ASYNCAPI

ASYNCAPI CONFERENCE WEBSITE UI KIT DEVELOPMENT

April 6, 2025

AKSSHAY BALASUBRAMANIAN
aksshaybala8846@gmail.com
github.com/aksshay88
Aksshay88

Contents

About Me

Hi, I'm Aksshay Balasubramanian, a 3rd-year B.E. Computer Science Engineering student at Chennai Institute of Technology with a deep passion for AI, open-source, and competitive programming. My programming journey began in 12th grade, and since then, I have explored diverse technologies, from system design to cloud computing.I actively contribute to **open-source projects** and have collaborated with organizations like **AsyncAPI, Google Season of docs , and stdlib-js** As a **GSoC aspirant**, I strive to build impactful projects that improve developer experience and community engagement.

- Email: aksshaybala8846@gmail.com
- Github: aksshay88
- Location: India (IST)
- First Language: Tamil
- Known Languages: English, Japanese ,Tamil

Previous Works

Open Source

Organization	**Pull request**	**Issue**
AyncAPI	1 merged, 1 open , 1 closed	2 closed, 1 open
Mystory(GSSOC'24)	4 merged , 2 closed	8 closed , 1 open
stdlib-js	1 merged	-

Table 1: Contributions to Open Source Projects

Meetings

- Community Working Group/Design Meetings/2025-02-20 - Designing Conference,Mascot Elements,Speaker Banner Design Discussion
- Community Working Group/Design Meetings/2025-03-25 - Guidelines for the Meetup section
- GSOC Introduction By @ace /2025-03-14 - GSOC'25 Experience for AsyncAPI

5XA

I cofounded a **student-led open source organization** during my second year of college, together with my friends. Currently, I serve as the **President**. Our organization conducts **workshops, hosts hackathons, and organizes competitions** focused on open-source development.
Our mission is to educate students about the significance of open source and to encourage their active participation in the community.

Recent Activities

- **Participated in** Local Host: Chennai (CIT) at **FossHack 2025** by **FossUnited**

Projects

I have worked on **multiple projects**—one during my internship at **IITM Pravartak Technologies Foundation** and another one for **EDII – TN (National Level Hackathon)**, where the project secured **1st place (Cold-Recog)** in the competition.

- https://github.com/Team-5XA/cold-recog

I am proficient in technologies such as:

- **Nextjs**
- **Storybook**
- **TypeScript**
- **Authentication (JWT, OAuth)**
- **TailwindCSS**
- **LangChain**
- **ChromaDB**
- **AWS**
- **Ollama**
- **Hugging Face**
- **Docker**
- **AWS**
- **RAG**
- **JS**
- **TS**
- **REST API**
- **GIT**
- **Prompt Engineering**

These skills are particularly valuable for this project. For more details on my other skills, check out my **Resume** and **GitHub**.

AVAILABILITY

By the end of April, my end semester exams will be over, and I will move on to my final year. At my college, there are **no coursework requirements** in the final year, as all subjects are completed in the previous semesters. The final year is entirely dedicated to internships.
My college is familiar with the **Google Summer of Code** program, thanks to past contributors, and provides **full-time support for students** to work on GSoC.

PROJECT DETAILS

Name: AsyncAPI Conference Website UI Kit Development

Description: Develop a **comprehensive UI Kit** to ensure **design consistency**, **modularity**, and **maintainability** for the **AsyncAPI Conference website**. The project focuses on building **reusable components**, integrating **Storybook**, and improving the **user interface experience**. It requires expertise in **React**, **TypeScript**, and **UI/UX design** to create a structured component library. The final outcome will be a **scalable** and **well-documented UI system** for seamless development.
Project Length: 350 hours
Difficulty: Medium
Coding Mentors: Azeez Elegbede, V Thulisile Sibanda

SOLUTION

Project Overview

The goal of this project is to develop a comprehensive and reusable UI Kit to ensure design consistency, modularity, and maintainability for the AsyncAPI Conference website. This UI Kit will provide a structured set of pre-designed components, enabling developers to build and extend the website efficiently while maintaining a cohesive user experience. By integrating Storybook, the project will offer a centralized environment for component documentation, testing, and visualization, streamlining the development process.

The focus is on leveraging React and TypeScript to create an optimized, scalable, and accessible UI system. Additionally, the UI Kit will enhance developer productivity by reducing redundancy and ensuring uniformity across different sections of the website. This project aims to contribute to the AsyncAPI ecosystem by delivering a well-documented and maintainable design system that aligns with modern web development practices

Deliverables

- **Migration to TypeScript:**
  - Refactor the AsyncAPI Conference website codebase to **TypeScript** for better maintainability and scalability.
- **Badge Creator:**
  - Implement a **drag-and-drop badge creation tool** with **real-time preview**.
  - Generate **customized badges** for attendees, speakers, and organizers.
- **Personalized Event Schedule:**
  - Allow users to create and view **custom schedules**.
  - Notify users about **session changes and reminders**.
- **QR Code Check-In System:**
  - Generate **unique QR codes** for attendees.
  - Enable **fast event check-in** and attendance tracking.
- **Meetup Hosting and Archive:**
  - Allow users to **host meetups** related to AsyncAPI topics.
  - Provide a **Meetup Dashboard** for users to **create, join, and manage meetups**.
  - Display an archive of **previous meetups**, including recordings, summaries, and key discussions.
- **Analytics and Insights Dashboard:**
  - Track **attendance, engagement, and feedback** with data visualization.
  - Provide **exportable reports** for event organizers.
- **Role-Based Access Control (RBAC):**
  - Implement **secure access** for different user roles (admin, speaker, attendee).
  - Use **JWT authentication** to protect endpoints.

Executive Summary

This project focuses on developing a **comprehensive UI Kit** to enhance the design consistency, modularity, and maintainability of the **AsyncAPI Conference website**. By migrating the codebase to **TypeScript** and implementing reusable components, the system will ensure a scalable and structured development approach.

The UI Kit will be built using **React** and integrated with **Storybook** for component documentation and testing. A structured design system will improve the user experience, ensuring accessibility and uniform styling across the platform. To enhance collaboration, **design tokens** will be implemented for maintaining consistent typography, spacing, and color schemes.

Additionally, **new features** such as a **Badge Creator**, **Meetup Hosting**, and **Automated Email System** will be introduced. The **Badge Creator** will allow attendees to generate personalized event badges, while the **Meetup Hosting** feature will enable users to organize and join discussions before and after the event. The **Automated Email System** will ensure attendees receive timely reminders and post-event summaries.

DESIGN

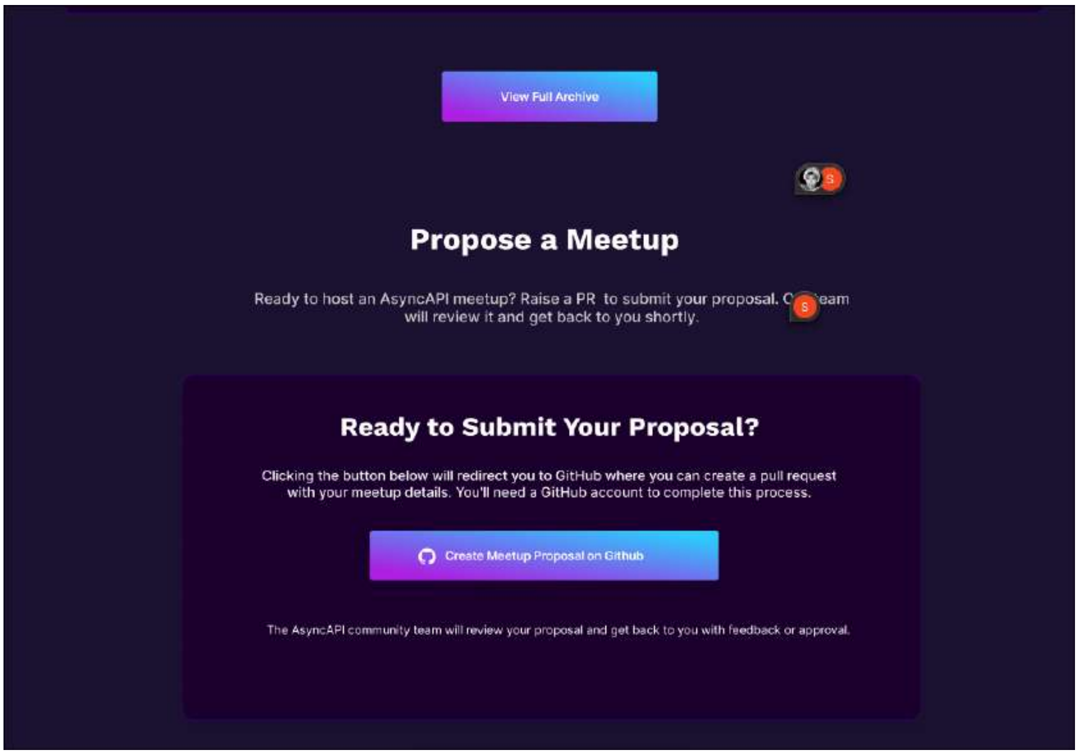
I am proposing only the layout of the interface, not the specific style or color scheme, as these aspects are best discussed with the UI/UX team to ensure alignment with AsyncAPI's design standards. I have outlined two possible layout options, but we can proceed with either of these or adopt an entirely different layout if AsyncAPI already has a preferred design in mind. I am fully flexible and open to implementing any design that best fits the project's needs.

All UI components and pages will follow the official design specifications provided by the AsyncAPI team.

Design Source

- Figma File: AsyncAPI Conf 2025 Design

Design - I


View Full Archive

Propose a Meetup

Ready to host an AsyncAPI meetup? Raise a PR to submit your proposal. O[illegible]eam will review it and get back to you shortly.

Ready to Submit Your Proposal?

