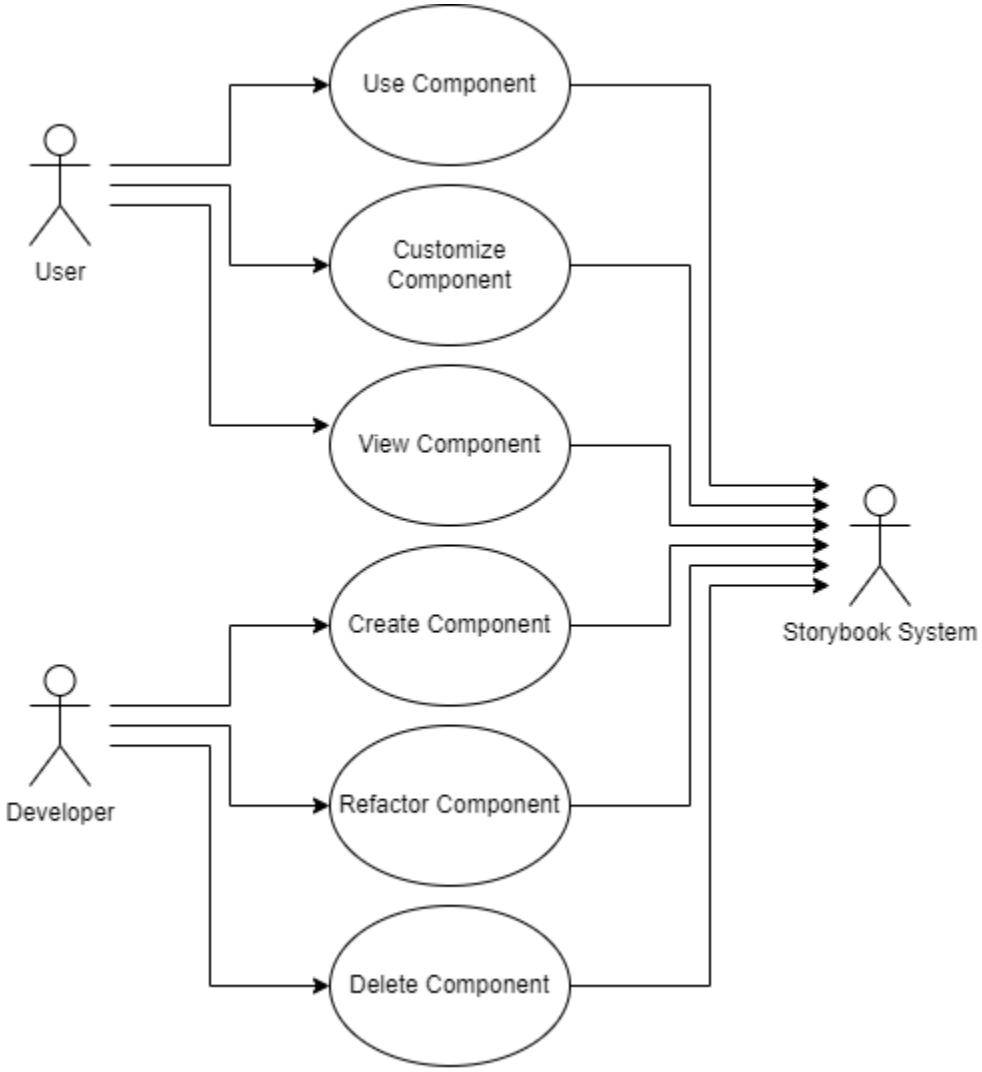


Figure 1. Graphic Use Case Model

USE CASE DESCRIPTIONS

Use Case	Description
UC1	A user of the library wants to implement a component into their react project.
UC2	A user of the library wants to modify a component to better fit their project.
UC3	A component developer wants to modify a component in the library.
UC4	A component developer wants to create a new component for the library.
UC5	A user of the library wants to view an existing component in the library.
UC6	A component developer wants to remove a component from the library.

Table 15. Use Case Descriptions

USE CASE RATIONALE

Use cases are split between “Users” and “Developers”. Users are individuals who would like to utilize components from the library within a separate project. The storybook system grants them the ability to view/import/customize components. Developers are individuals who maintain the component library; their role is to create/modify/delete components from the system. All use cases connect to the same storybook system because it is built to handle all of those types of interactions. Our job as group 4 is to utilize the storybook library to facilitate the interactions described within the use cases.

NON-FUNCTIONAL REQUIREMENTS

1. Usability
    a. The system shall provide easy access for creating/viewing/modifying/deleting components.
    b. The system shall provide the ability to easily use its components within other projects.
    c. The system shall have documentation for each component deployed.
2. Maintainability
    a. The system shall ensure that components are upgradable in the future.
    b. The system shall house all components within the same repository.
    c. The system shall be built using proper design standards.
    d. The system shall be built using open source libraries.
    e. The system shall automatically deploy new changes to the library.
3. Extensibility
    a. The system shall be able to accept the creation of new components.
    b. The system shall be able to allow the modification of old components.
    c. The system shall feature components with easily modifiable parameters.
    d. The system shall propagate changes across component implementations.

ARCHITECTURE

ARCHITECTURAL STYLE

Argo-Storybook

To ensure rapid development and ease of use, the Storybook project is architected using a bundled deployment strategy. Distributions of the project are made available for developers to consume, while changes to the distribution are managed by a versioning system. The proof-of-concept version of Storybook uses Node Package Manager to distribute the build files, and different versions of the distribution are handled by both NPM and Github. Changes to any version generate a new version, which is updated on Github and automatically bundled and deployed to NPM via a continuous deployment pipeline.

UI Implementations

All the UI interfaces uses one presentation layer made with React.js, and a data layer provided by third-party APIs, deployed on the client. This architecture does not include user authentication or storage. Highcharts is used to visualize the stock-specific data.

RATIONALE

Argo Storybook

Our team opted for a bundled deployment strategy for our project because it allows us to avoid the complexities of managing individual components and dependencies, and reduces the risk of compatibility issues between different versions. We can also reduce the time and effort required for deployment, as the entire application can be deployed with a single command.

Roadblocks, Issues, Concerns: One issue with a bundled deployment strategy is the risk of code bloat, as all of the application code and dependencies are bundled together. This can result in larger bundle sizes and longer load times, which can negatively impact the user experience. The constraints of using a bundled deployment strategy include the potential for increased bundle size, longer load times, and the risk of version conflicts.

UI Implementations

A 2-tier architecture allows us to separate the presentation layer and data storage layer, which offers greater flexibility and scalability. This architecture also eliminates the need for server provisioning and management, which reduces costs and improves performance. We decided to use React because it is a popular and widely-used JavaScript library for building user interfaces. It provides a rich set of features which can lead to faster development and easier maintenance. We chose to use Highcharts to provide powerful data visualization that can help present complex data in an intuitive and visually appealing way. Combining these technologies in a 2-tier architecture can result in a scalable, maintainable, and user-friendly application that can meet the needs of a wide range of users.

Roadblocks, Issues, Concerns: A potential roadblock is the complexity of the architecture because using multiple layers and services can make it difficult to ensure smooth integration and operation. One concern Highcharts may pose challenges when it comes to customization and integration with other libraries or frameworks.

