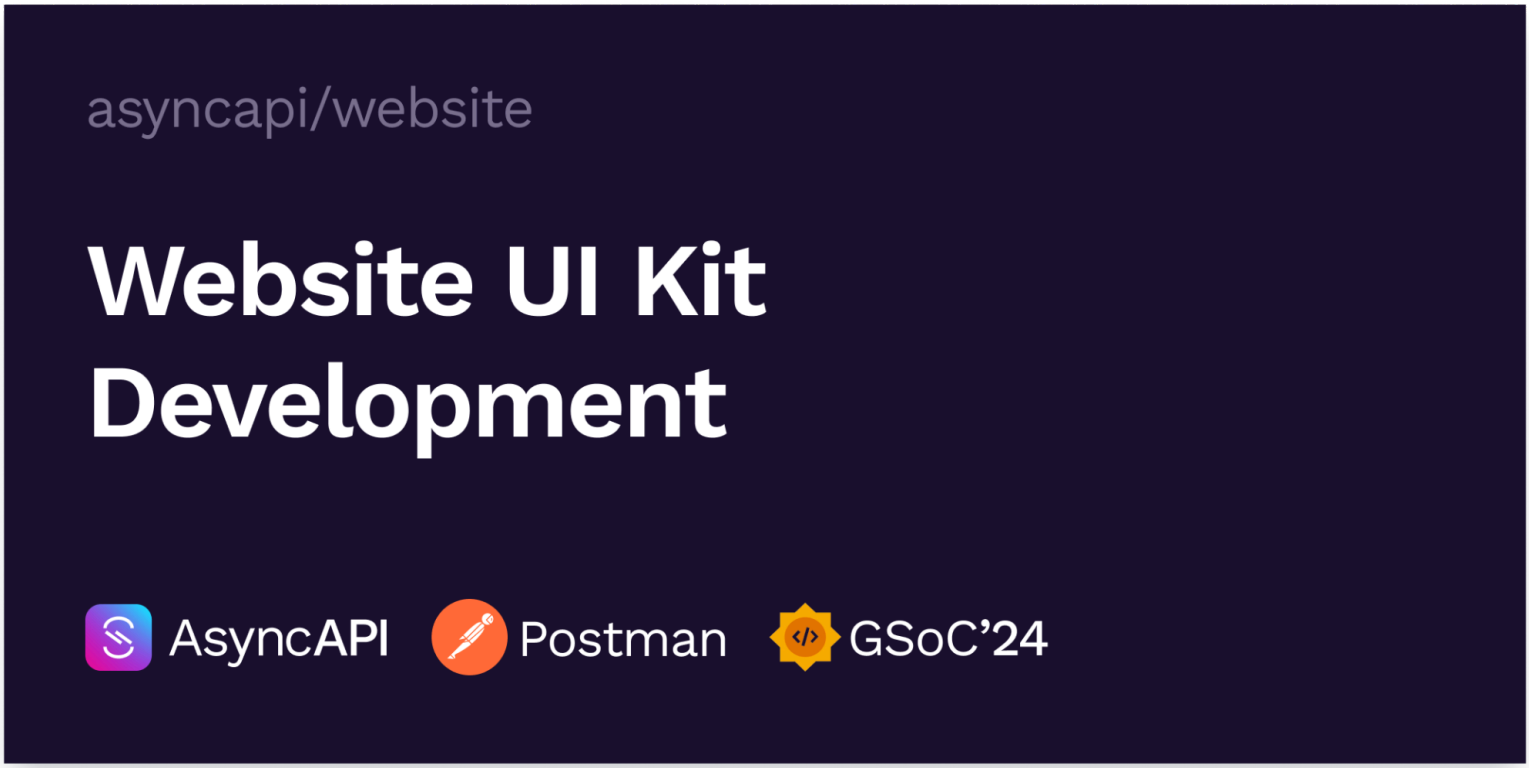
# GSoC ‘24 Proposal




## AsyncAPI Website UI Kit Development

# Contact Information


Name : Ashmit Gupta

Email : ashmitgupta.official@gmail.com

GitHub : devilkiller-ag

Website : https://ashmit-jaisarita-gupta.vercel.app/

LinkedIn : Ashmit JaiSarita Gupta

Phone : +91 8299538244

Twitter : Ashmit JaiSarita Gupta

Address : H.No. 30, Gurutola, Azamgarh, Uttar Pradesh, 276001

Time Zone : Kolkata, India (GMT +5:30)

University : National Institute of Technology Hamirpur

Degree : Bachelor of Technology (pre-final year)


How many hours will I work per week : 40 hrs/week minimum

Other commitments : College academics

# Project Information

**Title:** AsyncAPI Website UI Kit Development

**Length:** 175 Hours

**Time:** 12 Weeks

**Technologies:** JavaScript/TypeScript, React, Next.js, Storybook, TailwindCSS

**Mentors:** Azeez Elegbede (Ace), Akshat Nema

## Abstract: Problem and its solution

AsyncAPI is an open-source initiative that seeks to improve the current state of Event-Driven Architecture (EDA). It has a set of tools for documentation, code and model generation, event management, etc. which helps to easily build and maintain EDA. The AsyncAPI Website is the primary source of information for users and developers.

Currently, the website lacks visual consistency, repeated elements lack consistency in design, and duplicate styling is used for similar visual styles. This makes the codebase and design non-modular.  Existing UI patterns are undocumented which results in miscommunication and re-inventing the wheel instead of building new features.

