SE 4485: Software Engineering Project

Spring 2024

Architecture Documentation

Group Number	3
Project Title	Storybook POC Continuation with Chromatic and Storybook GPT
Sponsoring Company	ARGO
Sponsor(s)	Ponchai Reainthong, Kevin Roa, Raisa Gonzalez, Mark Bentson
Group Members	1. Lillie McMaster 2. Lauren Nguyenphu 3. Alina Khan 4. Hamdiya Abdulhafiz 5. Iman Sheriff 6. Timothy Naumov

ABSTRACT

This document outlines the architecture for the Storybook POC Continuation Project, sponsored by ARGO for Group 3. This document serves as a guide for the technology, software, and hardware utilized, architecture models and styles employed, and traceability from requirements to architecture. The project incorporates various subsystems, including Component Library, GPT-4, Github, Design Library, and more. The subsystems interactions are outlined to facilitate component development and integration. The document concludes with mapping functional and nonfunctional requirements to their respective architectural components, which provides an overview of the project's objectives.

Table Of Contents

List Of Figures

List Of Tables

Introduction

This document serves as the Architecture for the Storybook POC Continuation project. This project serves as a Proof of Concept for the ARGO team to confirm that Storybook GPT would be a beneficial investment for ARGO to continue to make. The document's purpose is to outline the architecture used in the project, and does not cover implementation details.

This document first describes the technology, software, and hardware used for implementing the Storybook POC Continuation project. Then, the document provides the architectural model and styles used, along with its corresponding rationale. Lastly, the traceability from requirements to architecture is provided to show a mapping between requirements and architecture. The structure of the document can be noticed within the table of contents. It follows the suggested structure given by the Professor for the course.

Technology, Software, And Hardware Used

This project requires a variety of technologies to be used together across the system. We will identify several perspectives where different entities will interact with the technology stack differently. The technologies grouped by system are listed below:

System	Technology
Component Library	TypeScript, React, Storybook, MUI, VS Code, GitHub, Figma
Storybook GPT	GPT-4, Storybook, React, MUI
GitHub Actions (CI/CD)	npm, semantic-release, Storybook, Chromatic, GitHub-hosted runners, Rollup

Table 1. Technology, Software, and Hardware Used

Component Library:

The component library requires the use of several technologies that work in tandem to export components into a package. The component's are written in TypeScript with React and we are using MUI components as a base. We use VS Code as our development environment and GitHub as our version control. Each component has a story written for it using Storybook which should adhere to design specifications defined in Figma.

Storybook GPT:

The GPT uses GPT-4 for the LLM and we have provided many existing components and stories for context which are written using React with MUI as well as stories through Storybook. The component library developers interfaces with the StorybookGPT through ChatGPT.

GitHub Actions (CI/CD):

When pull requests are merged into the master branch, a GitHub Actions workflow is triggered to release the latest updates to npm and Chromatic. These workflows are executed through the GitHub-hosted runners which are virtual machines. When building and releasing the latest component library, the new package has to pass visual tests before the package is built using Rollup and then released using semantic-release. Semantic-release updates the package version automatically and then publishes the latest components to npm. Additionally, another workflow is triggered which builds the latest Storybook documentation and deploys it to the hosted instance using Chromatic.

Communication between application server and database server:

While our project does not involve direct communication between an application server and a database server, we are leveraging tools that interact with database servers. One instance of a database server is publishing our components to npm. When releasing the package in the GitHub Actions workflow, the built component library is sent to be stored in the npm registry (which could be considered a persistent storage). Additionally, when the Storybook documentation is deployed using Chromatic, the static files are hosted on a web server to be distributed when users access the documentation. Although this is not considered a database server, there is a communication in our workflows that releases our latest changes to another system that stores the latest version of the component library as well as the documentation.