ARCHITECTURAL MODEL

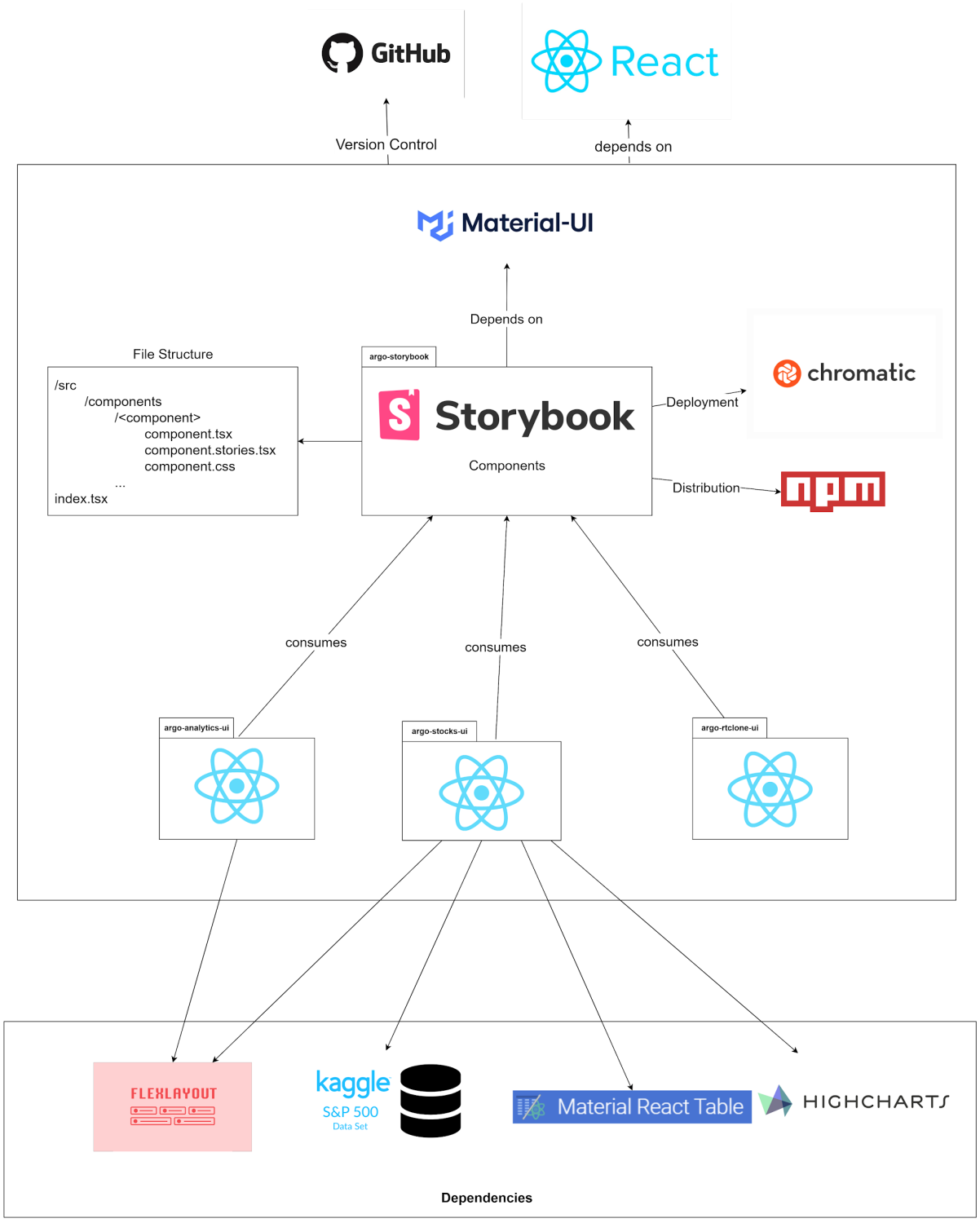


Figure 2. Architectural Model

RATIONALE

Our architectural model is a high-level abstraction of our project that represents its key components, their relationships, and the overall structure of the system. In our model we have included all technologies utilized in our project and their relationships, as well as how the interfaces will consume the components from the Argo-Storybook project. A software architectural model provides a common language for all stakeholders involved in the software development process, enabling them to communicate effectively and make informed decisions about the system's design and implementation. Because our model needs to be understood by all stakeholders, we chose to depict it in a more comprehensible format by adding icons to each package instead of using the plain package design. Additionally, since our project involves the development of 3 separate user interfaces, we chose to include each one as its own package in our model. Each interface has its own dependencies as well, so having each one in its own package allows us to accurately describe the dependencies of each one.

SOFTWARE

Storybook

Storybook is the main piece of software being used for this project. It is a tool used by developers to create, test, and showcase the components of a UI library project. In the context of a UX Master Library project, Storybook is used in the following ways:

- Component development: Storybook allows us to develop components in isolation, independent of the application or product they will eventually be used in. This helps to ensure that components are reusable, maintainable, and easy to use. Developers can create a new story for each component and test different use cases and states of the component.
- Collaboration: Storybook can be used as a collaboration tool between designers, developers, and other stakeholders. Designers can use Storybook to review and provide feedback on components, and developers can use it to showcase the progress of their work.
- Documentation: Storybook provides an easy way to document components, including their use cases, states, and props. This documentation can be shared with other developers, designers, and stakeholders to help them understand how to use and interact with the components.
- Showcasing: Storybook allows developers to showcase the components of a UX library project to other developers, designers, and stakeholders. This can help to demonstrate the value of the UX library project and encourage its adoption in other projects.

Storybook can be used to showcase the range of components available in the library, how they can be used, and how they look and feel in different contexts.
- Testing and debugging: Storybook provides a sandboxed environment where developers can test and debug components without having to navigate through the entire application or product.

React

ReactJS is a popular open-source JavaScript library used for building user interfaces (UIs) for web applications. It was developed by Facebook and is currently maintained by Facebook and a community of individual developers and companies. ReactJS allows developers to create reusable UI components and declaratively define the logic for how those components should render and behave in response to user interactions or changes in the application's state. [1] For our project, it is used in the following ways:

- Create the reusable UI components for ARGO. React works in conjunction with storybook by allowing the components to be used and modified within component stories.
- Develop mock websites to demonstrate the UX Component Library in action. The storybook components can be tested in isolation, however, building an application with them is the true test for whether the library is viable in the real world. React is used to build the prototype websites and the components are easily imported from the storybook library.

Material UI

Material UI is a popular React UI library that provides pre-built components and styles based on Google's Material Design guidelines. It is designed to help developers create modern, responsive, and intuitive user interfaces for web applications. Additionally, it offers a wide range of customizable UI components, such as buttons, text fields, dialog boxes, menus, and icons, which can be easily integrated into a React application. [2]

- In our case, most of ARGO’s components are built using Material UI with minor modifications to fit their company’s design philosophy. We will use Material UI as a basis for the Storybook library and make any modifications to it as necessary.

Highcharts

Highcharts is a JavaScript library that provides an easy way to create interactive and responsive charts and graphs for web pages or web applications. It allows developers to easily generate visually appealing and data-driven charts such as line charts, bar charts, pie charts, scatter plots, and more, using only a few lines of code.It also offers a wide range of customization options, including different chart types, colors, fonts, and animations. One of the main advantages of Highcharts is its ease of use and flexibility. The library is well-documented and has a large community of users and developers, which means that finding solutions to common problems or getting help with specific issues is relatively easy. [3]

- Highcharts will be used within the Stocks UI project to present historical stock data to the user.

FlexLayout

FlexLayout is being used as an alternative to Golden Layout. FlexLayout is a React library that provides flexible and efficient layout management for web applications. It is designed to simplify the process of creating complex and dynamic user interfaces by using flexible, responsive, and customizable layouts. With it, you can create dynamic layouts that adapt to the size of the screen or window, and that can be rearranged or resized by the user. FlexLayout also includes a number of built-in features and options that make it easy to customize the appearance and behavior of layouts to fit the UI mockups and component styles. [4]

Chromatic

Chromatic is a storybook built tool that makes it easy to publish, test, and document progress on the Storybook UX Master Library. We have integrated an automatic chromatic deployment via GitHub Action to hasten development and showcase how the library can function in a CI/CD environment. With it, we can easily demonstrate work done on the library and document past changes. [5]

HARDWARE

By the nature of our project, the hardware requirements are very lenient. To put it simply, our component library simply needs to run on a web browser; if the hardware supports modern web browsers, our library should function on it. (PC/Mac/Mobile). Our project uses cloud based solutions (Chromatic, NPM) to host the library/components, and no data needs to be explicitly stored within a database server.

DESIGN

GUI DESIGN

Stocks Analysis

Description

The stock page is designed to give users a dashboard that they can utilize to make stock trading decisions. The primary objective of this application is to aid users in making informed decisions on which stocks to trade or invest in. There is support for technical indicators and multiple stocks can be graphed on the chart at a single time. The user can choose which stocks to follow through the use of a watchlist. The user is also able to see their current positions and how much diversity is present in their portfolio.