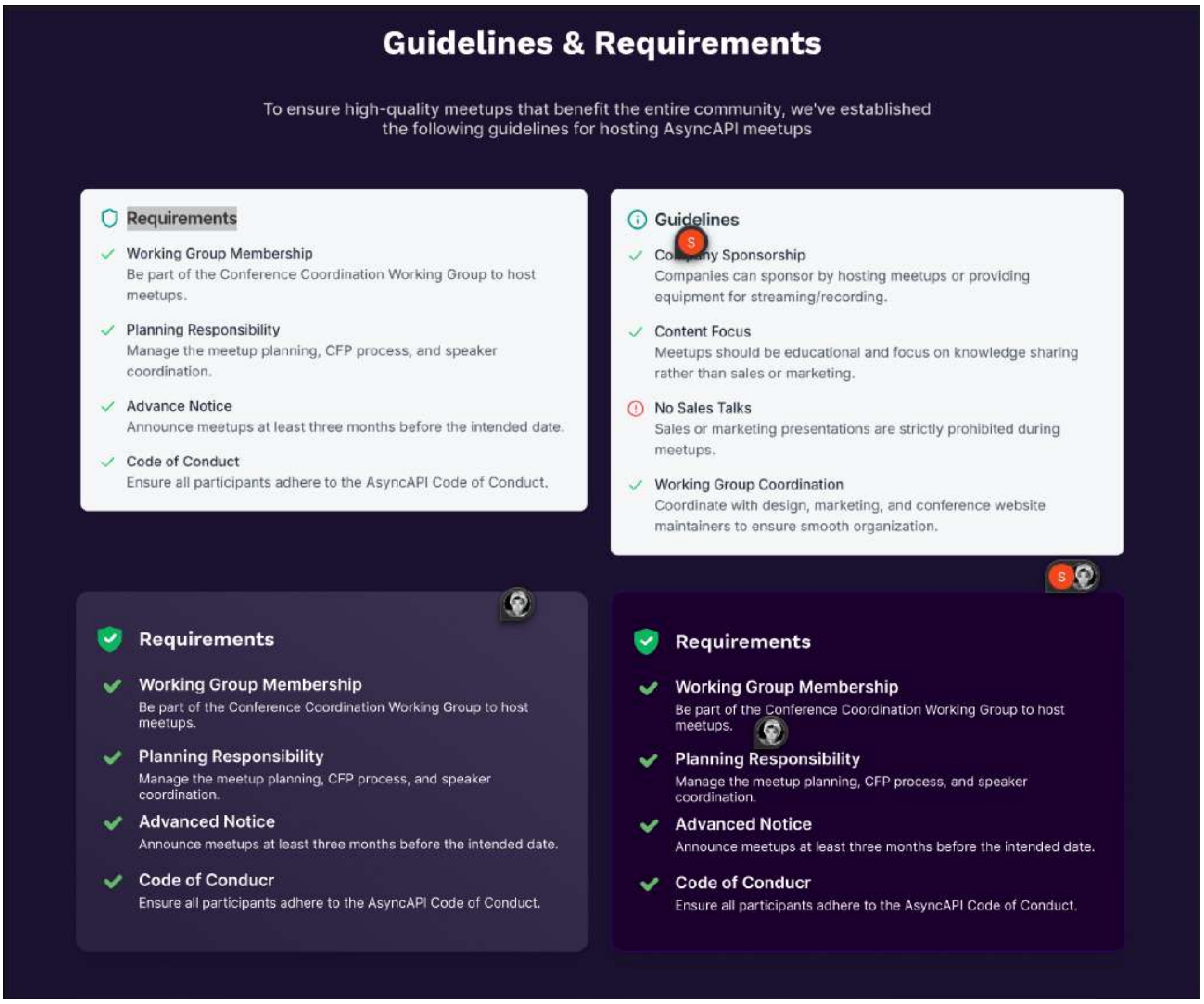
Clicking the button below will redirect you to GitHub where you can create a pull request with your meetup details. You'll need a GitHub account to complete this process.

Create Meetup Proposal on Github

The AsyncAPI community team will review your proposal and get back to you with feedback or approval.

Figure 1: Propose a Meetup

Design - II



Guidelines & Requirements

To ensure high-quality meetups that benefit the entire community, we've established the following guidelines for hosting AsyncAPI meetups

Requirements

- ✓ Working Group Membership
  Be part of the Conference Coordination Working Group to host meetups.
- ✓ Planning Responsibility
  Manage the meetup planning, CFP process, and speaker coordination.
- ✓ Advance Notice
  Announce meetups at least three months before the intended date.
- ✓ Code of Conduct
  Ensure all participants adhere to the AsyncAPI Code of Conduct.

Guidelines

- ✓ Co[illegible]ny Sponsorship
  Companies can sponsor by hosting meetups or providing equipment for streaming/recording.
- ✓ Content Focus
  Meetups should be educational and focus on knowledge sharing rather than sales or marketing.
- No Sales Talks
  Sales or marketing presentations are strictly prohibited during meetups.
- ✓ Working Group Coordination
  Coordinate with design, marketing, and conference website maintainers to ensure smooth organization.

Requirements

- ✓ Working Group Membership
  Be part of the Conference Coordination Working Group to host meetups.
- ✓ Planning Responsibility
  Manage the meetup planning, CFP process, and speaker coordination.
- ✓ Advanced Notice
  Announce meetups at least three months before the intended date.
- ✓ Code of Conducr
  Ensure all participants adhere to the AsyncAPI Code of Conduct.

Requirements

- ✓ Working Group Membership
  Be part of the Conference Coordination Working Group to host meetups.
- ✓ Planning Responsibility
  Manage the meetup planning, CFP process, and speaker coordination.
- ✓ Advanced Notice
  Announce meetups at least three months before the intended date.
- ✓ Code of Conducr
  Ensure all participants adhere to the AsyncAPI Code of Conduct.

Figure 2: Guidelines and Requirements

Indicator

Tech Stack

- **TypeScript**
- **React**
- **Next.js**
- **TailwindCSS**
- **Storybook**
- **JWT Authentication**
- **SendGrid (Email Service)**

IMPLEMENTATION

This section provides a comprehensive breakdown of the project's implementation, divided into several key components.

User Interface (UI)

For the UI, we will follow the existing **AsyncAPI conference website** design principles while incorporating **Nextjs** for frontend development. We will utilize the following key components:

- `Nextjs` – Material design framework for styling.
- `Zustand` – Lightweight state management for handling UI interactions.
- `next Router` – Managing page navigation.

Additionally, we will develop custom UI components within `components/` and extend existing elements to match the proposed design.

Theme Toggle

1. A `ToggleButton` or custom switch component built with Tailwind CSS or shadcn/ui will allow users to switch between light and dark modes.
2. State management using `useContext` or `Zustand` to persist theme preference across components.
3. User preference will be saved in `localStorage` and applied on load via a custom `useEffect` hook.
4. Automatic detection and synchronization with system theme using `window.matchMedia` and the `next-themes` library.

Additionally, we will develop custom UI components within `components/` and extend existing elements to match the proposed design.

0.1 REQUIRED UI COMPONENTS

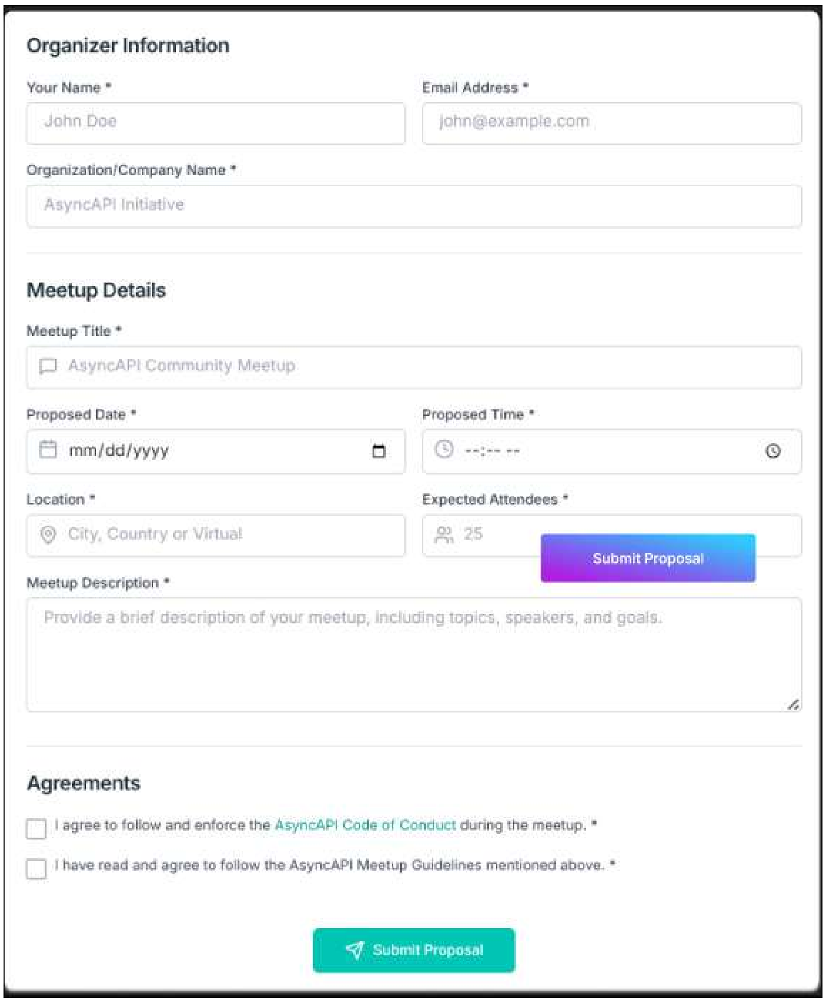
The following components will be essential for the project:

- **Meetup Submission Form** – Allows users to propose and submit meetup details.
- **Meetup Listing Cards** – Displays upcoming meetups dynamically.
- **Theme Toggle** – Enables switching between light and dark themes.
- **Search Bar(using Algora)** – Helps locate specific meetups quickly.
- **Notification System** – Alerts hosts and attendees of event updates.

0.1.1 Meetup Submission Form

The form will include:

- **Title Input** – Allows users to name their meetup.
- **Date & Time Picker** – Supports different time zones.
- **Location Selector** – Virtual or Physical event.
- **Description Field** – Rich text editor for event details.
- **Image Upload** – Drag-and-drop support for cover images.
- **Submit Button** – Sends meetup details for review.