The goal of this project is to develop a comprehensive UI Kit that can enhance the existing design, and streamline the development process to simplify the creation and management of cohesive elements in the website. This website UI Kit will help in preventing the process of rebuilding similar components.

## Project Current State

The issue #2090 in the website repository (adopted from issue #4 in the design-system repository) is currently monitoring the progress of this project. This project was divided into two sub-parts:

1. Design
    - Audit all design patterns such as common reusable UI components and design tokens(such as brand colors, spacing, and typography) on the current website.
    - Create a Design System in Figma that includes design tokens, atomic and molecular components, and their various states.

2. Development
    - Develop the stories for various states of these components in the storybook.
    - Do the visual tests, interaction tests, and accessibility tests of all these components in the storybook.
    - Create documentation of all these components giving their appropriate usage.

The design part of this project has been completed by Aishat Muibudeen (Maya) under the mentorship of Ace. This is the Figma file for the design system created by Maya. The next step and the scope of this GSoC project will be to do the development part of this project and deliver the complete UI Kit.

# Approach and Implementation


Modern user interfaces are assembled from hundreds of modular UI components rearranged to deliver different user experiences. AsyncAPI Website UI Kit will contain reusable UI components that will help developers build complex, durable, and accessible user interfaces across the website. It will be a *source of truth* for the website's common components. We will use Storybook for developing our website UI kit. Storybook provides a live, visual platform to develop and test UI components, enhancing efficiency and organization.

## Tech we will be using

- **Storybook** for UI component development and documentation
- **React** for developing component-centric UI

- **Typescript**
- **Tailwind** CSS for styling
- **Prettier** for automatic code formatting
- **ESLint** for JavaScript linting
- **Chromatic** to catch visual bugs in components
- **GitHub Actions** for continuous integration

Besides Storybook and Chromatic, the technologies listed above are already being used in the AsyncAPI ecosystem. The next section gives an introduction to Storybook and Chromatic and discusses why we need them.

**Storybook:**
Storybook is a front-end workshop for building UI components and pages in isolation. It helps us to develop hard-to-reach states and edge cases without needing to run the whole app. It provides an interactive playground to develop, test, and browse your components, making it an invaluable tool for component-driven development in React. Using it, developers can build UI components detached from the business logic and context of their app. This not only enhances the reusability of components but also improves testing and consistency across the application.

The need for a Storybook can easily be understood from the following scenario: Say we have to build a form that uses many smaller components like inputs, buttons, etc. Since the Async API Codebase is very huge, it might be difficult to check whether these smaller components are already available in the website codebase or not. Through Storybook developers can search through our project and be able to quickly kind of visually check and how it looks, its properties, play with them to get ideal props values.

**Chromatic:**
Chromatic is a visual testing & review tool that scans every possible UI state across browsers to catch visual and functional bugs. It catches visual and functional bugs in stories automatically. It runs UI tests across browsers, viewports, and themes to speed up front-end teams. We can assign reviewers and resolve discussions to streamline team sign-off. It streamlines the process of shipping UI components with higher quality. Chromatic is the maintainer of Storybook.

## What the UI kit will contain and what it will not

Website UI Kit should only contain atomic, molecular, and organism-level components that are used in many places. These components deal with how UI appears, respond exclusively to props, should not contain website part-specific business logic, and is agnostic to how data loads. A useful metric for selecting a component to be the part of

UI kit can be: *if the component is used at 3 or more places, then put it in the UI kit.* This selection process has already been done in the design part. The stories and documentation for the following components/design tokens will be included in the UI kit:

- Colors, Typography, Spacing, Shadows (Design Token/ styling constants)
- Buttons
- Dropdowns
- Icons
- Tags
- Pagination
- Toggle
- Input Fields
- Accordion
- Avatars
- Cards
- Notifications
- Header
- Footer
- Subscription
- Docs, Tools, and Community Dropdown
- Search
- FAQs

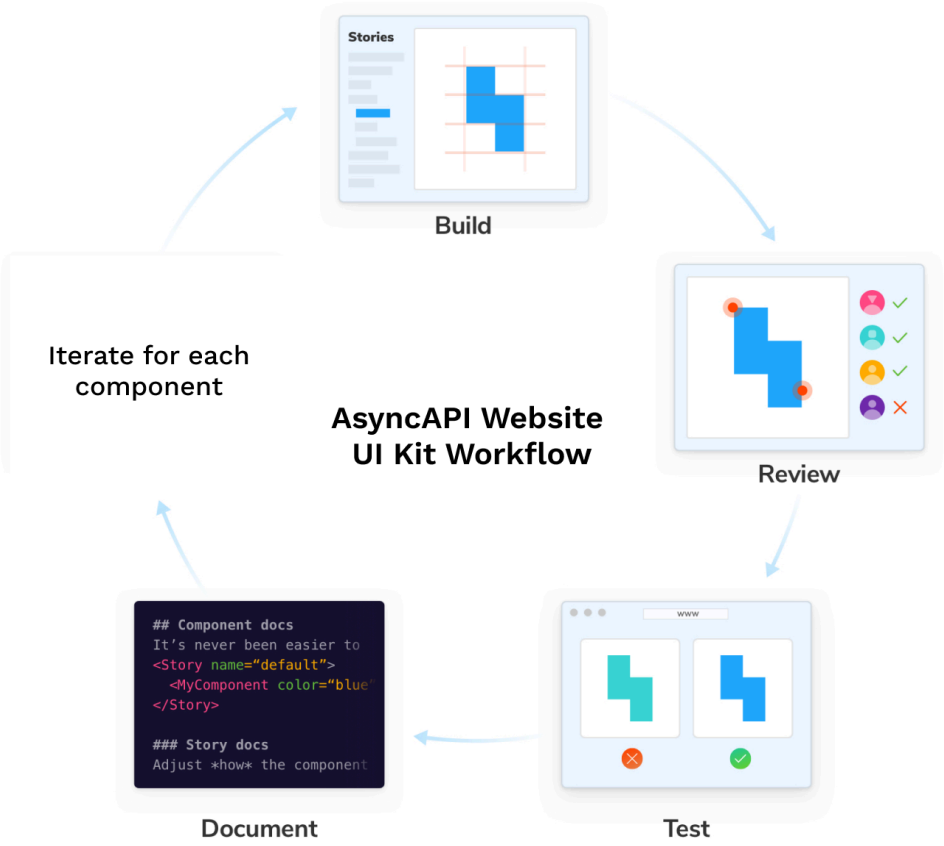
I haven't covered Templates (New Homepage & Playground Mock Up) given in the design system by Maya because they don't satisfy the above criteria. We will need to discuss these with TSC and Mentors to decide whether to include them in the UI Kit.