(This is the theoretical scenario for what the UI is used for, however, implementing actual functionality may not be present in the final product; the goal is to have the overall layout)

Libraries

- Highcharts
- Flex Layout React
- Material React Table
- Kaggle stocks dataset

Components

- Button
- List
- Search
- Card
- Header
- Data Table
- Accordion

Mockup

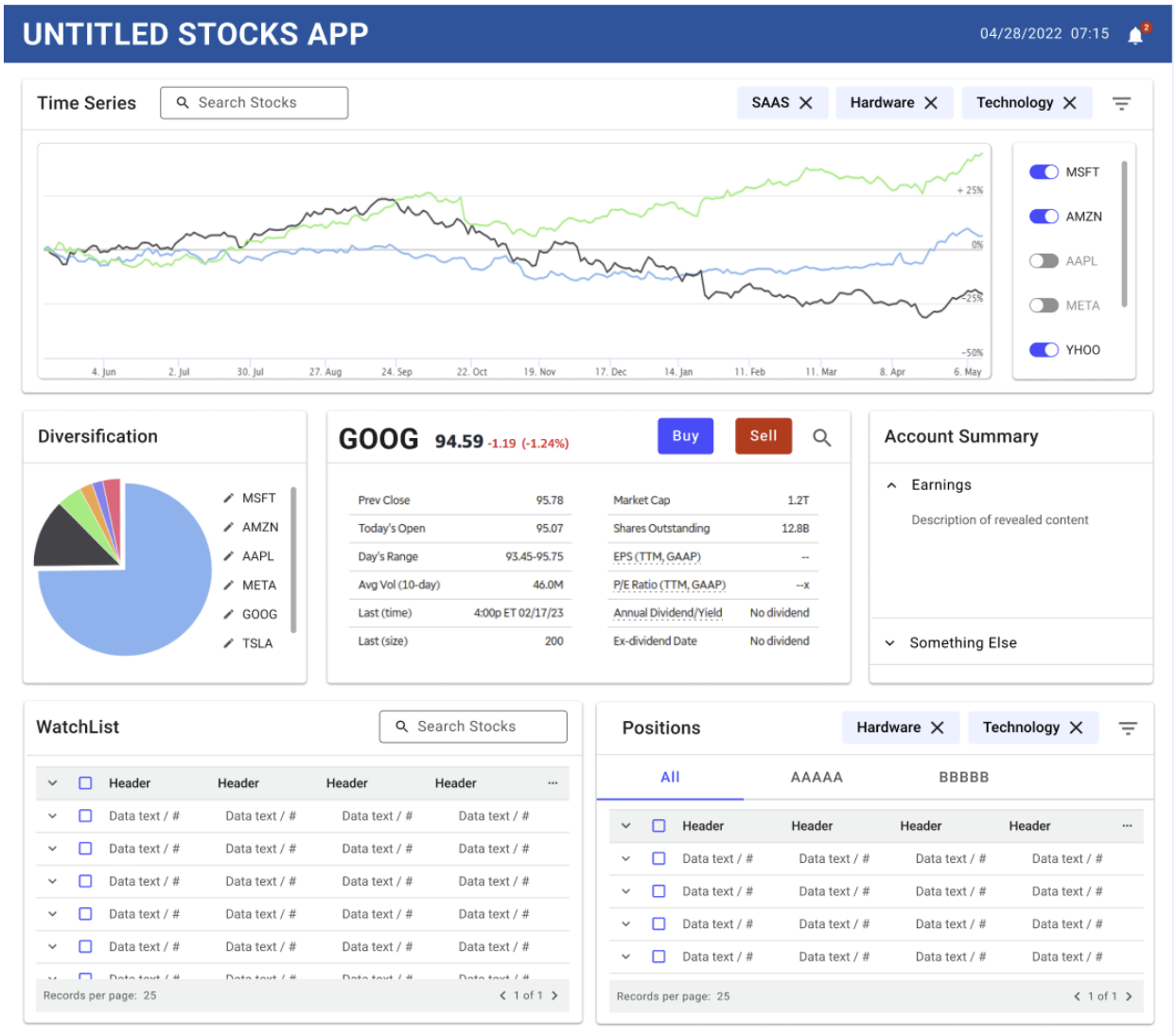


Figure 3. Stocks Analysis Mockup

Final Design

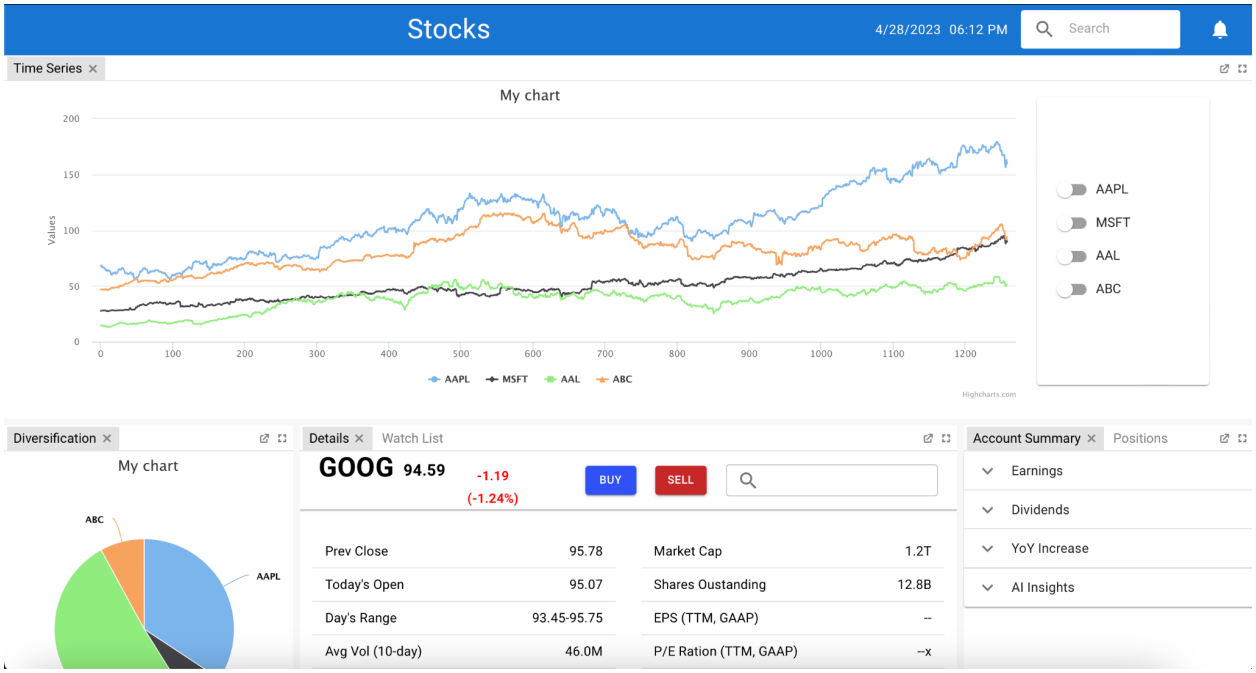


Figure 4. Stocks Analysis Final Design

References

[6]

Figure 5. Stocks Analysis Reference 1

2018 10 Best

Symbol	Bid	Ask	Mark Chg	Mark Chg %	Mark	Volume	Div. Per Share	Div. Yield	Ex-Div. Date
IBM	137.02	137.10	-0.72	-0.52%	137.02	3,975,115	1.64	4.7626	
ALLY	50.41	51.90	+0.29	+0.57%	51.03	2,351,405	0.25	1.9708	
ANTM	357.51	359.60	-13.92	-3.75%	357.51	1,902,977	1.13	1.2169	
AMAT	136.55	137.00	+1.84	+1.36%	136.84	9,054,701	0.24	0.7111	11/24/202
DAL	39.44	39.58	-1.73	-4.21%	39.36	12,501,215			
EPD	22.01	22.09	-0.12	-0.54%	22.09	3,815,804	0.45	8.1045	
PXD	150.50	151.08	+0.78	+0.52%	151.08	1,525,857	0.56	1.9907	09/29/202
USFD	30.00	33.00	-0.61	-1.85%	32.37	1,608,328			
BRK.B	277.45	277.70	-1.01	-0.36%	277.60	4,554,095			

StockBrokers.com

[7]

Figure 6. Stocks Analysis Reference 2

Social Media Analytics

Description

The social media analytics page is designed to give users a comprehensive understanding of their social media activity. The primary objective of this application is to assist users in evaluating the impact of their social media content on their audience, identify the type of content that resonates with their followers, and develop a more effective social media strategy. The dashboard provides clear and detailed information on various aspects of the users' online media presence, including the performance of their content. The user may choose what aspects they find most important and organize the UI however they best see fit. It's important to note that the site will utilize simulated data and will not have any actual connection to any real accounts.