Organizer Information

Your Name *

John Doe

Email Address *

john@example.com

Organization/Company Name *

AsyncAPI Initiative

Meetup Details

Meetup Title *

AsyncAPI Community Meetup

Proposed Date *

mm/dd/yyyy

Proposed Time *

--:-- --

Location *

City, Country or Virtual

Expected Attendees *

25

Submit Proposal

Meetup Description *

Provide a brief description of your meetup, including topics, speakers, and goals.

Agreements

☐ I agree to follow and enforce the AsyncAPI Code of Conduct during the meetup. *

☐ I have read and agree to follow the AsyncAPI Meetup Guidelines mentioned above. *

Submit Proposal

Figure 3: Design By AsyncAPI - Meetup Form

0.1.2 Meetup Listing Cards

Each meetup card will display:

- **Event Name** – Title of the meetup.
- **Date & Time** – Scheduled timing with timezone support.
- **Host Information** – Name and organization of the host.
- **CTA Button** – “View Details” to navigate to the meetup page.

0.1.3 Theme Toggle

1. A `ToggleButton` component from will handle light/dark mode toggling.

2. State management with `zustand` to remember user preferences.
3. Automatic synchronization with system theme settings.

Badge Creator Utility

The **Badge Creator** is a standalone but integrated module that generates digital badges for event contributors and attendees. It is designed with dynamic input, customization, and export capabilities in mind.

Purpose:

To enable organizers to programmatically or manually generate and issue badges for community engagement, fostering recognition and gamification within AsyncAPI events.

Frontend

- **Form-Based UI**
  - Fields: Badge title, description, recipient name, date, badge type.
  - Upload field for optional badge background image.
  - Live preview section with real-time updates based on user input.
- **Customization Options**
  - Font selection, badge color customization, and icon choices.
  - Predefined templates for common roles: speaker, host, attendee.
- **Export Options**
  - Downloadable badge in `.svg` and `.png` formats.
  - Social sharing links or embeddable profile badge URLs.

Backend

- **Badge Metadata API**
  - Endpoint to create and store badge metadata linked to user and event IDs.
  - Endpoint to retrieve all badges associated with a specific user.
- **PDF/SVG Generation Service**

  - Uses `Node.js` with `Puppeteer` or SVG libraries to generate downloadable outputs.
  - Optionally embeds QR codes that link to the event page or user's profile.

- **Integration with Meetup System**
  - Automatic badge issuance when an event is marked as "Completed".
  - Admin interface for manually issuing or revoking badges.

Badge Template API Endpoint

- The following Django REST Framework view supports both `GET` and `POST` methods to retrieve and create badge templates.
- This is a core API used by the Badge Creator module to store and fetch reusable badge designs.

Listing 1: Django REST API for BadgeTemplate

```
from rest_framework import status
from rest_framework.decorators import api_view
from rest_framework.response import Response
from .models import BadgeTemplate

@api_view(['GET', 'POST'])
def badge_templates(request):
    if request.method == 'GET':
        templates = BadgeTemplate.objects.all()
        serializer = BadgeTemplateSerializer(templates, many=True)
        return Response(serializer.data)

    elif request.method == 'POST':
        serializer = BadgeTemplateSerializer(data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data,
                                status=status.HTTP_201_CREATED)
        return Response(serializer.errors,
                            status=status.HTTP_400_BAD_REQUEST)
```

MeetupCard Component

Component Overview

The `MeetupCard` is a reusable UI component for displaying information about past or upcoming meetups. It supports dynamic props for date, format, URLs, and host.

Props Interface

Listing 2: Props Interface

```
interface MeetupCardProps {
  title: string;
  date: string;
  format: string;
  recordingUrl: string;
  slidesUrl: string;
  host?: string;
}
```

Component Implementation

Listing 3: MeetupCard.tsx

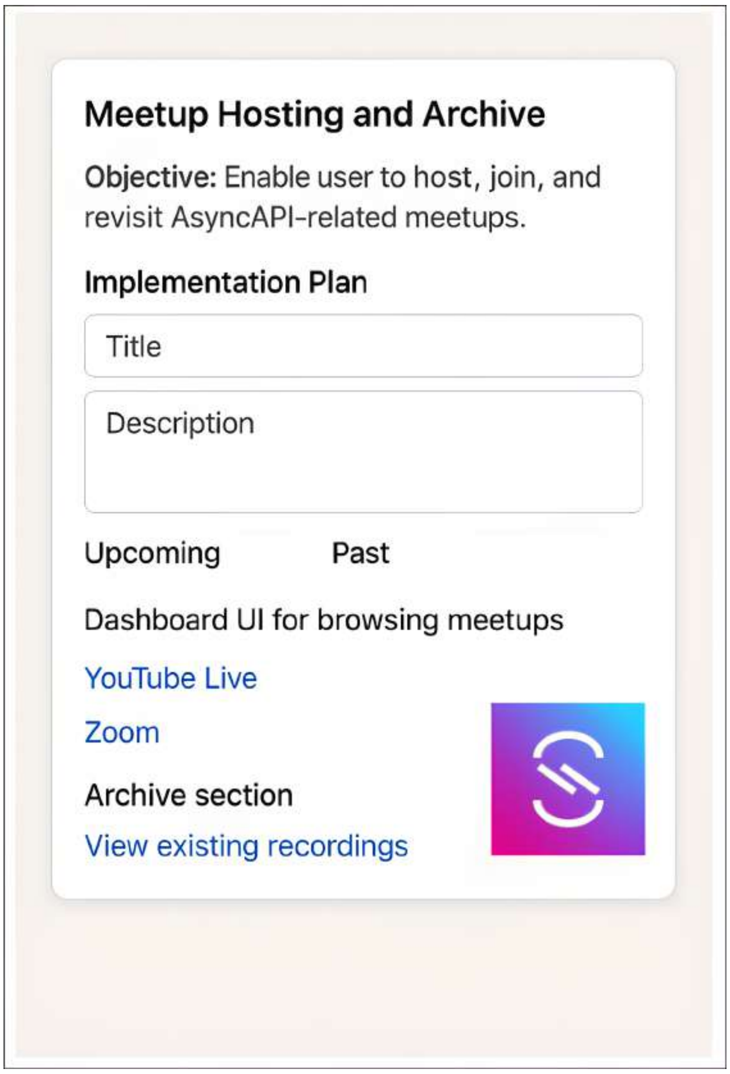
```
export const MeetupCard: React.FC<MeetupCardProps> = ({
  title,
  date,
  format,
  recordingUrl,
  slidesUrl,
  host = 'AsyncAPI Initiative',
}) => (
  <div className="border rounded-xl p-4 shadow-md space-y-2">
```

```
    <h2 className="text-xl font-bold">{title}</h2>
    <ul className="text-sm text-gray-700">
      <li><span className="font-semibold text-primary">Date:</span
         > {date}</li>
      <li><span className="font-semibold text-primary">Format:</
         span> {format}</li>
    </ul>
    <div className="space-x-4 mt-2">
      <a href={recordingUrl} className="text-secondary font-
         semibold">Watch Recordings</a>
      <a href={slidesUrl} className="text-secondary font-semibold"
         >View Slides</a>
    </div>
    <p className="text-xs text-gray-500 mt-2">Hosted by {host}</p>
  </div>
);
```

Example Card

Listing 4: Example Usage

```
<MeetupCard
  title="AsyncAPI Conference 2023"
  date="October 10, 2023"
  format="Virtual"
  recordingUrl="https://example.com/recording-2023"
  slidesUrl="https://example.com/slides-2023"/>
```


Meetup Hosting and Archive

Objective: Enable user to host, join, and revisit AsyncAPI-related meetups.

Implementation Plan

Dashboard UI for browsing meetups

Archive section

Figure 4: Design I MEETUP CARD COMPONENT

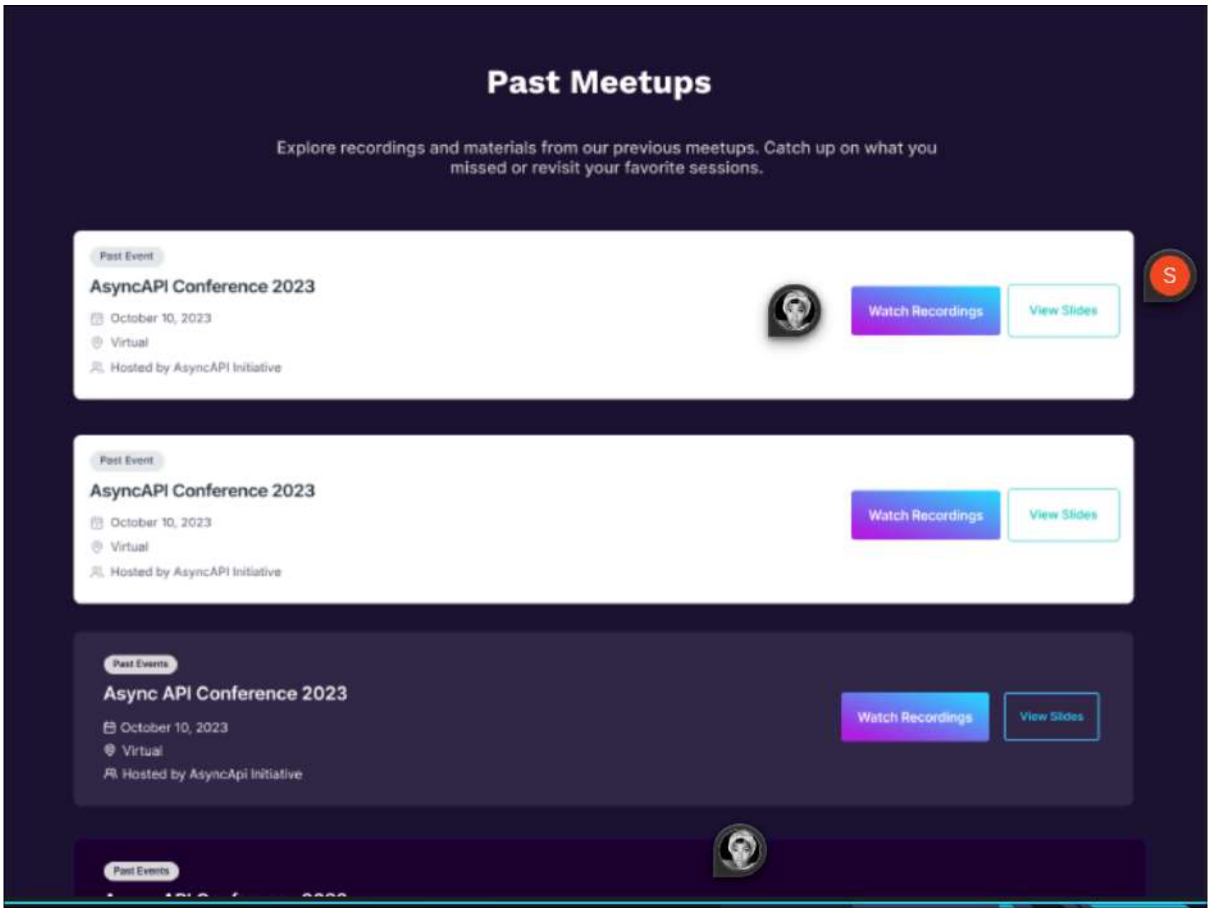


Figure 5: Design II By Asyncapi

Component Development with Storybook

Storybook will serve as the central environment for building, testing, and documenting UI components used across the Badge Creator Utility and broader AsyncAPI interface.