## Workflow

Different organizations use different workflows for building UI kits for their website. However, the engineers at Storybook after researching the best practices used by developers of successful UI Kits and Design Systems suggested the following workflow:

**Build stories for components**
Most of the components present in the Design System made by Maya are already developed on the website. Now we have to develop stories for each of these components covering all states that it can have. However, I will be developing the missing components and updating the existing components if required to make them independent of any specific business logic. Currently, I am part of the team working on the migration of the website to TypeScript and have migrated more than 35

components. Through this work, I have developed a deep understanding of the codebase which will help me in the future while working on them.

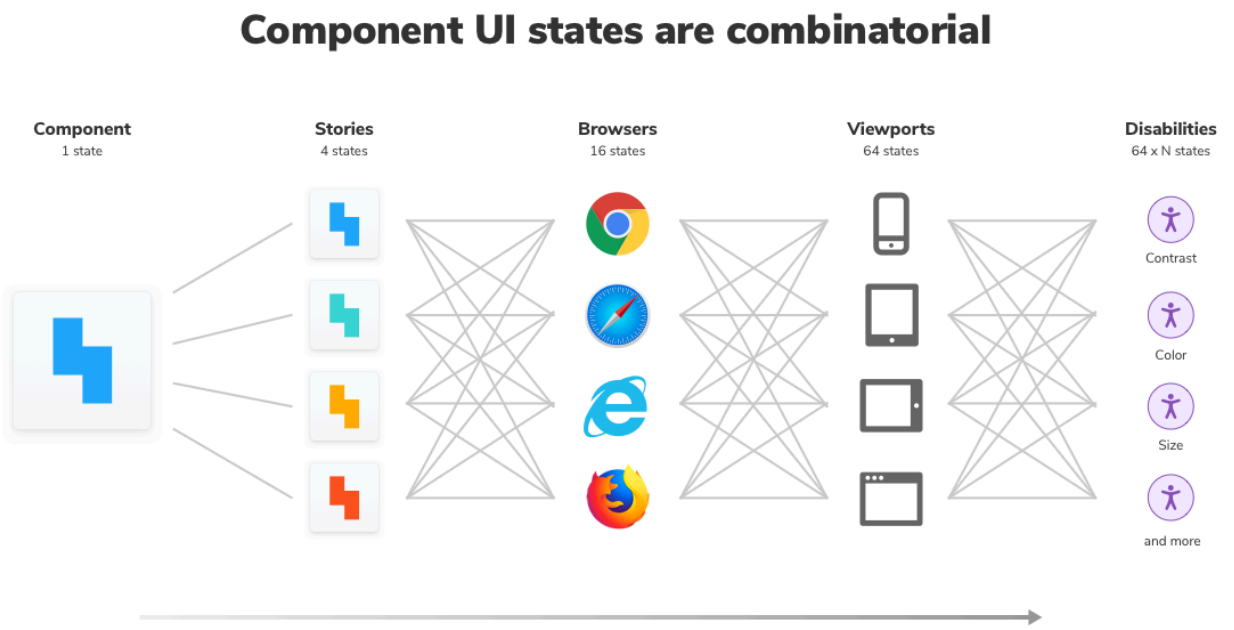



**Get a review of the component**

Once all the stories for a component are developed, get a review from mentors and the designer. However, we can move this step to the end of the loop, once tests and documentation for the component are completed. We can add the Figma frame for each of the components from the Design System in Figma to Storybook using the Figma plugin. This will help developers to cross-reference from the design of the components.

**Test the component to prevent UI bugs**

Each UI component includes stories (permutations) that describe the intended look and feel given a set of inputs (props). Stories are then rendered by a browser or device for the end-user. Keeping manual track of all these stories is an unsustainable and hectic task. Storybook enables us to automate tests which helps to detect and rectify bugs. Many types of tests can be performed on the components, but the research done by Storybook engineers suggests that these UI tests are most effective for UI Kit:




- **Visual tests**
  Visual tests capture an image of every UI component in a consistent browser environment. New screenshots are automatically compared to previously accepted baseline screenshots. When there are visual differences, you get notified. This will save the time and effort required for manual reviews. We can use Chromatic for this and automate this using GitHub actions.