Architectural Model

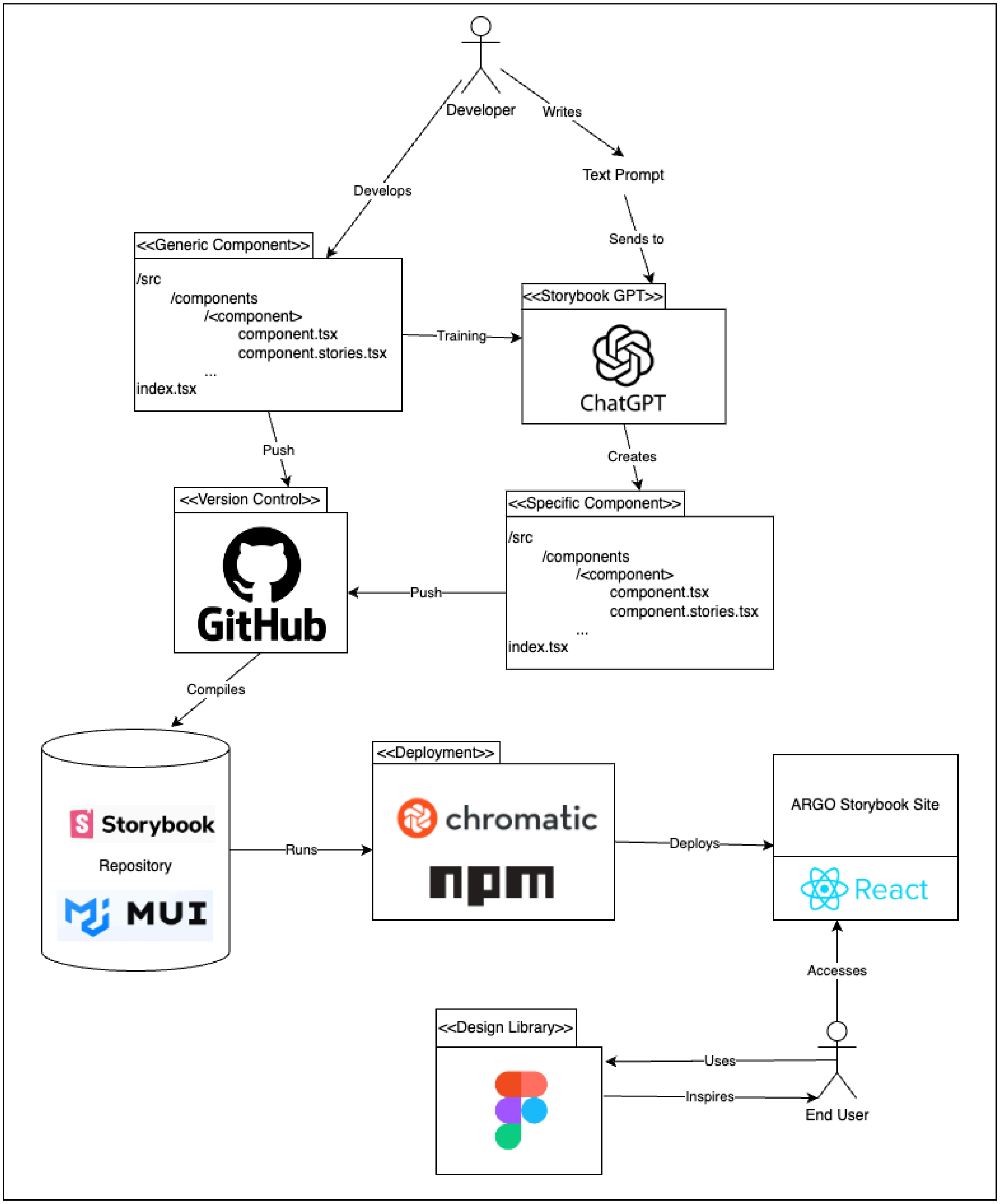

Figure 1. Architectural Model

Architectural Style(s) Used

Factory Method Pattern

The creation of components is best described using the Factory Method Architectural Pattern. Material UI is leveraged to create and define basic components in TypeScript. The Storybook GPT system serves as a factory to create different variations of the basic components.

Decorator Pattern

This pattern is used to add functionality to existing objects. Our system utilizes the existing Material UI library as the basis for components, but builds upon it to meet our needs. Storybook allows developers to modify a component's properties through its graphical user interface. The Storybook GPT stories that are generated can be considered to be adding functionality to the existing base components.

Facade Pattern

The GitHub Actions workflow and its configuration server as a facade for more complex processes such as releasing to npm and deploying to Chromatic.