(This is the theoretical scenario for what the UI is used for, however, implementing actual functionality may not be present in the final product; the goal is to have the overall layout)

Libraries

- React Circular ProgressBar
    - Display percentage via radial bar
- Framer Motion
    - Animate cards
    - Reorder cards
- MUI Icons

Components

- Header
- Left Navigation Menu
- Button
- FilterChip
- Card

Mockup

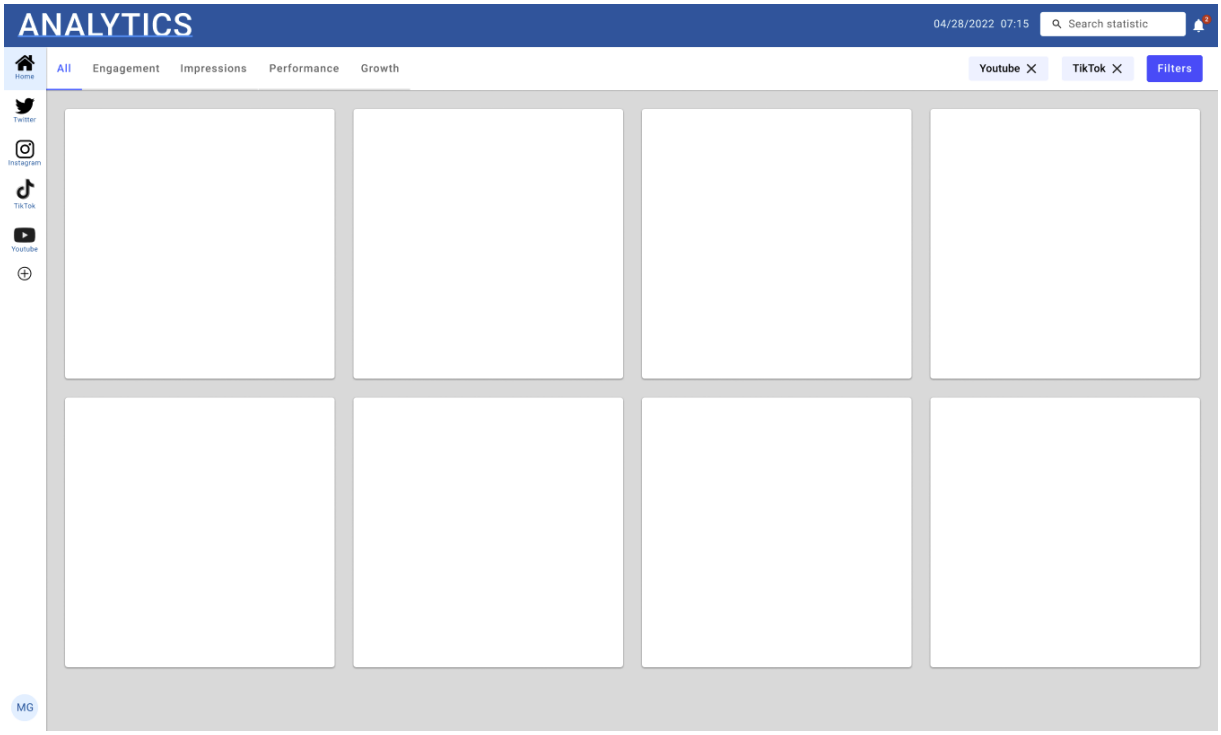


Figure 7. Analytics Mockup

Final Design

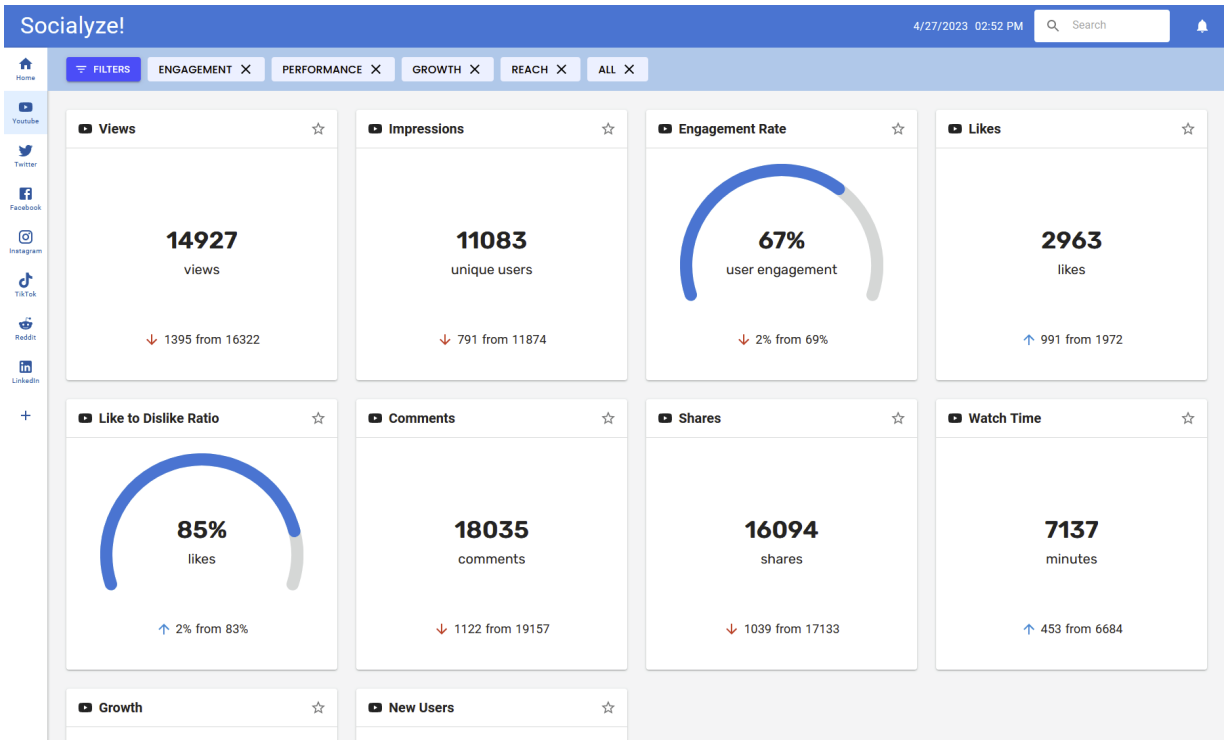
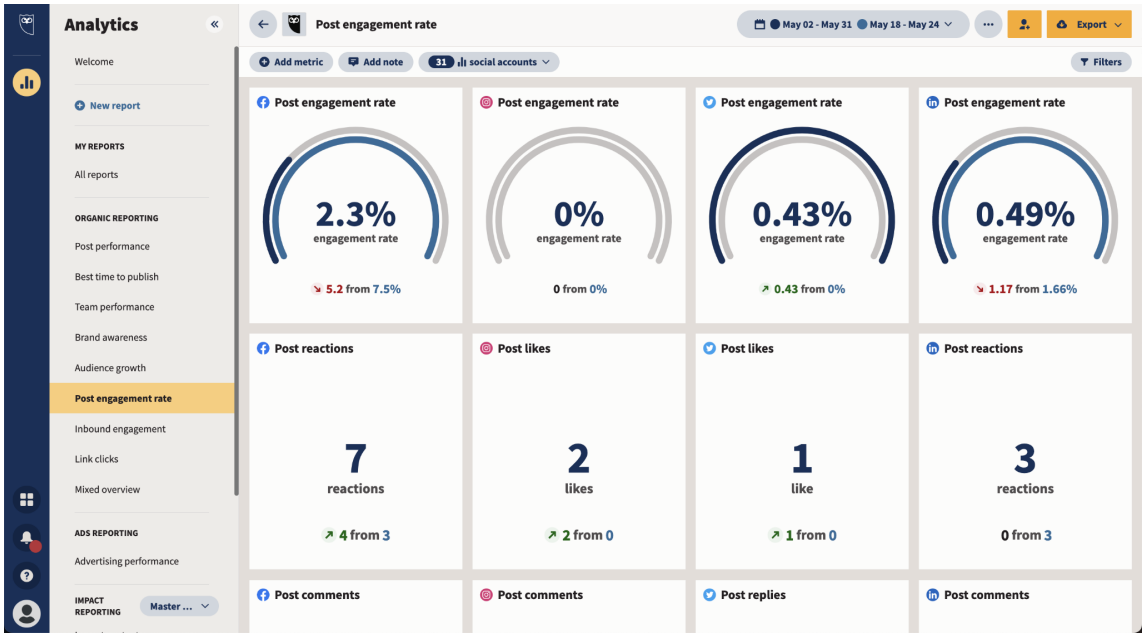


