Position: Software Development Intern (SDE1 INTERN) – Advanced Front-End Development
Deadline: 48 Hours from Assignment Receipt
Submission: Provide a GitHub repository link with clear instructions on how to run the project. Optionally, include a live demo link.

About PGAGI

PGAGI is an AI-based startup committed to delivering cutting-edge solutions in a fast-paced development environment. We value creativity, efficiency, and the ability to deliver high-quality user experiences. This advanced assignment is designed to assess your proficiency in modern front-end technologies, your ability to architect complex applications, and your skill in creatistandardsng dynamic, user-friendly interfaces with high performance and accessibility .

Assignment Overview

You are tasked with developing a "Comprehensive Analytics Dashboard" that displays data fetched from multiple APIs, incorporates interactive and complex animations, and offers a seamless, high-performance user experience. The application should be built using Next.js, React, TypeScript, SCSS/CSS, and Tailwind CSS. Additionally, implement advanced animations using 3.js, Lottie, or advanced CSS animations. The project should demonstrate best practices in state management, performance optimization, accessibility, and testing.

Detailed Requirements

1. Project Setup

- **Framework & Language:**
  - **Use Next.js with React and TypeScript.**
  - **Configure TypeScript with strict type checking (`strict: true` in `tsconfig.json`) to enforce type safety.**
  - **Initialize the project with a custom Next.js configuration to support advanced features like absolute imports and module path aliases.**
- **Styling:**
  - **Utilize SCSS/CSS and Tailwind CSS for styling components.**
  - **Implement CSS Modules for component-scoped styling.**

    - Customize the Tailwind CSS configuration to include:
        - Custom color palettes.
        - Extended spacing scales.
        - Additional breakpoints for responsiveness.
        - Custom utilities for specific design needs.
- Project Structure:
    - Organize the project with a scalable folder structure:
        - `components/` – Reusable UI components.
        - `pages/` – Next.js pages with dynamic routing.
        - `styles/` – Global styles and Tailwind configurations.
        - `hooks/` – Custom React hooks.
        - `store/` – State management setup (e.g., Redux slices).
        - `utils/` – Utility functions and helpers.
        - `services/` – API service integrations.
        - `public/` – Static assets.
    - Implement code splitting and dynamic imports for performance optimization.
    - Set up absolute imports and path aliases to simplify import statements.
- State Management:
    - Use Redux Toolkit for complex state management.
    - Implement RTK Query for data fetching and caching.
    - Structure the Redux store with slices for different data domains (e.g., weather, news, finance).

2. API Integrations

Integrate and fetch data from the following three public APIs:

1. Weather API:
    - Service: OpenWeatherMap API
    - Features:
        - Display current weather information for a user-specified location.
        - Show a 7-day weather forecast with temperature trends, humidity, wind speed, and weather conditions.
        - Implement geolocation to automatically fetch weather data based on the user's current location.
    - Implementation Details:
        - Create a search bar with autocomplete suggestions for city names using GeoDB Cities API.
        - Display weather data using interactive charts (e.g., temperature over the next 7 days) with Recharts or D3.js.
2. News API:
    - Service: NewsAPI
    - Features:

        - **Display the latest news headlines categorized into sections like Technology, Sports, Business, Health, and Entertainment.**
        - **Allow users to filter news based on selected categories.**
        - **Implement pagination or infinite scrolling for news articles.**
    - **Implementation Details:**
        - **Create a news feed with article cards displaying headlines, images, summaries, and source information.**
        - **Implement a detail view modal for each article with full content and external links.**
3. **Finance API:**
    - **Service: Alpha Vantage API**
    - **Features:**
        - **Display real-time stock market data for user-selected stock symbols.**
        - **Show interactive stock charts (e.g., line charts for price over time, candlestick charts for trading activity).**
        - **Include historical data analysis with options to view data over different time ranges (1 day, 1 week, 1 month, 1 year).**
    - **Implementation Details:**
        - **Implement a search functionality with autocomplete for stock symbols.**
        - **Display key metrics such as current price, daily high/low, volume, and percentage change.**
        - **Use Chart.js or Victory for rendering stock charts with interactivity (zoom, hover details).**

Bonus:

- **Third API Integration:**
    - **Choose an additional API that complements the dashboard, such as:**
        - **GitHub API: Display repository statistics, commit history, and contributor information.**
        - **TMDB API: Show movie data, trending films, and user ratings.**
        - **Spotify API: Present user listening statistics, top tracks, and playlists.**

Note:

- **If API keys are required, include instructions on how to set them up in your README.**
- **For evaluation purposes, mock data can be used if necessary, but dynamic data fetching is preferred.**