Core Responsibilities

- **Isolated Component Development**
  - Each UI element—such as buttons, forms, previews, and modals—will be developed in isolation to ensure reusability and visual consistency.
- **Component Documentation**
  - Storybook automatically generates interactive documentation that can be shared with designers, developers, and stakeholders.

  - Each story includes usage examples, prop definitions, and visual variants (e.g., dark mode, hover state).

- **Visual Testing**
  - UI changes will be visually validated using Storybook's built-in testing tools (e.g., Chromatic).
  - Regression checks will help maintain visual integrity as the UI evolves.

Component Integration Strategy

- **Component Library Structure**
  - All UI components will live in the `components/` directory, grouped by purpose (e.g., `BadgePreview/`, `FormInput/`, `ExportButton/`).
  - Story files (`*.stories.tsx`) will be created for each component to enable preview, control knobs, and live interactions.
- **Mapping to AsyncAPI Requirements**
  - Components will be aligned with the AsyncAPI UI Kit standards and mapped to the required UI elements defined in the project scope.
  - Final implementation will involve importing and assembling these components in pages or views based on the AsyncAPI layout (e.g., sidebar navigation, badge cards, meetup submission).
- **Cross-Team Collaboration**
  - Designers can preview stories without running the full application.
  - Contributors can reuse documented components across multiple features, reducing duplication.

Benefits of Storybook Usage

- Faster UI iteration and feedback cycles.
- Centralized reference for all reusable components.
- Streamlined onboarding for new contributors.
- Helps enforce visual and functional consistency across the app.

Personalized Schedule Component

The Personalized Schedule component allows users to curate their own agenda from the list of available sessions, enhancing the event experience through interactivity and customization.

Core Responsibilities

- **Session Selection Interface**
  - Users can browse all scheduled sessions in a calendar or list view.
  - Sessions are selectable via checkboxes or drag-and-drop interaction.
- **User-Specific Schedule Storage**
  - Selections are stored locally (in browser) and remotely via API (`/api/schedule`).
  - Authenticated users can persist their schedule across devices.
- **Calendar Integration**
  - Built using `@fullcalendar/react` for dynamic rendering.
  - Supports day, week, and list views with real-time updates.
- **Responsive Accessible UI**
  - Fully responsive layout optimized for both desktop and mobile.
  - Keyboard navigation and screen reader support included.

Component Breakdown

- `components/ScheduleCalendar.tsx` – Main calendar UI using FullCalendar.
- `components/SessionCard.tsx` – Reusable cards for each session item.
- `hooks/useSchedule.ts` – Custom hook to manage user's schedule state.
- `pages/api/schedule.ts` – API endpoint for schedule persistence.

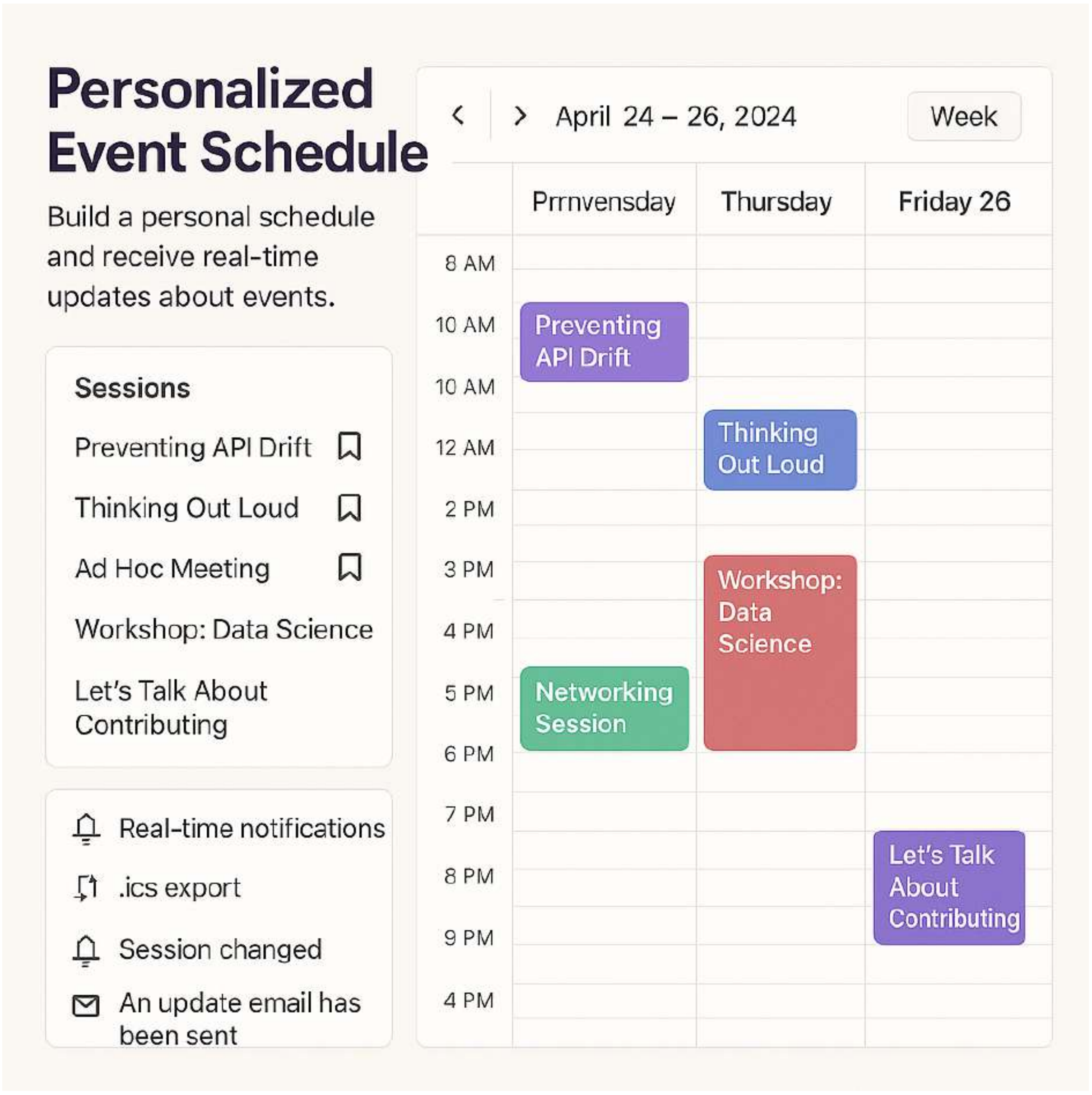


Figure 6: Design I MEETUP CARD COMPONENT

Analytics and Insights Dashboard

The dashboard enables organizers to monitor platform engagement, session popularity, and attendee behavior through interactive visualizations and exportable reports.

Core Objectives

- Track user interactions such as session views, check-ins, and personalized schedules.

- Visualize data trends and metrics with charts and graphs.
- Enable export of aggregated insights for external analysis or reporting.

Implementation Details

- **Event Tracking**
  - Tools: `PostHog`, `Amplitude`, or custom event tracking.
  - Events: page views, session joins, schedule edits, feedback submissions.
- **Data Visualization**
  - Libraries: `Recharts`, `Chart.js`, or `@nivo`.
  - Charts: bar, pie, and line charts to show usage and trends.
  - Component: `components/AnalyticsDashboard.tsx`.
- **Backend Aggregation**
  - Endpoint: `GET /api/analytics`.
  - Aggregates metrics like daily active users, session attendance, average engagement.
  - Filters: date range, session ID, user segments.
- **Data Export**
  - CSV export via REST API: `GET /api/analytics/export`.
  - Utility to convert JSON to downloadable CSV on frontend.
  - Button: `<ExportCSVButton />`.