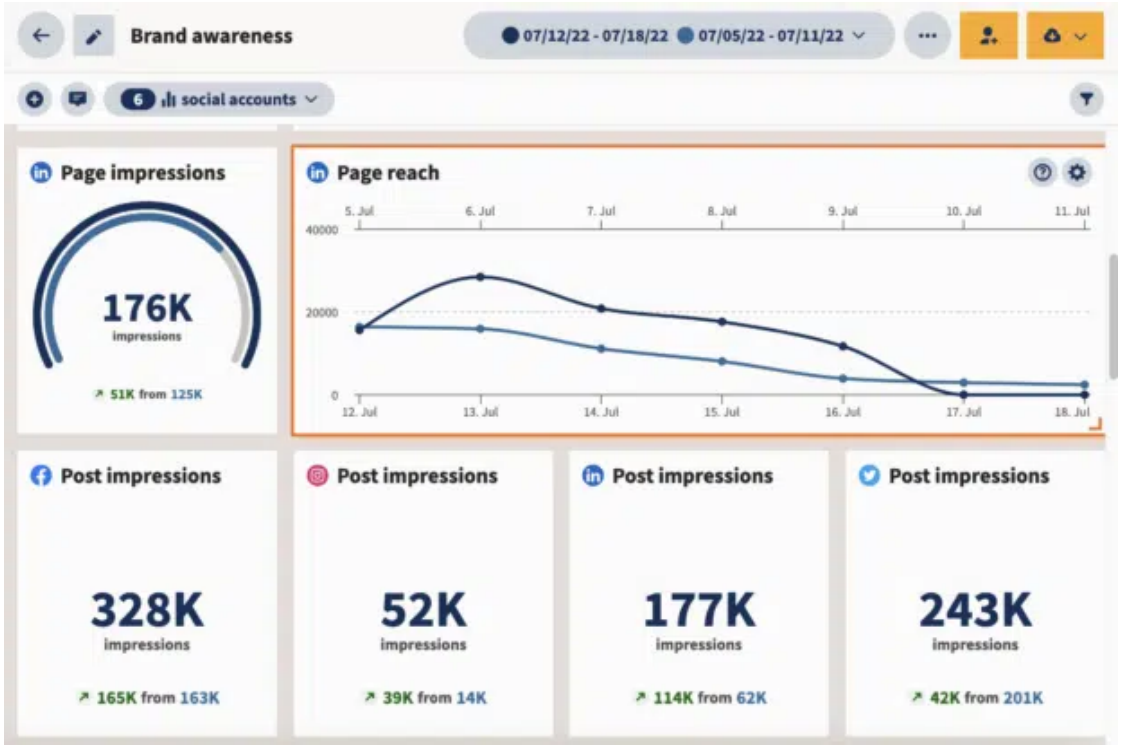
Figure 8. Analytics Final Design

References



[8]

Figure 9. Stocks Reference 1



[8]

Figure 10. Stocks Reference 2

Rotten Tomatoes Clone

Description

The design mockup for the rotten tomato clone is a visually appealing and user-friendly platform for movie and tv show enthusiasts. The main focus of the page is a list of the top movies in a particular category, displayed in a neat and organized manner. Each movie is represented by its poster and basic information such as title, score, directors, and description.

To the right of the movie list, there is a form for users to write their own reviews about a movie or tv show of their choice. The form is simple and straightforward, allowing users to rate the movie/tv show and write a review. At the bottom right of the page, there is a news panel. This section showcases a list of images with their corresponding text.

(This is the theoretical scenario for what the UI is used for, however, implementing actual functionality may not be present in the final product; the goal is to have the overall layout)

Components

- Button
- Badge
- Radio
- Search
- Dropdown
- Text Field
- Form
- Header
- Checkbox