Rationale For Your Architectural Style And Model

Generic and Specific Component Subsystems

Generic and Specific components consist of a component.tsx Typescript file, and component.stories.tsx Stories Typescript file, and an optional component.css file. The Typescript file is the React component file, defining the structure, functionality, and behavior of the component. The Typescript file is also where you would define your components props and state. The Stories Typescript file is specific to Storybook. A "Story" or "Stories" represent the visual state or configuration of the component, allowing multiple behaviors and conditions of the component. The optional CSS file defines styling for the component, such as layout, typography, colors, or other design elements. Styles can also be defined in the Typescript file, but maintaining a separate Cascading Style sheet can make the component easier to maintain. These files use the Storybooks ecosystem, and additional storybook documentation can be found online.
More specifically, our system defines Generic and Specific Component Subsystems. Generic components use the MUI library to define a typical react component, such as a button, slider, and more. The MUI library contains dozens of basic react components, alongside their coding program. A developer would use tools such as MUI to develop a component. In a Specific Component Subsystem architectural flow, the developer would describe a variation or more specific component to the trained Storybook GPT. The GPT would create either a Stories Typescript as a variation of an existing component, or create a Stories Typescript and a Typescript file for a brand new component.

Storybook GPT Subsystem

The Storybook GPT Subsystem is a deliverable of the ARGO Storybook POC. During the POC, we (as one time developers) use the Generic Components as input to train the expected output of the Storybook GPT. Once completed, the purpose of this trained GPT is to accept developer text prompts and create Specific Components based on learned expectations. The Storybook GPT Subsystem will use GPT-4 for the LLM.

Version Control and Component Library

Once the developer creates, runs, and successfully tests the Generic and Specific Components, they are pushed to the Component Library using version control. Our team uses a GitHub repository for hosting our Storybook repository. The Storybook repository maintains a combination of developer created components to be quickly used and reused by the End User and closely tied to the design library. GitHub Actions workflow automates releases of the Repository system to the latest update of npm and Chromatic. An additional automated workflow is used to build the latest Storybook documentation and deploy it to the hosted Chromatic instance.

Deployment and ARGO Storybook Site

Once the automatic GitHub Action workflow is used, it deploys the ARGO Storybook site. This React site is accessed by the End User. In this instance, the End User is another layer of developer but not exclusively an individual that differs from the Developer user. While this stakeholder may be the same or a unique individual, the intention between accessing the React Site is to utilize, reuse, variant, and implement using a component stored within the site. This site is intended to be used to save the end user much time in creating components. Instead of programming, testing, inserting, designing, and deploying every component for a page, the developer accesses the site to grab a component that is already programmed, designed to match specifications in styling, and tested. In a very specific use case, the end user accesses the site to find a component to use for their site. They may notice a component needs to be varied in order to work within their system better. In this case, the End User now becomes the developer that writes a text prompt to the GPT. The architecture is followed all the way down to deploying the site. Now, the new variant will be not only developed quicker using the Storybook GPT, but also conveniently documented for reuse in the repository.

Design Library Subsystem

The Design Library Subsystem is a combination of Documentation, Design, and Code. This system shall be used by the ARGO team as a front end tool for using new components, complying with ARGO design standards, and overall streamlining the developer experience.

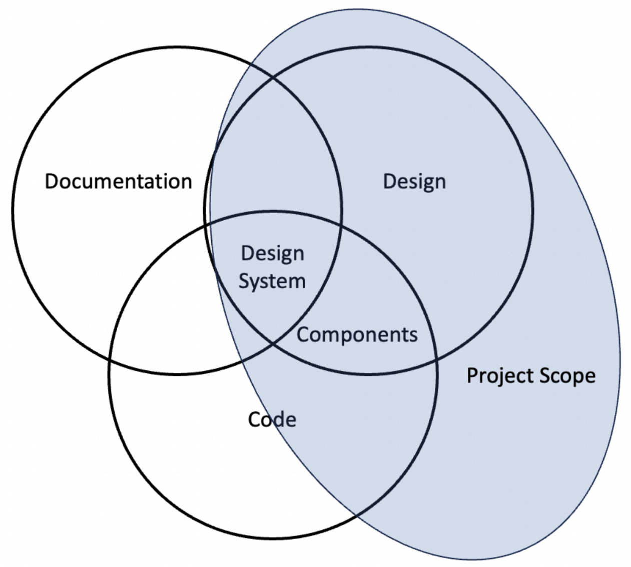

Figure 2. Design Library Subsystem

Traceability From Requirements To Architecture

- provide a mapping between requirements and architecture
- clearly describe how each requirement in the *Requirements Documentation* is captured in the architecture