- **Interaction tests**
  If our components handle state management or fetch data, we should do the interaction testing of our component using mocked data, Storybook *play* function, Storybook *test* package based on the Vitest, and Storybook *test-runner* for automation. However, in our case, we won't necessarily need this test.

- **Accessibility tests**
  Disabilities affect 15% of the population, according to the World Health Organization. Therefore we need to check the accessibility of our components. We can do this easily using the accessibility addon by Storybook which verifies the web accessibility standards (WCAG) in real-time.

We can automate all these tests using GitHub workflows. Though there are other tests like code coverage tests, snapshot tests, and end-to-end tests, they are not suitable for

website UI kits and design systems since they contain atomic components with simple functionality. In our case, Visual tests and accessibility tests are best to have.

**Document the component**
To achieve the full work-saving benefits of a website UI kit, components should be easy to understand and widely reused. This can be made possible using documentation for components in the UI kit. However, maintaining and keeping the documentation up-to-date is a tedious task. Storybook enables us to auto-document the components which can be further customized. They provide boilerplate code and offer customizability so that developers don't have to rewrite common patterns. We can generate documentation from existing stories using the *docs* add-on which reduces the maintenance time.

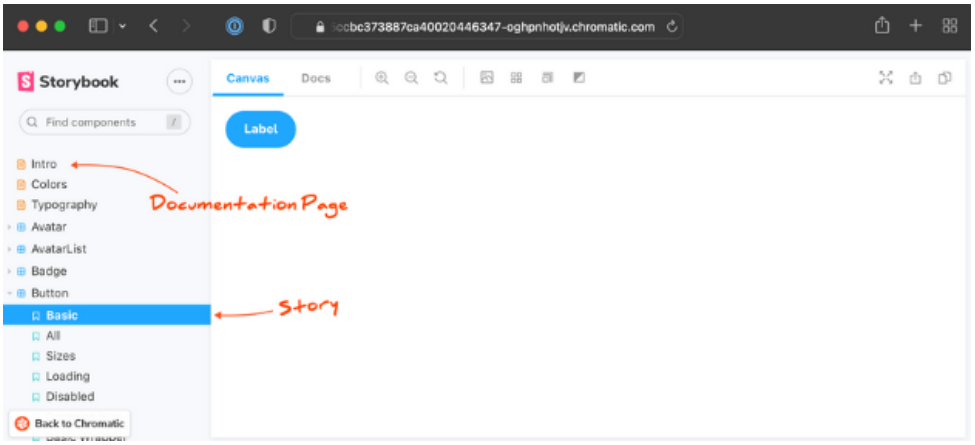
## Code formatting and linting

Enforcing code consistency increases the readability and maintainability of the code. Using tools that fix syntax and standardize formatting serves to improve contribution quality. To ensure a consistent code style, we should use tools like Prettier and ESLint. These tools are widely used, support multiple languages,  and seamlessly integrate with most editors.

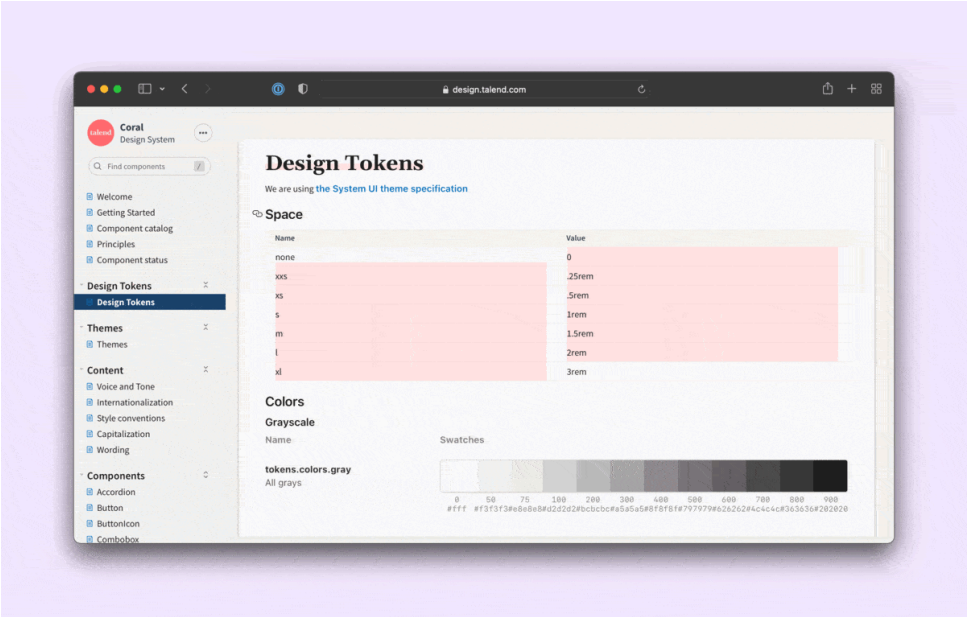
## Organizing the Storybook

Organizing our storybook in an appropriate manner is important for sending a clear and effective message to developers who will be using it and to those who will be maintaining it in the future. I propose to use the following widely used structure for organizing our storybook.

**Introduce the UI Kit Storybook:**
We will have documentation pages for the Introduction, Getting Started, and changelog at the top. This will be followed by Design Tokens Documentation.



For example, Talend Coral uses a mix of DocBlocks and custom components to document their design tokens.




**Grouping and sorting the components in Storybook:**
Stories for a component are automatically grouped by Storybook. Storybook also allows you to group multiple components into categories and adjust their order in the sidebar. We will be following the *Atomic Design* hierarchical system. It classifies components into five levels: atoms, molecules, organisms, templates, and pages. However, our UI kit will only have atoms, molecules, and organisms. Each component will have its documentation page inside its category.

