Architecture Documentation

ARGO UX Master Component Library

ABSTRACT

This architecture document provides a comprehensive high-level overview of the architecture utilized in the project. It encompasses the three user interfaces, as well as the UX Master Component library. To showcase the project architecture, the document presents the architectural patterns, a model, and a description of the software and hardware. An extensive analysis of the significance of each aspect of the architecture is presented to showcase an in-depth understanding of how the architecture manages the system quality attributes.

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

TABLE OF CONTENTS

LIST OF FIGURES

List Of Images

INTRODUCTION

This document details and outlines the Architecture of the UX Master Component Library and three sample UIs that will employ the Master Component Library. The purpose of this document is to provide an overview of the Architecture used for the project. This document includes the Architectural Styles, Architectural Model, a description of the technology, software and hardware used in the project and how they are used, and a rationale for the above. This document does not cover implementation details.

This document first lists the Architectural Styles used, and a brief description is provided for each. Then, there is a diagram of the Architectural Model employed for the Library as well as the three sample UIs. Afterwards, there is a listing of each technology, software, and hardware employed, with explanations of each and justification for why it is employed. Finally, the Rationale for each of the above decisions is elucidated. This section further delves into the thought process behind each decision, as well as lays out potential challenges and roadblocks that may be faced in implementation.

ARCHITECTURAL STYLE(S) USED

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

Our first UI implementation, the Stock interface, uses a 3 tier serverless architecture consisting of the presentation layer made with React.js, serverless functions deployed in the cloud, and data provided by third-party APIs. This architecture does not include user authentication or storage. Highcharts is used to visualize the stock-specific data.

ARCHITECTURAL MODEL

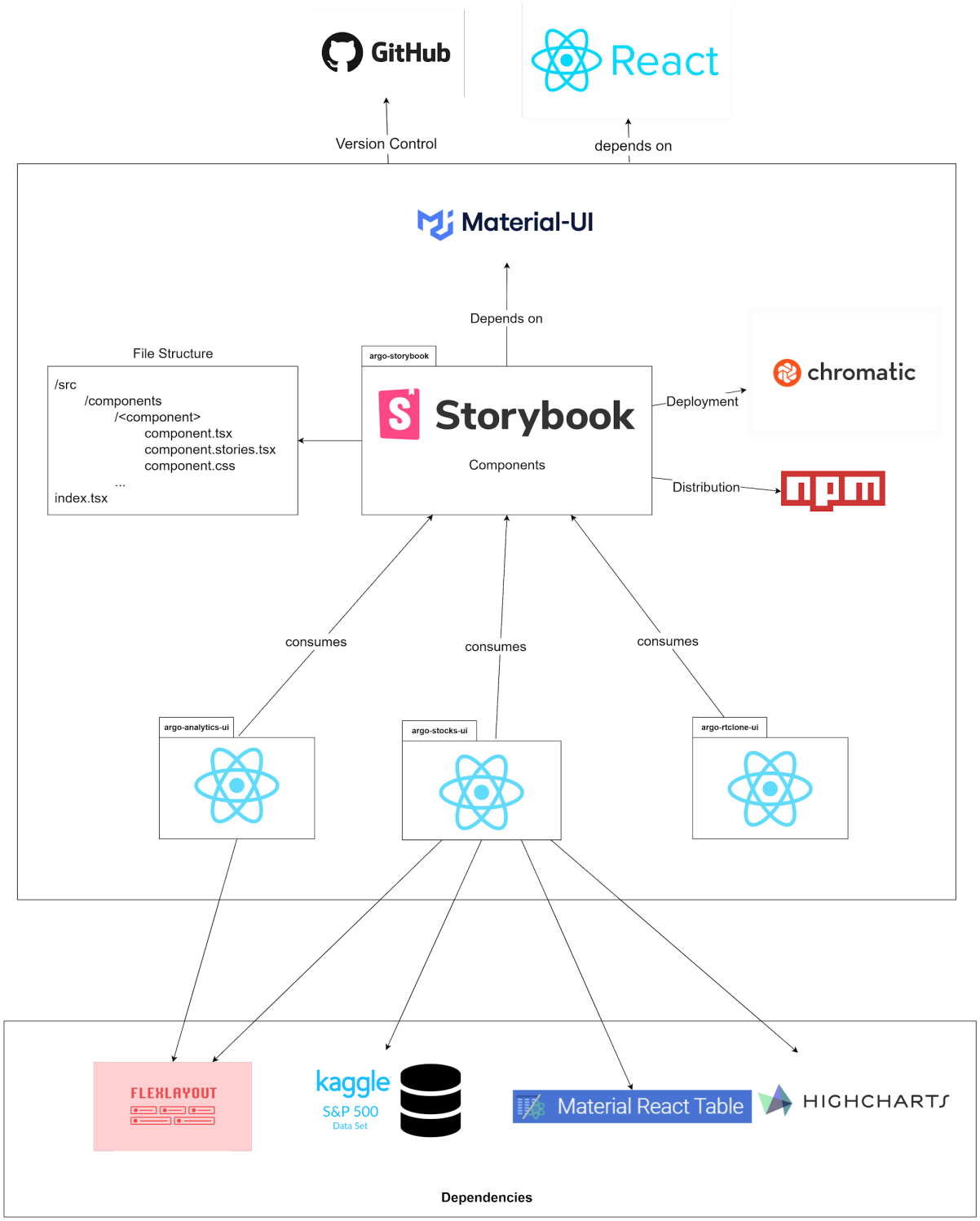

Figure 1. Architectural Model

TECHNOLOGY, SOFTWARE, AND HARDWARE

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.