Mockup

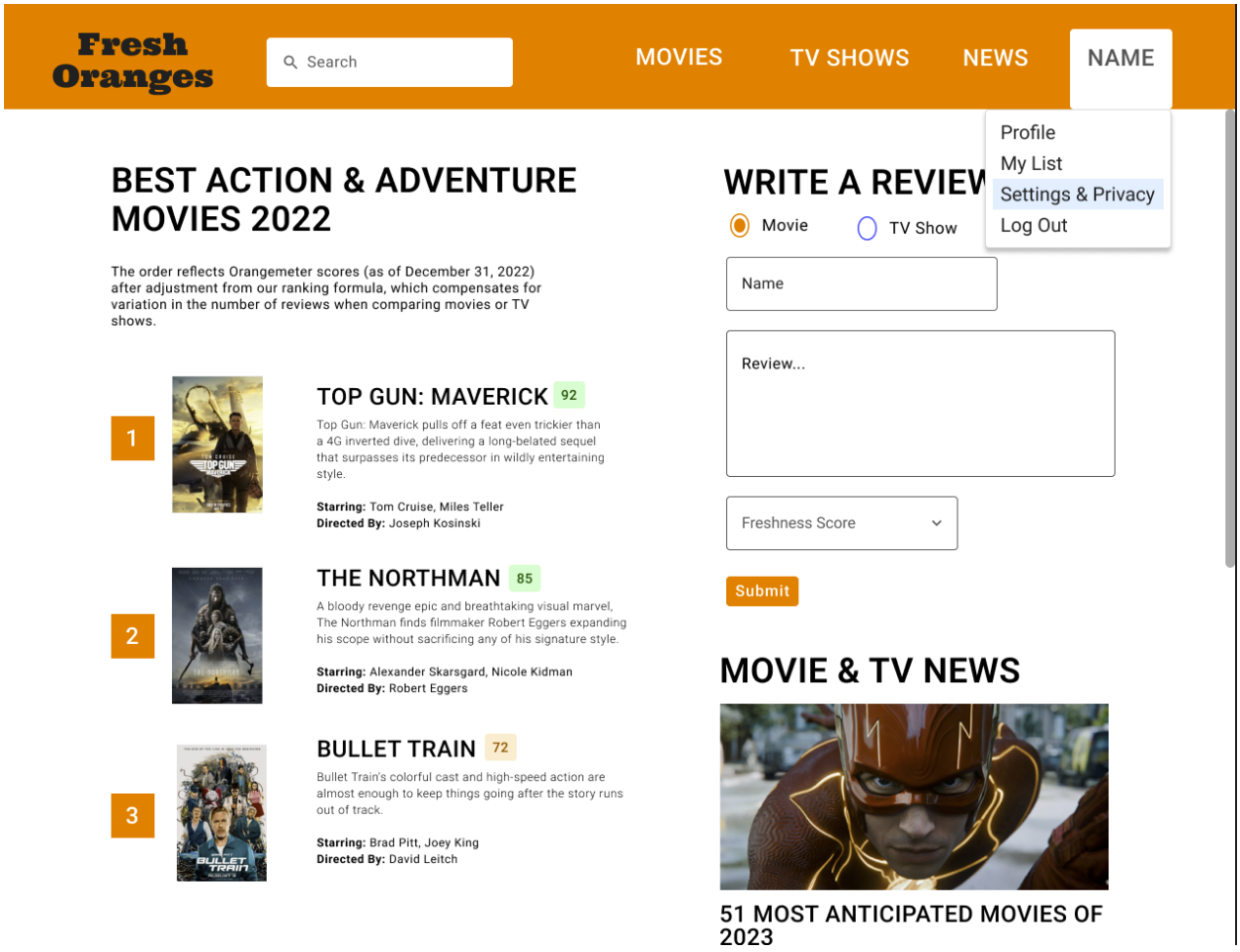


Figure 11. Fresh Oranges Mockup

Final Design

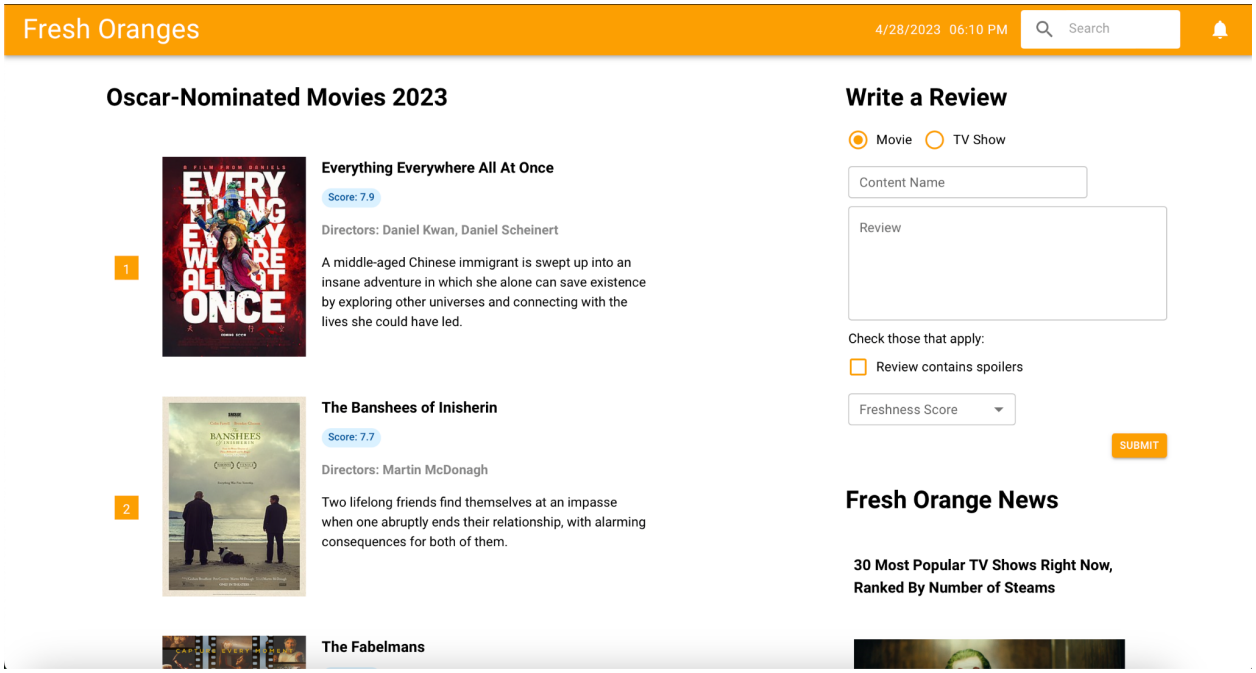


Figure 12. Fresh Oranges Final Design

References

[9]