**File Structure in the codebase:**
As suggested by the default installation of the storybook, we can keep all stories in one folder separate folder since our website has a large number of components which can make it hard to navigate between stories and components. So instead of having a single big `*stories/*` directory, I will keep my stories for a component in the same folder in which the component is located.

## References:

I have developed my workflow from the following resources where engineers from Storybook have given detailed suggestions on working with Storybook after researching about more than 60 production Storybooks:

- Intro to Storybook
- Design Systems for Developers
- UI Testing Handbook
- Visual Testing Handbook
- Structuring your Storybook

Besides these, I have studied the Design System created by Maya in Figma and the website codebase in detail. The images I have used belong to various Storybook and AsyncAPI tutorials.

# Project Timeline

During the pre-proposal period, I contributed to the migration of the website to TypeScript to familiarize myself with the existing codebase and the upcoming codebase (migrated to TypeScript) of the website. I familiarized myself with the Design System built by Maya. I explored and analyzed each atom, molecule, organism, and template present in this design system. I tried creating stories for a few of the atoms (buttons and tags), present in the design system using their react component present in the migrated codebase.

During the pre-selection period in April 2024, I will continue helping in the migration of the website to TypeScript so that we can finish it before the beginning of the GSoC Community Bonding Period. This is necessary to be completed before mentees begin working on any project related to the website.

## Community Bonding Period (May 1, 2024 - May 26, 2024):

During the community bonding period, I will need to discuss and refine my timeline and strategy. Since my mentors and I are from different time zones, we will need to fix a common meetup time.  Although I am aware of and used to the website codebase, I will re-analyze the codebase to understand each component again. To make sure that all components are engineered dynamically and documented properly, communication with the design team (Maya and Ace) is important. I will discuss, refine, and iterate through my strategy and plan with them to make sure that I won't get blocked by anything in the coding period. I will also be traveling back to my home from my college on the 2nd or 3rd week of May. So, I may not be able to actively participate for 3 days during my travel.

## Coding Period (May 27, 2024 - August 19, 2024):

- **Week 1 (May 27, 2024 - June 1, 2024)**
  **Component Covered:** *Design Tokens (Colors*, *Typography*, and *Shadows)*
  Document Introduction page and design tokens

  **Deliverables:**
  1. Create a new branch *storybook* in which I will be doing my work until it's complete.
  2. Create an Intro Page in the Storybook from the Readme in Design System in Figma.
  3. Document the color pallets for Primary, Secondary, Grey (Neutral), Base, Parser, Studio, Glee, Modelina, Success Swatches, Warning Swatches, Error Swatches, and Gradients. We will use the ColorPalette functionality of Storybook to achieve this.
  4. Document the Typography. This page will include the system, style, and example use case for Header 1, Header 2, Header 3, Sub-heading 1, Sub-heading 2, Paragraph, Caption, and Subtitle. We will use the Typeset functionality of Storybook to achieve this.
  5. Create a documentation page for shadows covering: *shadow-xsmall*, *shadow-small*, *shadow-medium*, *shadow-large*, *shadow-xlarge*, and *shadow-xxlarge* variants suggested in the Shadow Effects Page of the Design System.
  6. PR #1: Commits for all above deliverables.

- **Week 2 (June 3, 2024 - June 8, 2024)**
  **Component Covered:** *Icons*
  There are currently 59 icons that are used in the AsyncAPI website. However, Maya has mentioned more than 1000 icons on the Icons Page of the design system she created on Figma. After a discussion with Akshat and Maya, we have decided to include only 59 icons in the UI Kit that are currently being used in our design system. Other icons can be added to the UI Kit when they are being created and used anywhere on the website.

  **Deliverables:**
  1. Code the missing icons (if we need them) using the `copy as SVG` functionality in Figma.
  2. Document all the Icons using the IconGallery functionality of Storybook.
  3. PR #2: Commits for all above deliverables.

- **Week 3 (June 10, 2024 - June 15, 2024)**
  **Component Covered:** *Buttons*

  **Deliverables:** Create and document stories for the following variants of Buttons suggested on the Buttons Page of the design system.
  1. Primary CTA (4 Content Variants * 4 Size Variants * 2 Color Variants)
  2. Default Buttons (4 Content Variants * 4 Size Variants)
  3. State Buttons (3 State Variants * 4 Size Variants)
  4. Secondary (4 Content Variants * 4 Size Variants * 2 Color Variants)
  5. PR #3: Commits for all above deliverables.

- **Week 4 (June 17, 2024 - June 22, 2024)**
  **Component Covered:** *Dropdowns*
  Currently, there is no Dropdown Component on our website similar to what is present in the Dropdown Page of the design system. Instead, there are different kinds of dropdown components (Filter Dropdown, and Category Dropdown) used in the filter section of the AsyncAPI Tools Dashboard. So I will have to first develop a new dropdown component if we want to develop stories for the dropdown suggested in the design system.

  **Deliverables:**
  1. Create a Dropdown component.
  2. Create and document stories for two variants: Filter by Type and search by name
  3. PR #4: Commits for all above deliverables.