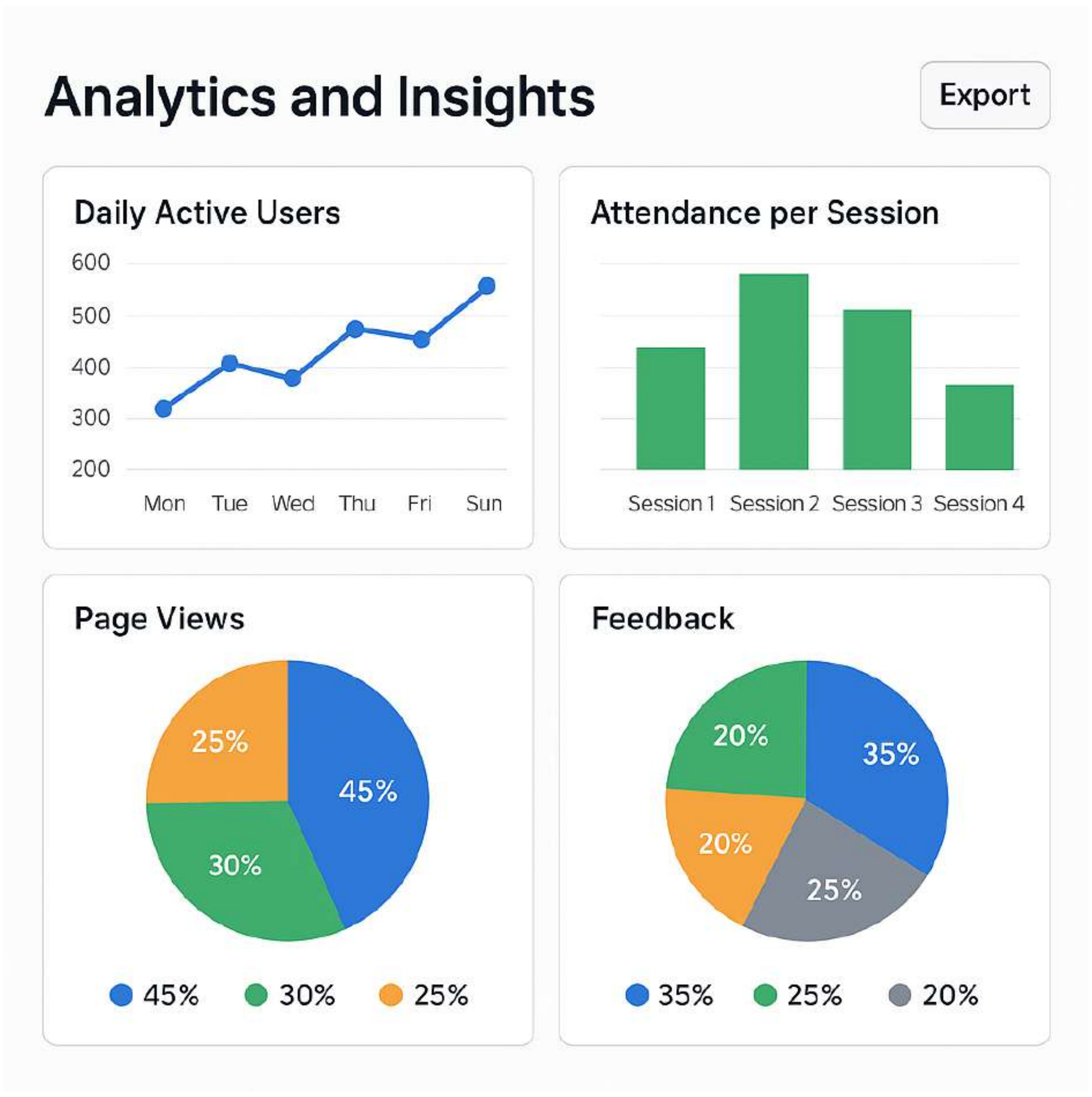


Figure 7: Analytics and Insights Dashboard integration with QR check-in system

QR Code Check-In System

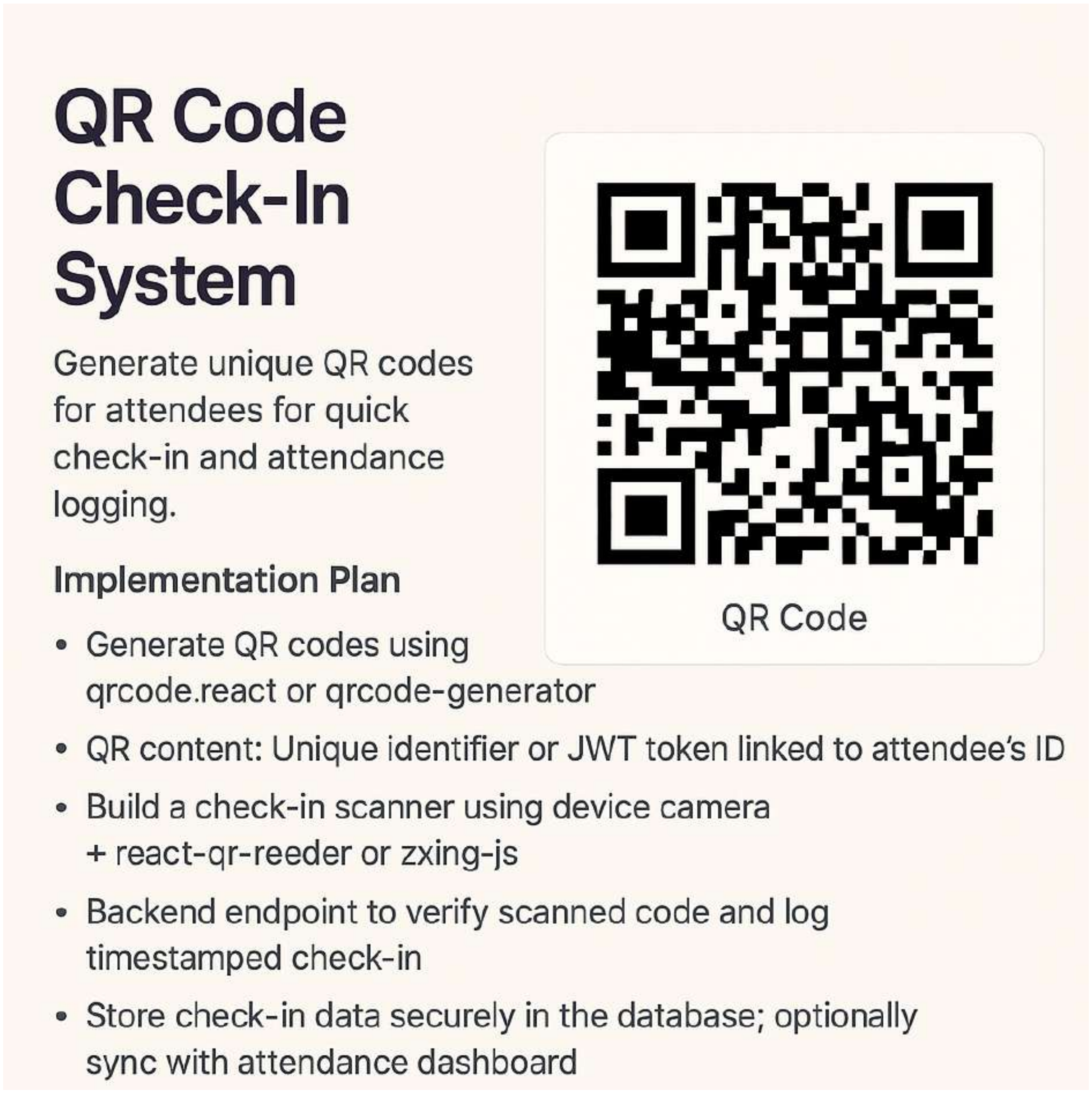
The QR Code Check-In System streamlines event entry and enables accurate attendance tracking by integrating client-side QR generation with real-time scanning and backend verification.

Core Objectives

- Generate unique QR codes per attendee for secure, efficient check-in.
- Log attendance with timestamps and persist data in the backend.

Implementation Details

- **QR Code Generation**
  - Libraries: `qrcode.react` or `qrcode-generator`.
  - QR payload: unique identifier or signed JWT referencing user ID.
  - Component: `components/QRCode.tsx`.
- **Check-In Scanner**
  - Libraries: `react-qr-reader` or `zxing-js/browser`.
  - Scans attendee QR via webcam or mobile camera.
  - Component: `components/QRScanner.tsx`.
- **Backend Verification**
  - Endpoint: `POST /api/checkin`.
  - Validates QR payload and logs check-in with timestamp.
  - Secured with token-based auth and input validation.
- **Data Storage and Sync**
  - Check-in data stored in Supabase or PostgreSQL.
  - Optionally linked with attendance analytics dashboard.
  - Structure: `checkins (user_id, session_id, timestamp)`.



QR Code Check-In System

Generate unique QR codes for attendees for quick check-in and attendance logging.

Implementation Plan

- Generate QR codes using qrcode.react or qrcode-generator
- QR content: Unique identifier or JWT token linked to attendee's ID
- Build a check-in scanner using device camera + react-qr-reeder or zxing-js
- Backend endpoint to verify scanned code and log timestamped check-in
- Store check-in data securely in the database; optionally sync with attendance dashboard

Figure 8: QR Code Check-In System

Chatbot Integration

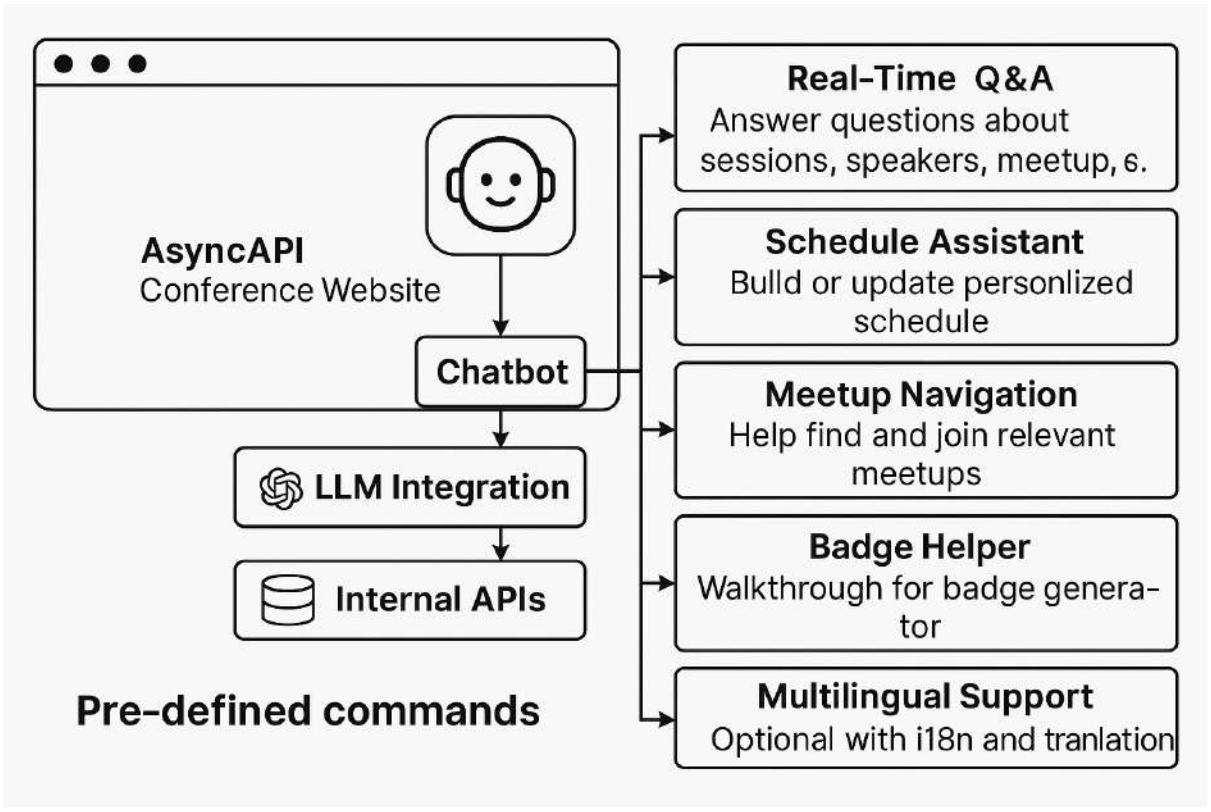
- To enhance user engagement and provide real-time assistance, I will integrate an intelligent chatbot into the AsyncAPI Conference Website.
- The chatbot will be embedded as a floating widget accessible from every page, designed with minimal UI to maintain design consistency.