RATIONALE FOR ARCHITECTURAL STYLE AND MODEL

Architectural Style

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 3-tier architecture with serverless computing allows us to separate the presentation layer, business logic 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 3-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. Since serverless architectures rely heavily on third-party services, downtime or outages of these services can impact the application's performance and availability, which is another concern. An important constraint for our implementation is the need for a reliable and stable network connection, as the application relies on cloud-based resources that are accessed over the internet. This can lead to issues with latency and connectivity, which may affect the user experience.

Architectural Model

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.

Technology, Software, Hardware

Storybook

Our team chose to use Storybook because it offers several benefits that can help optimize the development process. By allowing developers to work on individual UI components in isolation, Storybook enables faster development times. This is important because our project has tight time constraints. Storybook also promotes component reusability, making it easier to share and reuse UI components across different projects. This can save time and effort over time, as developers can leverage existing components rather than recreating them from scratch. This also ensures consistency across different parts of the application, improving the overall usability.

Roadblocks, Issues, Concerns: One concern is maintaining consistency across different components when working with a large team or multiple contributors. Multiple people editing the components at once may lead to discrepancies, which can be confusing for users and make the application harder to use. One constraint is that Storybook requires developers to create and maintain separate stories and components, which can add additional overhead and complexity to the development process. Another constraint is that Storybook may not integrate well with certain technologies or frameworks, which can limit its usefulness for some projects.

React

Our team selected the React JavaScript library because it supports the creation of reusable UI components which reduces redundant code and improves maintainability. React also improves the usability of web pages by providing a fast, responsive, and intuitive interface. Additionally, most members of our team have experience with React which increases our development efficiency.

Roadblocks, Issues, Concerns: One roadblock is compatibility issues with different versions of React or other libraries, which can lead to conflicts and bugs in the application. Once concern is that React can be performance-intensive, especially when handling large amounts of data or complex user interfaces. This can lead to slow rendering times and an overall sluggish user experience. Additionally, React has certain constraints, such as its reliance on the Virtual DOM and its one-way data flow architecture, which can make it challenging to integrate with other libraries or frameworks.

Material UI

The decision was made to use Material UI because it helps speed up development by providing a library of pre-built components that can be easily customized and integrated into the application. This can reduce development time and costs, while also improving the consistency and quality of the user interface. Material UI also offers a range of features, such as responsive design, and accessibility which can improve the usability and accessibility of the application.

Roadblocks, Issues, Concerns: One concern is the library's large size, which can impact application performance if not optimized correctly. Another is that developers may face compatibility issues with Material UI and other libraries, especially when using specific versions of React or other dependencies. Another concern is the long-term sustainability and maintainability of the codebase when relying heavily on pre-built components from a third-party library.

Highcharts

Our team chose to use Highcharts because it provides a wide range of chart types and options for customization. This allows for the creation of engaging and interactive data visualizations that effectively communicate insights and trends to users. Additionally, Highcharts supports interactive features such as tooltips, zooming, and panning, which can help users explore and analyze data in more detail. This can improve the usability of the chart by providing users with more control and flexibility in how they interact with the data. Highcharts can also be easily integrated with the JavaScript React library.

Roadblocks, Issues, Concerns: One issue with Highcharts is the cost of licensing. As the project scope increases, it may be required to purchase a more expensive license type to support the application. Another concern is the library's large file size, which can impact application load times and performance. There may be compatibility issues with certain browsers or mobile devices, which can impact the user experience.

FlexLayout

Our team has decided to use FlexLayout due to its support for CSS flexbox properties, which can help developers create complex layouts with minimal effort. This makes it easier to implement layouts that respond to different screen sizes and orientations, improving the usability and accessibility of the application. FlexLayout also supports drag-and-drop layout editing, allowing developers to easily modify and customize layouts as needed. This can help improve development efficiency and reduce development time. Additionally, FlexLayout provides a simple and intuitive API for defining layouts, making it easier for developers to learn and use the library effectively.

Roadblocks, Issues, Concerns: One concern is the potential for layout conflicts, as FlexLayout may not work well with certain browsers or older devices that do not support the Flexbox model. Another concern is the maintainability of code that relies heavily on FlexLayout, as it may require more effort to modify or update the layout as the application evolves.

Chromatic

Our team decided to use Chromatic because it integrates directly with Storybook, allowing developers to automatically capture screenshots of UI components and compare them against previous versions to identify any visual changes. Chromatic also provides automated UI testing, allowing developers to catch issues and regressions early in the deployment process. Additionally, Chromatic provides a collaborative review process, allowing team members to provide feedback and review changes to UI components in real-time.

Roadblocks, Issues, Concerns: One issue is the cost, as Chromatic requires a paid subscription for teams or projects that exceed a certain usage threshold. (This does not directly apply to us because we are a small team, however, if more people were to work on the library it becomes an issue. Additionally, the privacy and security of project code and design assets when using a third-party tool like Chromatic is another concern.

CONFIGURATION MANAGEMENT

Version In	Version Out	Changes	Reviewed By
0.0	1.0	Initial Document Creation	All Group Members

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).