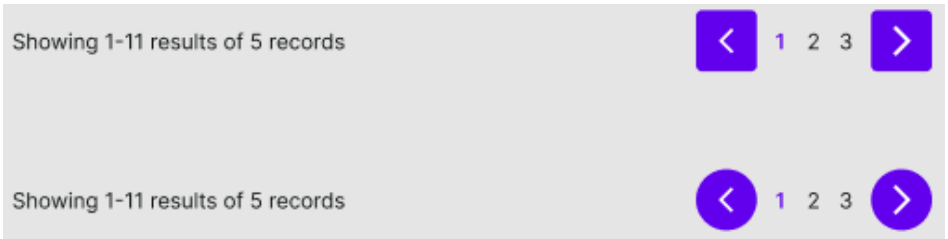
- **Week 5 (June 24, 2024 - June 29, 2024)**
  **Components Covered:** *Tags, Checkbox, Radio,* and *Toggle*
  Currently, the tags component in the website does not have the functionality to add leading and trailing icons as suggested in the Tags Page of the Design System. Also, there are no Checkbox, Radio, or Toggle components in the codebase.

  **Deliverables:**
  1. Add leading and trailing icons functionality to the tags component.
  2. Create and document stories for the four variants of tags given in the Design System.
  3. Create a checkbox component and replace it at all places where the checkbox input type was used.

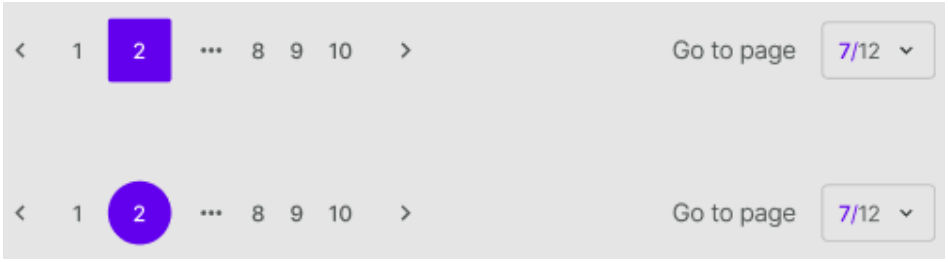
4. Create a radio component and replace it at all places where the radio input type was used.
5. Create a toggle component and replace it at all places where the toggle was used.
6. Create and document stories for two size variants of checkbox and radio, and one variant of toggle.
7. PR #5: Commits for all above deliverables.

- **Week 6 (July 1, 2024 - July 6, 2024)**
  **Component Covered:** *Pagination*
  Currently, there is one Pagination Component in the website codebase, which was developed for use in the community dashboard but it isn't used anywhere. The Pagination Page in the Design System, suggests six kinds of pagination components:
  - Type 1:

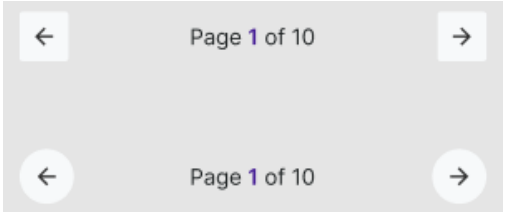

  - Type 2:




  - Type 3:

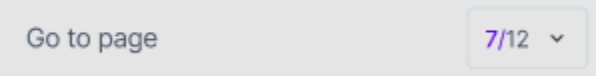



  - Type 4:



- Type 5:


- Type 6:




**Deliverables:**

1. Develop all these types of pagination components.
2. PR #6: Commits for all above deliverables.

- **Week 7 (July 8, 2024 - July 13, 2024)**
  **Component Covered:** *Pagination*

  **Deliverables:**

  1. Create stories and documents for all six kinds of pagination components.
  2. PR #7: Commits for the above deliverable.
  3. Mid-Term Evaluation Submission and cover up any remaining tasks.

- **Week 8 (July 15, 2024 - July 20, 2024)**
  **Component Covered:** *Avatar* and *Input Field* Components
  Currently, there are two similar components for Avatar in the codebase: Profile and AuthorAvatar. We can combine these two and also add appropriate props to support the Avatar format given in the Avatar Page of the Design System. There is currently no Input Field component in the codebase.
  **Deliverables:**

  1. Update Avatar Component.
  2. Create stories and documentation for all the variants of avatars given in the design system.
  3. Create an input component and replace it at all places where the input text type was used.
  4. Create stories and documentation for all the variants of the input field given in the Input Field Page design system.
  5. PR #8: Commits for all the above deliverables.

- **Week 9 (July 22, 2024 - July 27, 2024)**
  **Component Covered:** *Accordion* and *Card* Component
  Currently, there is no Accordion Component in the codebase similar to what is present in the Accordion Page of the design system. Card components are used

in many places like DocsCard on the Docs Page, BlogPostItem on the Blogs Page, and Cards on the Community Page. We will have to create one common card component that can be used to create all the variants given in the Cards Page of the Design System using props.

**Deliverables:**

1. Create an accordion component and replace it at all places where the accordion was used.
2. Create stories and documentation for all the variants of accordions given in the design system.
3. Create a card component and replace it at all places where the accordion was used.
4. Create stories and documentation for all the variants of cards given in the design system.
5. PR #9: Commits for all the above deliverables.

- **Week 10 (July 29, 2024 - August 3, 2024)**
  **Component Covered:** Algolio Search, Notifications
  This component is already developed and being used. With some minor styling and icon changes, it will become the same as given the Search Page of the Design System. Currently, there is no notification component in the website codebase. So, we will have to create this component from scratch as given in the Notification Page of the Design System.

  **Deliverables:**

  1. Develop stories and documentation for Algolia Search.
  2. Create the Notification Component.
  3. Create stories and document all the variants of notification given in the design system.
  4. PR #10: Commits for all the above deliverables.