Key Features

- **Real-Time Q&A:** The bot will answer questions related to sessions, speakers, meetups, schedule conflicts, and FAQs.
- **Schedule Assistant:** Users can ask the bot to build or update their personalized schedule.
- **Meetup Navigation:** The bot can help users find and join relevant meetups based on topics or tags.
- **Badge Helper:** Walkthrough for using the badge creator and saving/exporting designs.
- **Multilingual Support:** Optional integration with i18n libraries and translation APIs to support global users.

Technical Implementation

- Built with `react-chatbot-kit` or `Botpress`, styled with Tailwind CSS for seamless UI blending.
- Integrated with internal APIs for dynamic responses (e.g., fetching session details or user schedule).
- LLM-based fallback using OpenAI or similar for generalized queries, rate-limited and privacy-respecting.
- Context-aware interactions based on user role (attendee, speaker, admin) and authentication state.
- Event tracking hooked into the chatbot to analyze usage and improve guidance quality.



AsyncAPI
Conference Website
Chatbot
LLM Integration
Internal APIs
Pre-defined commands
Real-Time Q&A
Answer questions about sessions, speakers, meetup, ɕ.
Schedule Assistant
Bulld or update personlized schedule
Meetup Navigation
Help find and join relevant meetups
Badge Helper
Walkthrough for badge genera-tor
Multilingual Support
Optional with i18n and tranlation

Figure 9: Chatbot Implementation

Architecture

Overview

The proposed design will be a modular full-stack application that powers a fully functional, up-to-date website. The technology stack includes:

- **Frontend:** Next.js (React-based) with full TypeScript support
- **UI System:** Component-driven development using Storybook

Frontend Architecture (Next.js + TypeScript + Storybook)

1. TypeScript Migration Strategy The entire frontend will be developed in TypeScript to ensure type safety, better tooling, and fewer runtime errors.

- Convert all source files to `.ts` and `.tsx`
- Enable `strict` mode in `tsconfig.json`

- Shared types/interfaces defined in `types/` or `@types/`
- ESLint + Prettier configured for TypeScript

2. Component System with Storybook All UI elements will be designed as isolated, reusable components with Storybook.

- `components/InputBox.tsx` – Controlled textarea input
- `components/SuggestionCard.tsx` – Displays meetups suggestions
- Each component includes:
  - TypeScript definition
  - Storybook story file (`.stories.tsx`)
  - Unit tests (`.test.tsx`) using Jest + RTL

3. Routing and Pages (Next.js)

- `/` – Landing page
- `/meetup` – asyncapi host a meetup
- `/any func req in future` – Project future requirements

4. Styling and Design System

- TailwindCSS for utility-first styling
- Consistent theme defined in `tailwind.config.js`
- Atomic Design pattern for UI architecture (atoms, molecules, organisms)

The frontend and backend may be colocated in a Turborepo-style monorepo for consistency and shared tooling.

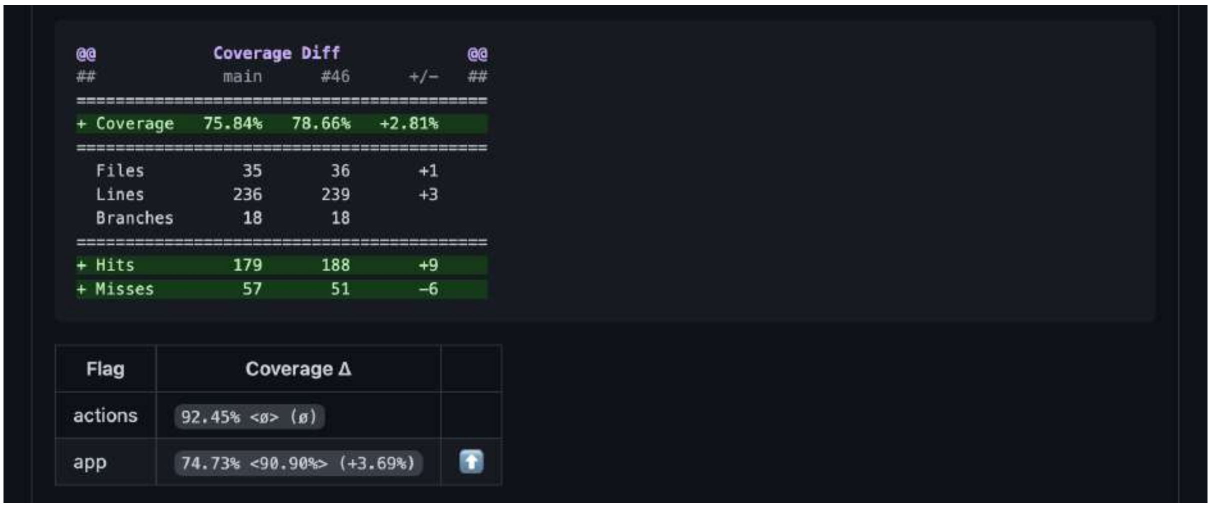
DevOps and Tooling

- **Docker:** Containerized backend and frontend
- **CI/CD:** GitHub Actions to lint, test, and deploy
- **Testing:**
  - Frontend – Jest + React Testing Library + cypress + codecov
  - Backend – Pytest + HTTPX(if required)
- **Deployment:(completely optional)**
  - Frontend via Vercel
  - Backend via Render, Railway, or Fly.io

Tests (Optional): Frontend + CI/CD Coverage

Ensuring **robustness and reliability** is a key part of developing this web-based interface. To support scalable development, we will implement a modern testing workflow using:

- **Cypress** for end-to-end (E2E) and integration tests.
- **GitHub Actions** for continuous testing automation.
- **Codecov** for tracking coverage reports.



```
@@            Coverage Diff            @@
##              main      #46     +/-   ##
==========================================
+ Coverage   75.84%   78.66%   +2.81%
==========================================
  Files          35       36       +1
  Lines         236      239       +3
  Branches       18       18
==========================================
+ Hits          179      188       +9
+ Misses         57       51       -6
```

Flag	Coverage Δ	
actions	92.45% <ø> (ø)	
app	74.73% <90.90%> (+3.69%)	⬆

Figure 10: Codecov Report PR (For tracking the code coverage)

1. Cypress Testing Setup

Cypress is used for browser-based testing. It allows real-time feedback, DOM traversal, and time-travel debugging.

Test targets:

- Component-level testing integrated via **Storybook stories**.
- Full-page E2E tests for user flows (e.g., form submission, navigation).
- Validation of dynamic data, loading states, and error handling.

Example test snippet for form validation:

Listing 5: Example: Cypress Test for Input Validation

```
describe('Check stuff', () => {
  it('shows validation error for empty text', () => {
    cy.visit('/storybook')
    cy.get('button[type=submit]').click()
    cy.contains('Text input is required').should('be.visible')
  })
})
```

2. Storybook Integration with Cypress

- All UI components will have stories written in `.stories.tsx` files.
- `@storybook/test-runner` and `@storybook/cypress` will automate interaction tests for each component.
- This ensures design accuracy and avoids regression when styling or props change.

3. Code Coverage with Codecov

Code coverage metrics will be collected using `nyc` (Istanbul) and reported via **Codecov**. Output includes:

- Function and line-level test coverage.
- Visual diffs on PRs to highlight coverage drop.

4. GitHub Actions CI Pipeline

A GitHub Actions YAML workflow will orchestrate:

- Installation of dependencies.
- Linting and static analysis.
- Cypress tests (headless mode).
- Uploading coverage to **Codecov**.