Functional Requirements: based of use cases from requirements documentation

Functional Requirement	Subsystem implemented for the requirement	Subsystem relation to the requirement
The system allows designers to review, customize, and finalize components, ensuring that all designed components are easy to use and align with user expectations and behaviors.	Figma ,Storybook, Component Library, npm	Figma allows designers to gain inspiration. Whereas, Storybook allows for components to individually be showcased with a modular and reusable design.
The system enables frontend developers to access and review documentation created, ensuring adherence to necessary standards and guidelines, and approval upon review.	Storybook	Enables frontend developers to access and review documentation created within Storybook, ensuring adherence to necessary standards and guidelines, and approval upon review.
The system must support Component Library Developers in reviewing design specifications and writing code for components.	Component Library, MUI, ChatGPT	Subsystems assist in facilitating the reviewing of design specifications and writing code for components.
The system should allow input parameters for component stories and should generate stories accordingly, also allowing developers to review and approve the generated stories.	Storybook GPT	Storybook GPT assists in facilitating the input parameters for component stories, generating stories accordingly, and enabling developers to review and approve.
The system must enable Component Library Developers to commit code changes to the Component Library repository.	GitHub, npm	npm provides the functionality for version control and code management, allowing developers to commit code changes to the Component Library repository.
The system should allow Component Library Developers to create pull requests for component changes.	GitHub	These subsystems facilitate the creation and management of pull requests, allowing for collaboration and review of component changes before merging.
The system must successfully integrate components into the Component Library after implementation.	Component Libraries, npm, Chromatic	Component libraries handle the integration of components into the library after they have been implemented and approved, ensuring that they are available for use within the system.

The system should support iterative refinement of components until they meet specifications.	Figma, Chromatic, React	Figma, Chromatic, and React collectively support iterative refinement. Figma provides design tools for refining components, Chromatic facilitates visual testing and feedback loops, and React allows for implementation adjustments based on specifications.

Table 2. Traceability from Requirements to Architecture of Functional Requirements

Non-Functional Requirements

Non-functional Requirement	Subsystem implemented for the non-functional requirement	Subsystem relation to the Non-functional requirement
Usability	Figma	Figma ensures usability with an intuitive interface for designing and customizing components.
Maintainability	GitHub	GitHub allows the system to be easily maintained and updated over time.
Extensibility	GitHub, Storybook	Storybook provides extensibility by accommodating future enhancements or modifications without significant changes.
Scalability	React	Ensures the system can handle increased workload and user demand effectively.

Table 3. Traceability from Requirements to Architecture of Non-Functional Requirements

Evidence The Document Has Been Placed Under Configuration Management

We use Google Docs as a tool to create our document. Following is a table to describe the version changes the document has experienced.

Version In	Version Out	Changes	Reviewed By	Notion Task Numbers
n/a	0.0	Document Creation based on Template	Lillie McMaster and Alina Khan	UG3-3
0.0	1.0	Architectural Model, Styles used, Rationale, Traceability, Technology, Software, and Hardware Used. Update to template for ABET accreditation	All Group Members	UG3-73, UG3-74, UG3-75, UG3-80
1.0	2.0	Feedback provided by ARGO	All Group members	UG3-81

Table 4. Configuration Management

Engineering Standards And Multiple Constraints

- IEEE Std 1471-2000: Software Architecture [pdf]
- ISO/IEC/IEEE Std 42030:2019: Software, Systems and Enterprise
    - Architecture Evaluation Framework [pdf]

Additional References

- Lattanze, A.J., 2008. *Architecting Software Intensive Systems*: A Practitioner's Guide. CRC Press
- Bass, L., Clements, P. and Kazman, R., 2003. *Software Architecture in Practice*. Addison-Wesley
- Storybook Image from Medium. *Configuring Storybook: 6 Tips You Can't Miss*. Accessed March 11th 2024. https://medium.com/strands-tech-corner/storybook-configuration-in-react-project-ec59869f3e7d
- React Image from *GitHub React-Ui Topics*. Accessed March 11th 2024. https://github.com/topics/react-ui
- GitHub Image from Project Pythia. *What is GitHub?* Accessed March 11th 2024. https://foundations.projectpythia.org/foundations/github/what-is-github.html
- MUI Image from MUI. Accessed March 11th 2024. https://mui.com/
- Figma Image from Figma. Accessed March 11th 2024. https://www.facebook.com/figmadesign/
- ChatGPT from JacobsMedia. *ChatGPT Logo Square*. Accessed March 11th 2024. https://jacobsmedia.com/a-radio-conversation-with-chatgpt-part-1-sales/chatgpt-logo-square/