  **Note:** This week I will also be traveling back to my college (it approximately takes three days of travel).

- **Week 11 (August 5, 2024 - August 10, 2024)**
  **Component Covered:** Header & Footer, Subscription, Docs Dropdown, Tools Dropdown, Community Dropdown
  **Deliverables:**

  1. Create stories and documentation for the Header component.
  2. Create stories and documentation for the Footer component.
  3. Create stories and documentation for the Subscription component.
  4. Create stories and documentation for the Docs Dropdown component.

5. Create stories and documentation for the Tools Dropdown component.
6. Create stories and documentation for the Community Dropdown component.
7. PR #11: Commits for all the above deliverables.

**Note:** I have also kept this week as a buffer period to counter any unforeseen delay.

**Week 12 (August 12, 2024 - August 19, 2024)**
**Component Covered:** FAQs
**Deliverables:**

1. Create stories and documentation for the FAQs component.
2. Add functionality to accessibility tests and visual tests.
3. Integrate Chromatic with our Storybook.
4. Add continuous integration workflows for tests using GitHub actions.
5. Deploy the Storybook.
6. PR #12: Commits for all the above deliverables.

**Note:** I have also kept this week as a buffer period to counter any unforeseen delay.

- **Week 13 (August 19 - August 26, 2024)**
  **Deliverables:**
  1. Merge the *storybook* branch with the *master* branch.
  2. Submit the final work product and final mentor evaluation.

I have kept a day gap between each week for holiday or to cover up any remaining tasks of the previous week.

## Tracking the progress and sharing with the community:

I will create a separate branch in the website repository for working on this project. I will use the existing issue for this GSoC project or create a new one to track my progress. I would prefer to create PR for one component at a time so that I can get quick reviews from my mentor and not repeat any mistakes I made while working on the current component.
From my experience of working on a previous mentorship program on Quantum Neural Networks, I love to share my weekly journey through a Hashnode blog. It enables me to get feedback from others interested in similar work. Here is an example blog I wrote for my previous project.

## Plans regarding this project after GSoC 2024:

I look forward to the opportunity to build a Design System for the whole AsyncAPI Initiative Ecosystem around this project in the future. Currently, the websites for different AsyncAPI tools like Modelina also use similar components as those used in the AsyncAPI Website.

Having a design system and distributing it as a package will help in maintaining a consistent brand style across all the websites under AsyncAPI and will prevent re-inventing the wheel of developing similar components. Before starting this, we will need discussions and suggestions from other contributors, maintainers, and TSC members in the AsyncAPI.

## How can mentors get the best out of me:

During the community bonding period, mentors can get the best out of me by discussing and refining my timeline and strategy. During the coding phase, mentors can get the best out of me by having a weekly or bi-weekly meeting to analyze my work and suggest the best possible route and coding practice that I should follow in the upcoming week. Besides these, I will need regular code reviews and feedback after stories for every one or two components (atom/molecule/organism) to ensure I am not doing something wrong and stay on track.

It would be awesome if I could get regular feedback from Maya, who created the Design System in Figma. (I know mentors are busy people too, so it will be okay if we miss some of these check-ins, I will be proactive to ensure everything is going on according to the strategy and timeline agreed with my mentors during the community bonding period.)

## Other commitments during GSoC 2024:

From May to August 2024, I will have my summer vacation and after that, I will be having my semesters (9 AM - 4 PM every day). I will be easily able to take out 4-5 hours a day during May-August and 3-4 hrs after that during my semester. Besides this, I plan to study Operating Systems and Statistical Mechanics in my summer vacations.

# Understanding & Involvement in the AsyncAPI Community

I have been contributing to the AsyncAPI Initiative for the last six months. My journey started with AsyncAPI when I came across the AsyncAPI Onboarding Playlist by Lukasz Gornicki on YouTube while exploring Event-Driven Architecture. Since then I started contributing to various AsyncAPI Tools while exploring them. My interest in the community increased when I first joined the community meeting in December 2023. Since then, I have been joining the community meetings regularly.

I have worked on projects like the re-design and development of the Modelina Playground, the development of the Modelina Website according to the new Figma Design, migrating the Modelina Website and Playground from class-based components to functional components, migration of AsyncAPI Website to TypeScript, etc. These contributions have developed my deep interest and understanding of the website and interface development of AsyncAPI tools. I am the code owner of the Modelina Website and a TSC Member inside AsyncAPI. I have reviewed 15+ pull requests in modelina. Till now I haven't been able to use AsyncAPI in any project, I am planning to search and work on a project which uses AsyncAPI. Currently, I am exploring the AsyncAPI documentation to deepen my knowledge of AsyncAPI.

Besides working on this project, I will be continuing to work on the Modelina CLI with Jonas Lagoni. I am planning to contribute to writing tests for Async API projects (especially the website). Earlier in January 2023, I planned to work on the Real-Time Collaboration feature in the AsyncAPI Studio, but it got postponed. So I want to work on this in the future if it reopens. After the completion of this GSoC project, I am looking forward to becoming the maintainer of the website and a mentor in next year's mentorship program.

# Contribution History in AsyncAPI

**April 2024**

- Migrate the files of the community page in the AsyncAPI Website to TypeScript.