Figure 13. Fresh Oranges Reference 1

STATIC MODEL

Component Class Diagram

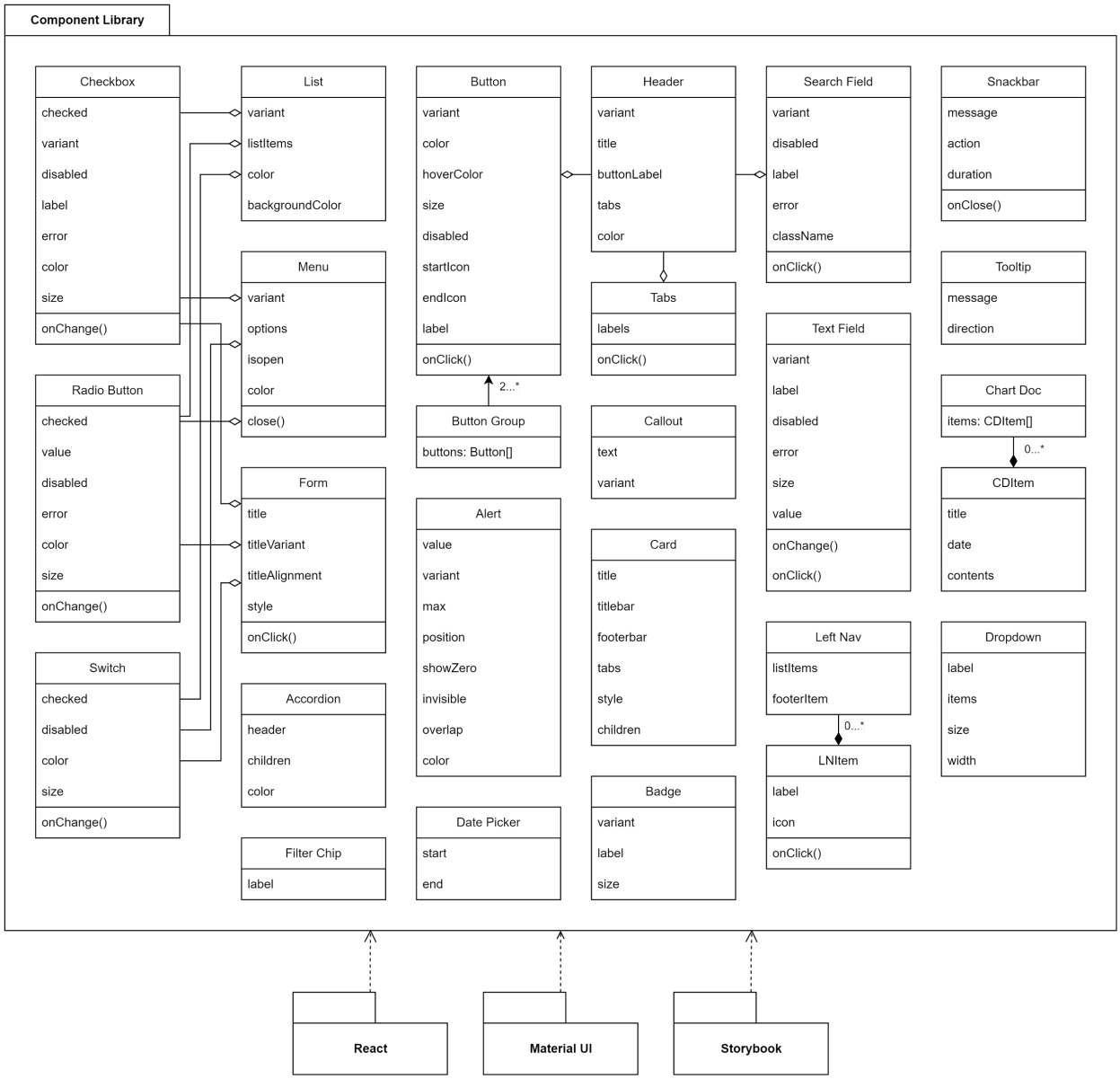


Figure 14. Component Class Diagram

DYNAMIC MODEL

Stock UI Sequence Diagram

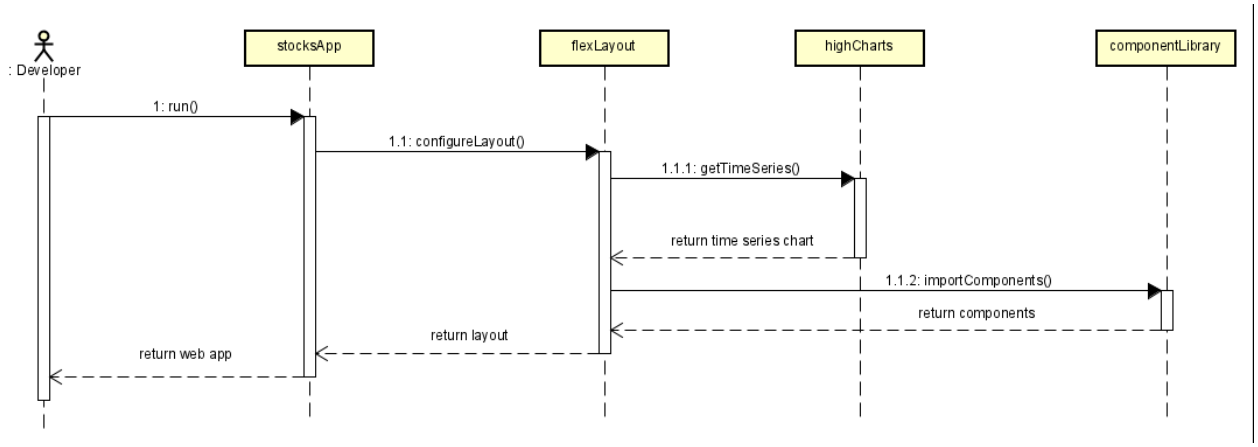


Figure 15. Stocks Analysis Sequence Diagram

Fresh Oranges Clone Sequence Diagram

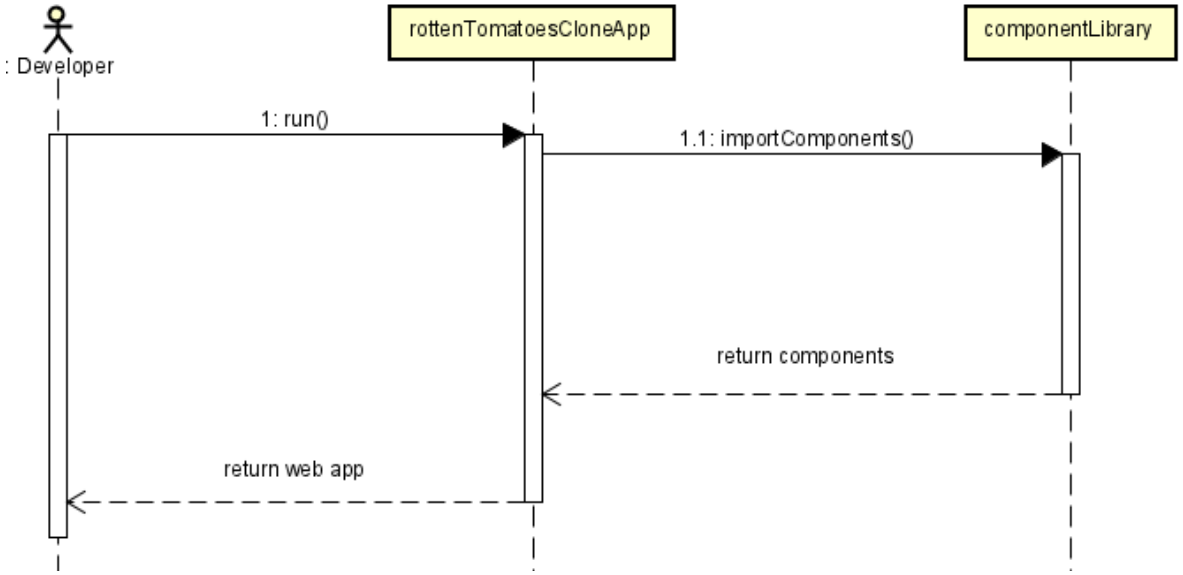


Figure 16. Fresh Oranges Sequence Diagram

Social Media Analytics Sequence Diagram

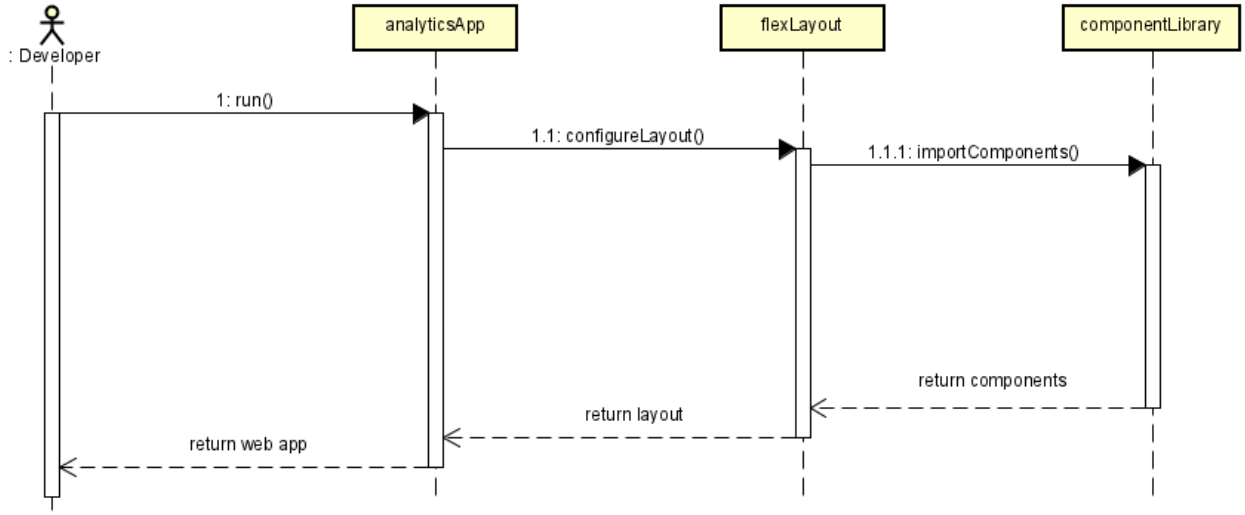


Figure 17. Analytics Sequence Diagram

RATIONALE FOR DETAILED DESIGN MODEL

The detailed design models used are the UML class diagram and sequence diagram. The following provides more information on what these diagrams show about the architecture of the project:

- **What is a UML Class Diagram**? It consists of a set of classes, which represents the objects within the system and the relationship between these objects. A class, visualized in a rectangle shape, contains a set of attributes and methods that define the structure and behavior of that class. Relationships between classes are represented by arrows connecting them. A class diagram is meant to show a *static* view of the system.
- **What is a UML Sequence Diagram**? It shows the interactions between the objects in the system in a sequenced manner. They are often used to model the flow of messages or events between objects in the system. Each object in the system is represented with a vertical line (it's "lifespan") and interactions are labeled on horizontal lines between lifespans. These horizontal lines have a direction, indicating the flow of the message. A sequence diagram is meant to show a *dynamic* view of the system.

The **rationale** to *why* these two models were chosen to design the architecture of the project includes the following:

- **UML Class Diagram:** the component library comprises primarily of objects; each component, like a button for example, has a set of attributes associated with it (label, size, color) as well as a set of methods (onClick()). Therefore, it's important to think of the components in the library as objects. Building a class diagram provides a high-level view of the components to be built and how they are related to each other, making it easier for us to understand and build them. In addition, a high-level view helps in designing with modularity in mind; we can identify common patterns among the components and make them more standardized and modular.
- **UML Sequence Diagram**: a sequence diagram sheds light into events and workflows that may not have been captured by the class diagram. The sequence diagrams provided in this documentation show the interactions between the developer, the specific UI application, the component library, and other necessary libraries used such as HighchartsJS. This is helpful in visualizing how users interact with the system and how the system elements interact with each other in order to fulfill user goals. The diagrams provided focus primarily on high-level interactions rather than low-level because the project's emphasis on the component library and its use within the three UIs do not require substantial workflows relevant to the architecture of the project. Therefore, the diagrams provided are a way to describe at a high-level how the libraries present in the architecture are utilized in the design level.

REQUIREMENTS TO DESIGN TRACEABILITY

1. Usability
    a. The system shall provide easy access for creating/viewing/modifying/ deleting components.
        i. The Storybook Class Component provides the services for creating/viewing/modifying/deleting components
    b. The system shall provide the ability to easily use its components within other projects.
        i. The React Class Component allows the system to consume components within React applications
    c. The system shall have documentation for each component deployed.
        i. The Storybook Class Component provides a framework for providing documentation for each component
2. Maintainability
    a. The system shall ensure that components are upgradable in the future.
        i. The Storybook Class Component provides the ability to upgrade components
    b. The system shall house all components within the same repository.
        i. The Storybook Class Component houses all components within the same Github repo.
    c. The system shall be built using proper design standards.
        i. The Material UI Class Component houses its own peer-tested design standards.
    d. The system shall be built using open source libraries.
        i. Open source libraries include the Material UI, React, and Storybook Class Components
    e. The system shall automatically deploy new changes to the library.
        i. Each Component Class is deployed via a CI-CD pipeline via the Storybook Class Component repo.
3. Extensibility
    a. The system shall be able to accept the creation of new components.
        i. The Storybook Class Component provides the ability to upgrade components
    b. The system shall be able to allow the modification of old components.
        i. The Storybook Class Component provides the ability to upgrade components
    c. The system shall feature components with easily modifiable parameters.
        i. Each Component Library class houses it’s own parameters which are consumable by users of the component.
    d. The system shall propagate changes across component implementations.
        i. The Storybook Component Class handles versioning within its repo.