Example: `.github/workflows/test.yml`

Listing 6: CI Workflow: Cypress + Codecov

```yaml
name: Run Cypress and Upload Coverage

on: [push, pull_request]

jobs:
  test:
    runs-on: ubuntu-latest

    steps:
    - name: Checkout code
      uses: actions/checkout@v3

    - name: Install Node dependencies
      run: npm ci

    - name: Run Cypress tests
      run: npm run cy:run

    - name: Run Coverage
      run: npm run test:coverage

    - name: Upload coverage to Codecov
      uses: codecov/codecov-action@v3
      with:
        token: ${{ secrets.CODECOV_TOKEN }}
```

ARCHITECTURE

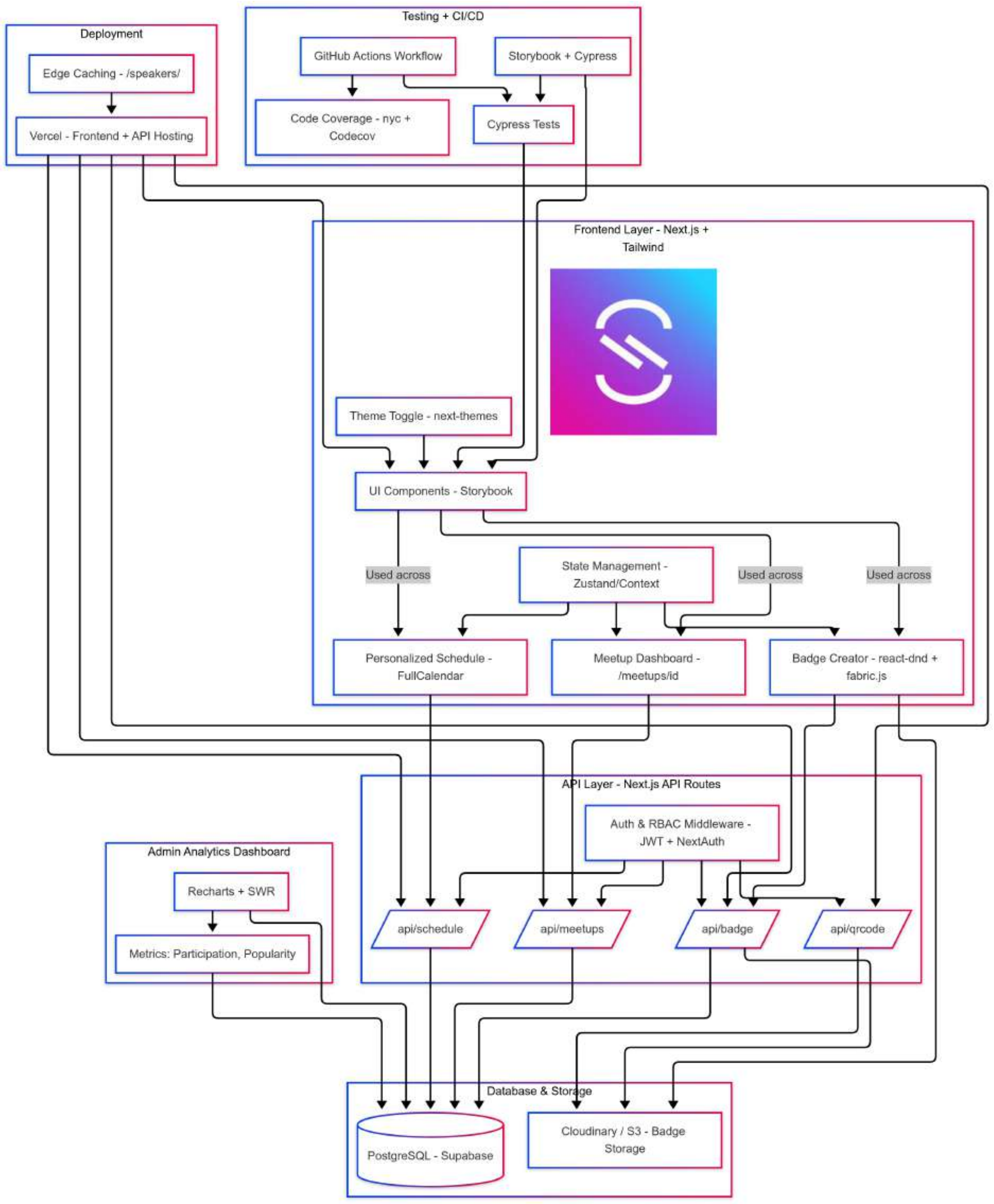

Deployment
Edge Caching - /speakers/
Vercel - Frontend + API Hosting
Testing + CI/CD
GitHub Actions Workflow
Storybook + Cypress
Code Coverage - nyc + Codecov
Cypress Tests
Frontend Layer - Next.js + Tailwind
Theme Toggle - next-themes
UI Components - Storybook
Used across
State Management - Zustand/Context
Used across
Used across
Personalized Schedule - FullCalendar
Meetup Dashboard - /meetups/id
Badge Creator - react-dnd + fabric.js
API Layer - Next.js API Routes
Auth & RBAC Middleware - JWT + NextAuth
api/schedule
api/meetups
api/badge
api/qrcode
Admin Analytics Dashboard
Recharts + SWR
Metrics: Participation, Popularity
Database & Storage
PostgreSQL - Supabase
Cloudinary / S3 - Badge Storage

Figure 11: Final look of Prototype

IMPACT

1. Enhancing User Experience for Conference Attendees

This project will **transform** how users interact with the AsyncAPI Conference website by providing a **consistent, responsive, and customizable UI**. Rather than focusing solely on the backend, it will:

- **Enhance user experience** with reusable and accessible UI components.
- **Improve navigation and usability** via modular, easy-to-maintain components.
- **Increase user engagement** through a visually appealing, user-friendly design.

By implementing a scalable UI kit, this project will deliver a **seamless experience** for both attendees and speakers.

2. Strengthening AsyncAPI's Digital Presence

This project aligns with AsyncAPI's mission of **building robust and accessible web apps**, by:

- Expanding AsyncAPI's conference site to better serve its audience.
- Introducing a **modular design system** reusable across AsyncAPI projects.
- Improving accessibility and responsiveness across all devices.

3. Advancing Open-Source UI Development in the Tech Community

By contributing a **comprehensive UI kit** to the AsyncAPI ecosystem, this project will:

- Show how UI kits foster design consistency and faster web development.
- Encourage use of reusable components in open source for better collaboration.
- Lay a foundation for future UI systems that are both scalable and accessible.

This effort will bridge the gap between **design and development**, allowing developers to contribute easily while improving UX for attendees.

POST-GSOC PLANS & FUTURE OF THE PROJECT

1. Long-Term Sustainability & Maintenance

After GSoC, I will ensure the project stays **maintainable, scalable, and adaptable** by:

- **Actively contributing** to the AsyncAPI community and maintaining the UI kit.
- **Providing thorough documentation** for design and development workflows.
- **Supporting modular updates** that enhance the system without breaking it.

2. Future Enhancements & Improvements

Though the focus is on UI Kit development and TypeScript migration, future work includes:

1. **Mobile Responsiveness**
    - Adapting the UI kit for all screen sizes and browsers.
    - Creating dynamic, responsive components.
2. **Accessibility Improvements**
    - Building **WCAG-compliant components**.
    - Ensuring ARIA compliance with continuous accessibility tests.
3. **UI/UX Customization**
    - Adding a theming system for custom branding and user preferences.

3. My Continued Contribution

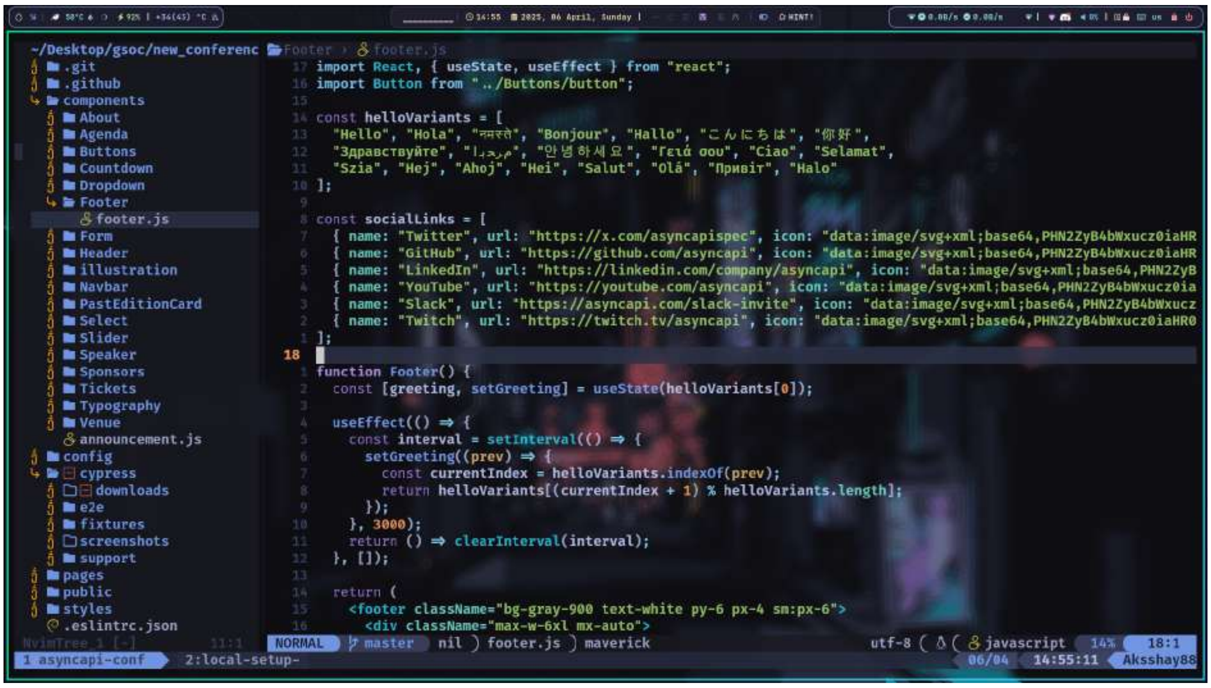
I will stay involved in the AsyncAPI project beyond GSoC by:

- **Onboarding new contributors** and reviewing PRs.
- **Improving UI components** based on feedback and usage.
- **Contributing new features** to enhance the platform.