**March 2024**

- Migrate NewsletterSubscribe, CaseTOC, GeneratorInstallation, Hero, InlineHelp, Testimonial, NewsletterSubscribe, MDX, and Hero components in the AsyncAPI Website to typescript (PR).
- Migrate Figure, Loader, Remember, Warning, and Profile components in the AsyncAPI Website to typescript (PR).
- Migrate components of tools folders in the AsyncAPI Website to typescript (PR).
- Updated conference details in the AsyncAPI Website (PR).
- Changed the button props type to enum in the AsyncAPI Website (PR).
- Migrate components of community folders in the AsyncAPI Website to typescript (PR).
- Migrate components of sponsor and support folders in the AsyncAPI Website to typescript (PR).

**February 2024**

- Migrate components of docs, data, and editor folders in the AsyncAPI Website to typescript (PR).
- Extracted the functionality of Modelina from AsyncAPI CLI and created the Modelina CLI from it (Issue, PR 1, PR 2).

**January 2024**

- Redeveloped Modelina website according to the new design (Figma, Issue, PR 1, PR 2).
- Added typescript map-type option in Modelina Playground (Issue, PR).
- Added functionality in *spec json schemas* to share *avroSchema_v1* and *openapiSchema_3_0* to all versions (Issue, PR).

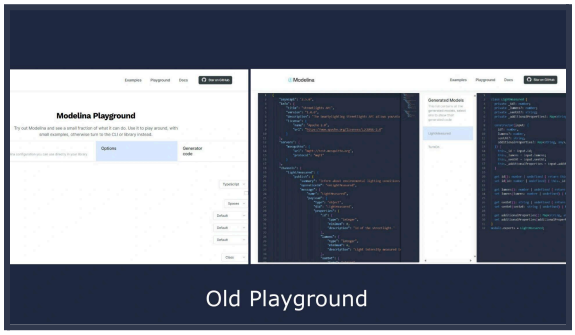
**December 2023**

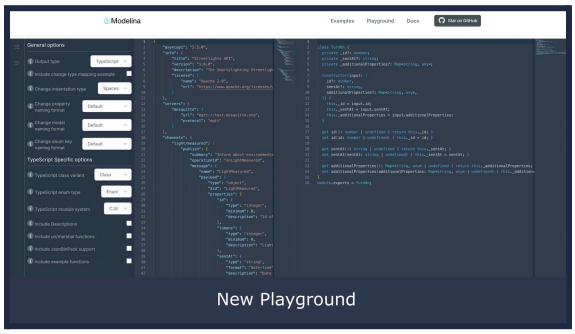
- Added tests for Python constraints in Modelina (Issue, PR).
- Wrote documentation explaining the constraint rules for Python in Modelina (Issue, PR).
- Multiple bug reports in Modelina Documentation (Issue)
- Bug report: Update the store link in the Modelina Website (Issue)

- Migrated the Modelina Website from class-based react components to functional components (Issue, PR) which made the codebase cleaner, easier to understand, and up-to-date with the latest development techniques.

**November 2023**

- Designed and developed a new layout for Modelina Playground (Issue, PR) which enhanced the accessibility and design of the playground.




**October 2023**

- Created Issue: Wrong Instructions in the readme.md to run the server-api locally

# About Me

I am an Engineering Physics Undergraduate from the National Institute of Technology Hamirpur. I am a Web Developer and a Quantum Computing Researcher. In my free time, I love to travel and play video games (BGMI and Clash of Clans being at the top).

## Why I am interested in this project and why I am the right person for this project:

My contributions to the Modelina website and migration of the AsyncAPI website have developed my deep interest and understanding of the website and UI development of AsyncAPI tools. I have deep knowledge of the code base, the design principles, and the development styles of not only the AsyncAPI Website but also the Modelina Website. Through this project, I can contribute further to the AsyncAPI Initiative in this domain and enhance my development skills under the close guidance of mentors who have vast experience in development and learn best development practices from them. Since I have also a deep knowledge of the Modelina Website and Playground, I will be able to maintain a consistent development style across both projects.

## Work Preference:

I prefer maintaining an organized work table. I use notion and my diary to schedule and track my tasks and divide my day in the morning. Once or twice a week, analyze my progress and work. After switching multiple times between Windows and Linux Environments, I have found my peace in working in a Windows Subsystem for Linux (WSL) environment. I love to master the tools and technologies I use. While selecting technologies to solve any problem/or build a new product, I prefer to pick as less number of tools as possible that can cover the entire problem domain to find a balance between cost and ease of maintenance in the future. I prefer discussing with other developers in and outside the team before adding any technology to my stack and start working on any project.

I mainly develop in JavaScript/TypeScript, Python, and C++. For the front end, I mostly use React.js and Next.js. For UI development, I use ShadCN UI, Tailwind CSS, and Storybook. The most used cloud database by me is MongoDB Atlas. For the backend, I use Express and Socket.io. For testing APIs, I use Postman. For UI designing I use Figma. For scientific work (Quantum Computing and Machine Learning), I use Python and libraries like Qiskit and PyTorch. I use C++ to practice my DSA skills. I would like to mention that these technologies represent my commonly used tools, and I am proficient in various other libraries and adaptable to new ones as needed. For development purposes, I mainly use Visual Studio Code Editor in WSL and Visual Studio IDE.

When stuck with errors, I head toward the Stack overflow and Pieces for Developers (an AI-enabled secure productivity tool with an on-device copilot). Although Pieces can solve my errors quickly I avoid using it as my first option because searching on the internet, Stack Overflow, and reading blogs gives chance to read stories of different developers and learn about other scenarios where similar errors can occur (Yeah it's little time-consuming but who doesn't loves spicy entertainments and reading about dumb errors developers get stuck with 😅).