TESTING PLAN

SYSTEM TEST CASES

UI Test Cases

ID	Component	Test Cases
C1	Accordion	A. Title is displayed properly B. Contents shows/hides on click
C2	Card	A. Title is displayed properly B. Titlebar displays custom content properly C. Body content is displayed properly D. Footerbar displays custom content properly E. Card displays tabs properly F. Apply custom styles properly
C3	Form	A. Title is displayed properly according to variant B. Displays correct button color based on color prop C. Width of form component rendered correctly D. Padding around form component rendered correctly E. Individual form components rendered correctly in the right sequence
C4	Header	A. Title is displayed properly B. Title font weight adjusts title properly C. Searchbar is aligned properly D. Tab is aligned properly E. Tab displays correct custom labels F. Button is aligned properly G. Menus are aligned properly H. Menu options are displayed correctly I. Time is updated accurately J. Each tab is mapped to open its respective menu
C5	Image	A. Text rendered correctly for appropriate variant B. No text rendered for image only variant C. Image aligned correctly according to variant D. Image size displayed according to height and width props E. Header text displayed with black and bold font F. Subheader text displayed with gray font G. Body text displayed with black font H. Text aligned correctly according to align prop I. Padding around image component rendered correctly J. If available, badge rendered correctly in blue color scheme

C6	Login	A. A login form is displayed B. Username input accepts text C. Password input accepts text D. Project title/logo is displayed properly above the inputs E. Login button is aligned to bottom right of the card F. Changing the accentColor is reflected properly
C7	LoginPage	A. Login component is rendered in the center of the page B. Login component properly displays title/logo C. Properly displays different background color properties D. Properly displays different accentColor properties
C8	Alert	A. Alert icon properly renders on top of child component B. Value is properly displayed within the icon C. Icon can be aligned to the top/bottom of child component D. Icon can be aligned to the left/right of child component E. Value shows + if above the designated max F. Icon is hidden if showZero is enabled while value is 0 G. Icon background displays correct color scheme H. Value text color displays correct color scheme
C9	Tooltip	A. Tooltip displays correctly B. Label displays correctly C. Tooltip can be moved D. The conditions to display the tooltip can be modified
C10	Badge	A. Displays correct color scheme for each variant B. Displays correct label
C11	Button	A. Label is displayed properly B. Variant changes how the button is displayed C. The color of the button is customizable D. Button can be disabled
C12	Checkbox	A. Checkbox is unchecked by default B. Clicking on checkbox changes state C. Displays correct label by checkbox D. Displays correct color of checkbox E. Medium-sized checkbox by default F. State is not disabled by default G. Renders checkbox and label font color as gray when disabled H. Unable to change state when disabled
C13	Dropdown	A. Displays correct label B. Drops down user-specified items in list C. Supports 2 different sizes D. Allows custom width to be set

C14	FilterChip	A. Value is properly displayed within the chip B. Clicking on the X triggers the onClose function C. Clicking elsewhere on the chip triggers the onClick function
C15	List	A. Displays correct list based on variant B. List is just text by default C. Renders list items correctly in sequential display D. Displays correct button color for checklist/ radiolist/ toggle list E. Displays correct background color F. Radio list allows for selection of only one value at a time G. Checklist/radiolist/toggle list buttons work correctly based on state changes
C16	Radio	A. Able to change checked or unchecked state through props B. Correctly renders correct value of radio button C. Displays correct color of radio button D. State is not disabled by default E. State is not error by default F. Renders radio button gray when disabled G. Unable to change state when disabled H. Renders radio button red when error I. Medium-sized radio button by default
C17	SearchBar	A. Displays text field with search icon correctly B. Displays outlined search bar by default C. Changes search bar display based on variant D. Displays correct label in search bar E. Renders search bar gray when disabled F. Unable to change state when disabled G. Renders search bar red when error H. State is not disabled or error by default
C18	TextField	A. Displays correct label if available B. Displays correct helper text if available C. Allows multiple rows of text using multiline and rows props D. Renders text field gray when disabled E. Unable to change state when disabled F. Renders text field red when error G. State is not disabled or error by default H. Displays error when length surpasses maxLength prop I. Changes margin appropriately based on margin prop value
C19	Toggle	A. Displays correct color of toggle button B. State is not disabled by default C. Renders radio button gray when disabled D. Unable to change state when disabled E. Medium-sized toggle by default

C20	LeftNav	A. Nav takes full height of parent container B. All list items are displayed properly C. ListItem icon is displayed properly and is centered D. ListItem text is displayed properly below the icon E. Clicking any ListItem triggers the onChange function F. Clicking any ListItem displays the selection accordingly G. Footer button is displayed at the bottom of the nav H. Clicking the footer button triggers the onFooterClick function
C21	Menu	A. Displays the provided options as menu items B. Displays the correct number of options C. Menu closes when the backdrop is clicked D. Default menu closes when a menu item is clicked E. Displays the correct component next to each menu item F. Displays the correct component color
C22	Tab	A. The number of tabs is equivalent to the number of labels B. Renders the correct label for each tab C. Sets the active tab when a tab is clicked D. Displays no tabs when the labels array is empty E. The selected tab has proper indicator styling
C23	Table	A. Supports user-defined columns B. Supports user provided rows C. 3 density values of “comfortable,” “spacious”, and “ “compact” D. Enabling and disabling of various visual elements of the table

Table 16. UI Test Cases

System Test Cases

ID	Test case
T1	Importing and using components in all of the user interfaces
T2	Importing components and passing props to them to customize behavior
T3	Workflow of making change, pushing it, making release, sending to NPM, and installing the newest package version in UIs.
T4	Component is created and sent to the storybook library
T5	User accesses the deployment and is able to view their desired component

Table 17. System Test Cases

TEST CASE TO USE CASE TRACEABILITY

A mapping of UI and System test cases to System use cases.

Test Case	System Use Case
ALL UI test cases	Add all system use cases
T1	UC1: Use Component
T2	UC2: Customize Component
T3	UC3: Refactoring Component
T4	UC4: Create Component
T5	UC5: View Component

Table 18. Test Case to Use Case Map

TEST GENERATION TECHNIQUES

First, tests are generated by the appearance of components in the Figma design document [10]. The library component is visually compared to its reference in the company-provided Figma document. These tests check to make sure that the component bears sufficient resemblance to the provided reference. A component fails the test if it cannot be easily identified or matched to its reference.

Second, tests are generated via the component's parameters. Each component has a number of parameters that can be modified. Tests are created for each parameter to test for both clarity of naming and ensuring it functions as intended (ex. A "color" parameter should change the color of the component). A parameter fails the tests if its name does not convey its function, or if it does not function properly.

TEST EVALUATION CRITERIA

No formal criteria was defined to evaluate the correctness of the test cases. We simply developed a rough list of cases for each component and attempted to ensure that the components met their respective list.

Testing was not a major concern in developing this design system because the system was meant to be a proof of concept for the actual system in the future.

ACKNOWLEDGEMENT

Special thanks to the following individuals:

- ➢ Eric Wong
- ➢ Dongcheng Li
- ➢ Zizhao Chen
- ➢ Mark Bentsen
- ➢ Ponchai Reainthong
- ➢ Raisa Gonzalez

REFERENCES

[1] React, “React – a JavaScript Library for Building User Interfaces,” *Reactjs.org*, 2022. https://reactjs.org/

[2] MUI, “MUI: The React component library you always wanted,” *mui.com*, 2023. https://mui.com/

[3] Highcharts, “Interactive JavaScript charts for your webpage | Highcharts,” *www.highcharts.com*, 2023. https://www.highcharts.com/

[4] Caplin, “FlexLayout,” GitHub, Feb. 21, 2023. https://github.com/caplin/FlexLayout#readme (accessed Feb. 25, 2023).

[5] Chromatic, “Automatically review, test, and document Storybook,” *www.chromatic.com*. https://www.chromatic.com/ (accessed Mar. 01, 2023).

[6] “Online Trading Platforms & Tools | TD Ameritrade,” *www.tdameritrade.com*. https://www.tdameritrade.com/tools-and-platforms.html (accessed Feb. 13, 2023).

[7] S. Levine, “7 Best Online Brokers 2020,” *StockBrokers.com*, 2023. https://www.stockbrokers.com/guides/online-stock-brokers

[8] Hootsuite, “Social Media Marketing & Management Dashboard,” *Hootsuite*, 2021. https://www.hootsuite.com/ .

[9] Rotten Tomatoes, “Best Action & Adventure Movies 2022,” *Rotten Tomatoes*, 2022. https://editorial.rottentomatoes.com/guide/best-action-adventure-movies-2022/

[10] ARGO. “UTD BHCC Design Library.” *ARGO*. https://www.figma.com/file/vyxYsmCyu5jTVY7wBpkG25/UTD-BHCC-Design-Library-(Copy)-(Copy)?node-id=6%3A2&t=6q0hlEemn9xrZG5o-0 .