Local Development Environment

For local development, I use a lightweight and efficient setup optimized for performance and terminal-based workflows:

- **Operating System:** Arch Linux
- **Editor:** Neovim (NVIM) with LSP, Treesitter, and Telescope
- **Terminal Multiplexer:** Tmux for session persistence and window management
- **Shell:** Zsh with plugins (e.g., zsh-autosuggestions, zsh-syntax-highlighting)

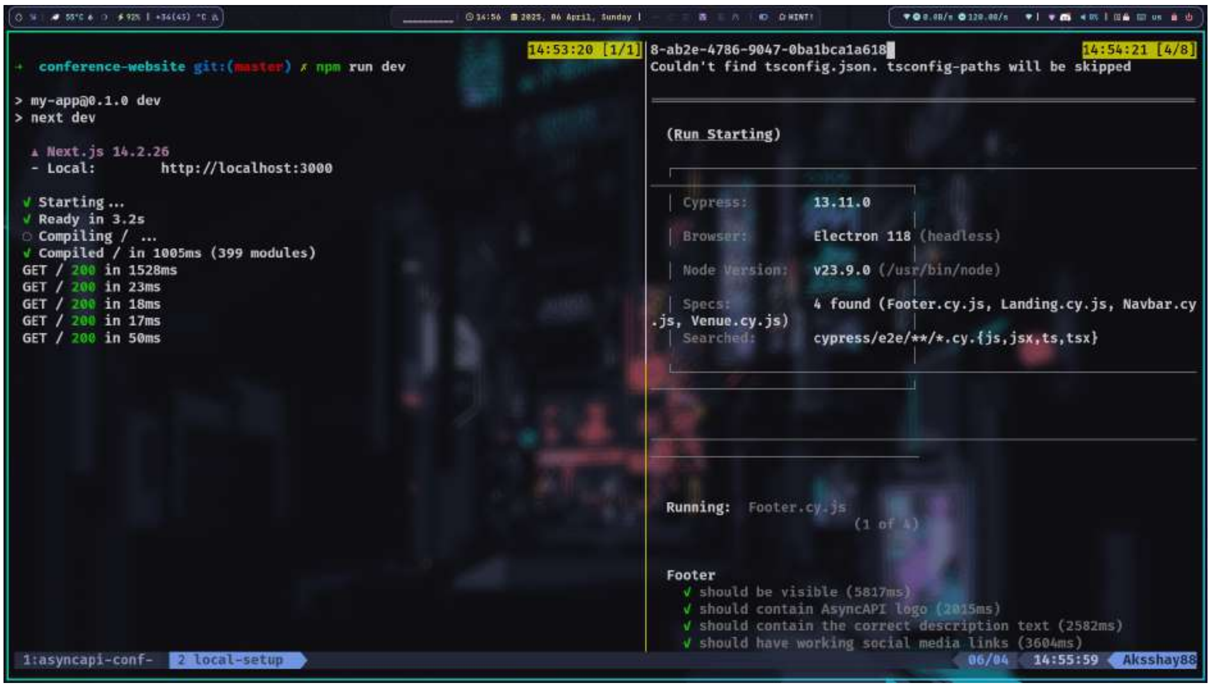


```
~/Desktop/gsoc/new_conferenc
  .git
  .github
  components
    About
    Agenda
    Buttons
    Countdown
    Dropdown
    Footer
      footer.js
    Form
    Header
    illustration
    Navbar
    PastEditionCard
    Select
    Slider
    Speaker
    Sponsors
    Tickets
    Typography
    Venue
    announcement.js
  config
  cypress
    downloads
    e2e
    fixtures
    screenshots
    support
  pages
  public
  styles
  .eslintrc.json
```

```
import React, { useState, useEffect } from "react";
import Button from "../Buttons/button";

const helloVariants = [
  "Hello", "Hola", "नमस्ते", "Bonjour", "Hallo", "こんにちは", "你好",
  "Здравствуйте", "مرحبا", "안녕하세요", "Γειά σου", "Ciao", "Selamat",
  "Szia", "Hej", "Ahoj", "Hei", "Salut", "Olá", "Привіт", "Halo"
];

const socialLinks = [
  { name: "Twitter", url: "https://x.com/asyncapispec", icon: "data:image/svg+xml;base64,PHN2ZyB4bWxucz0iaHR
  { name: "GitHub", url: "https://github.com/asyncapi", icon: "data:image/svg+xml;base64,PHN2ZyB4bWxucz0iaHR
  { name: "LinkedIn", url: "https://linkedin.com/company/asyncapi", icon: "data:image/svg+xml;base64,PHN2ZyB
  { name: "YouTube", url: "https://youtube.com/asyncapi", icon: "data:image/svg+xml;base64,PHN2ZyB4bWxucz0ia
  { name: "Slack", url: "https://asyncapi.com/slack-invite", icon: "data:image/svg+xml;base64,PHN2ZyB4bWxucz
  { name: "Twitch", url: "https://twitch.tv/asyncapi", icon: "data:image/svg+xml;base64,PHN2ZyB4bWxucz0iaHR0
];

function Footer() {
  const [greeting, setGreeting] = useState(helloVariants[0]);

  useEffect(() => {
    const interval = setInterval(() => {
      setGreeting((prev) => {
        const currentIndex = helloVariants.indexOf(prev);
        return helloVariants[(currentIndex + 1) % helloVariants.length];
      });
    }, 3000);
    return () => clearInterval(interval);
  }, []);

  return (
    <footer className="bg-gray-900 text-white py-6 px-4 sm:px-6">
      <div className="max-w-6xl mx-auto">
```

```
NvimTree_1 [-]   11:1   NORMAL   master   nil ) footer.js ) maverick   utf-8 (  ( javascript   14%   18:1
1 asyncapi-conf   2:local-setup-                                          06/04   14:55:11   Aksshay88
```

Figure 12: Local Setup Environment



```
→ conference-website git:(master) ✗ npm run dev

> my-app@0.1.0 dev
> next dev

  ▲ Next.js 14.2.26
  - Local:        http://localhost:3000

 ✓ Starting...
 ✓ Ready in 3.2s
 ○ Compiling / ...
 ✓ Compiled / in 1005ms (399 modules)
 GET / 200 in 1528ms
 GET / 200 in 23ms
 GET / 200 in 18ms
 GET / 200 in 17ms
 GET / 200 in 50ms
```

```
8-ab2e-4786-9047-0ba1bca1a618
Couldn't find tsconfig.json. tsconfig-paths will be skipped

====================================================================

  (Run Starting)

  ┌──────────────────────────────────────────────────────────────┐
  │ Cypress:        13.11.0                                       │
  │ Browser:        Electron 118 (headless)                       │
  │ Node Version:   v23.9.0 (/usr/bin/node)                       │
  │ Specs:          4 found (Footer.cy.js, Landing.cy.js, Navbar.cy
.js, Venue.cy.js)
  │ Searched:       cypress/e2e/**/*.cy.{js,jsx,ts,tsx}           │
  └──────────────────────────────────────────────────────────────┘

────────────────────────────────────────────────────────────────────

  Running:  Footer.cy.js                                    (1 of 4)

  Footer
    ✓ should be visible (5817ms)
    ✓ should contain AsyncAPI logo (2015ms)
    ✓ should contain the correct description text (2582ms)
    ✓ should have working social media links (3604ms)
```

Figure 13: cypress test

TIMELINE

Community Bonding Period

- Deep dive into the AsyncAPI ecosystem, tooling, and existing website infrastructure.
- Collaborate with mentors to finalize the architectural design and feature scope.
- Set up development environment (Next.js, Storybook, TypeScript, CI/CD).
- Plan initial tasks, dependencies, and weekly deliverables.

Week 1: Project Setup & Core Structure

- Set up Next.js app with TypeScript, Tailwind CSS, and Storybook.
- Configure linting (ESLint), formatting (Prettier), and testing (Cypress).
- Implement the theme toggle system with ‘next-themes‘.
- Add static pages for agenda and speakers using ISR.

Week 2: Badge Creator MVP

- Integrate ‘react-dnd‘ and ‘fabric.js‘ to enable drag-and-drop badge design.
- Implement real-time canvas preview.
- Store badge data in component state and begin API integration.

Week 3: Badge Generator Backend

- Build backend route ‘/api/badge‘ for badge export.
- Integrate canvas export (PNG/PDF).
- Add user authentication and session persistence via NextAuth.js.

Week 4: Personalized Schedule Builder

- Implement calendar view with ‘@fullcalendar/react‘.
- Build user interface for adding/removing sessions.
- Store schedule data locally and via ‘/api/schedule‘.

Week 5: Meetup Dashboard - Part 1

- Create dynamic routes ‘/meetups/[id]‘ with ISR.
- Build UI for listing and joining meetups.
- Design database schema in Supabase for meetups and attendees.

Week 6: Meetup Dashboard - Part 2

- Add dashboard for users to create and manage meetups.
- Implement role-based access control (RBAC) via JWT middleware.
- Write Cypress tests for all user flows.

Week 7: QR Code Check-in System

- Add QR code generation via ‘qrcode‘ package.
- Implement check-in verification endpoint.
- Track attendance metrics in Supabase.

Week 8: Analytics Dashboard

- Build admin dashboard using Recharts + SWR.
- Show visual metrics (e.g., session popularity, attendance trends).
- Add data export options for organizers.

Week 9: Polish, Test, & Accessibility

- Improve mobile responsiveness and keyboard navigation.
- Add unit + integration tests for edge cases.
- Fix visual bugs and validate form/error states.

Week 10: Final Refinements & Handoff

- Finalize documentation and deployment guides.
- Record demo video or write blog on project outcomes.
- Discuss feature roadmap with maintainers.
- Submit final evaluation materials.

Time Commitment & Progress Reporting

Time Allocation

- **4-6 hours per day**.
- **Avg 35 hours a week**.

Progress Reporting

1. **Regular Meetings**
    - Open to any meeting platform: **Jitsi**, **Google Meet, Zoom, or others**.
    - Weekly check-ins with mentors to discuss progress, challenges, and next steps.
2. **Private Communication**
    - Comfortable using **Slack, IRC, Discord, Email, or any preferred platform** for quick discussions.
    - Can provide updates and ask for feedback.
3. **Public Updates**
    - **Weekly blog posts** summarizing progress, technical learnings, and challenges faced.
    - Regular updates in **AsyncAPI community channels** to keep everyone informed.

Conclusion

- I am confident in my ability to successfully execute this project and deliver a **modern, feature-rich AsyncAPI Conference Website**. With my experience in **Next.js, TypeScript, full-stack architecture, and developer tooling**, I have the **technical expertise** required to build a scalable, performant, and community-friendly platform.
- Beyond development, I bring a strong background in **open-source contribution, event platforms, and UI/UX best practices**, ensuring this project aligns with the **AsyncAPI Initiative's mission and open governance standards**. My focus on **modular design, clean code, and maintainability** will make the platform future-proof and easier for new contributors to join and extend.

- I am **committed, collaborative, and feedback-driven**, ensuring **transparent and smooth engagement** with mentors and the AsyncAPI community. With clearly defined phases, **milestone-based delivery**, and regular progress updates, I aim to keep the project on track and at a high level of quality.
- By building key features such as a **badge creator, personalized schedule builder, QR check-in system, and meetup dashboard**, this project will significantly enhance the **community experience around AsyncAPI conferences**. I'm excited about the opportunity to create a platform that empowers organizers, speakers, and attendees alike — and to contribute something lasting to the AsyncAPI ecosystem.