3. User Interface & Experience

- **Layout:**

    - **Design a highly responsive and intuitive dashboard layout that adapts seamlessly to various screen sizes (mobile, tablet, desktop).**
    - **Implement a sticky sidebar with navigation links to different sections of the dashboard (Weather, News, Finance, etc.).**
    - **Include a header with:**
        - **Search Functionality: Global search bar for quick access to different sections or functionalities.**
        - **User Profile Access: Avatar with dropdown menu for profile settings, notifications, and logout.**
        - **Theme Toggling: Toggle switch for dark and light modes with smooth transitions.**
- **Components:**
    - **Reusable and Modular Components:**
        - **Card Components: For displaying summarized data (e.g., weather summary, news headlines, stock information).**
        - **Chart Components: Interactive charts with tooltips, legends, and dynamic data binding.**
        - **Tables: Paginated and sortable tables for detailed data views.**
        - **Modals: For detailed views, forms, and additional information.**
        - **Dropdowns: For category selection, filter options, and settings.**
        - **Buttons and Icons: Consistent styling with hover and active states.**
    - **Drag-and-Drop Functionality:**
        - **Implement widget customization allowing users to rearrange dashboard components.**
        - **Use React DnD or Framer Motion to enable drag-and-drop with smooth animations and state persistence.**
    - **Notifications:**
        - **Implement a notification system to alert users of important updates or actions (e.g., new data available, errors).**
- **Tailwind CSS:**
    - **Leverage Tailwind CSS utility classes for rapid styling while maintaining consistency.**
    - **Customize the Tailwind configuration to include:**
        - **Custom Themes: Define primary, secondary, and accent colors.**
        - **Extended Spacing: Add custom spacing values for unique design needs.**
        - **Responsive Breakpoints: Ensure components look good on all devices.**
        - **Typography: Define custom font families, sizes, and weights for headings, body text, and captions.**
- **Accessibility:**
    - **Ensure the application adheres to WCAG 2.1 accessibility standards.**
    - **Implement proper ARIA attributes for interactive elements.**
    - **Ensure keyboard navigation is fully supported across the application.**

- **Implement focus management to guide users through interactive components.**
- **Ensure sufficient color contrast and provide alternative text for images.**

4. Advanced Animations

- **Interactive Elements:**
  - **Incorporate advanced animations to enhance user interaction, including:**
    1. **Smooth Transitions: Between different sections and states.**
    2. **Animated Charts: Charts that animate data changes smoothly.**
    3. **Loading Spinners: Creative and engaging loading indicators.**
    4. **Hover Effects: Subtle animations on buttons, cards, and interactive elements.**
- **Data Visualization:**
  - **Use 3.js or Lottie to create at least two engaging animations:**
    1. **Animated Data Charts:**
       - **Implement interactive charts (e.g., bar charts, line graphs) that animate on data load and user interactions such as hover, click, and filter.**
       - **Add dynamic data binding where charts update in real-time based on API data.**
    2. **Dynamic Backgrounds:**
       - **Create animated backgrounds that respond to user actions or data changes, such as:**
         - **Weather-Based Animations: Rain, snow, or sunshine effects based on current weather data.**
         - **Finance-Based Animations: Subtle animations indicating stock market trends (e.g., rising lines, fluctuating bars).**
- **Performance:**
  - **Ensure animations are optimized for performance and do not hinder the application's responsiveness.**
  - **Utilize requestAnimationFrame, CSS transforms, and hardware acceleration where appropriate.**
  - **Avoid excessive use of heavy animation libraries that could bloat the bundle size.**

5. Functionality

- **Dynamic Data Fetching:**
  - **Implement efficient API calls using React Query for data fetching, caching, and synchronization.**
  - **Utilize SSR (Server-Side Rendering) and SSG (Static Site Generation) features of Next.js where appropriate for performance gains.**
  - **Implement data prefetching for smoother user experience.**

- **State Management:**
    - **Manage complex application state using Redux Toolkit with RTK Query for data fetching.**
    - **Implement selectors and memoization using Reselect to optimize state access and component re-renders.**
    - **Ensure state persistence across sessions using Redux Persist or similar libraries.**
- **Error Handling:**
    - **Implement comprehensive error handling for API failures, network issues, and unexpected errors.**
    - **Display user-friendly error messages with options to retry or navigate to safe sections.**
    - **Log errors to an external service like Sentry for monitoring (optional).**
- **Loading States:**
    - **Show sophisticated loading indicators, skeleton screens, or shimmer effects while data is being fetched.**
    - **Implement progressive loading for components to enhance perceived performance.**
- **Search & Filtering:**
    - **Implement advanced search functionality with debouncing and auto-suggestions.**
    - **Allow users to filter and sort data across different sections of the dashboard.**
    - **Implement multi-criteria filtering (e.g., filter news by category and date).**
- **Dark Mode:**
    - **Implement a seamless toggle for dark and light themes using Tailwind CSS.**
    - **Ensure all components and animations adapt appropriately to theme changes.**
    - **Persist user's theme preference using localStorage or cookies.**

6. Advanced Features

- **User Authentication:**
    - **Implement a secure authentication system allowing users to sign up, log in, and manage their profiles using NextAuth.js.**
    - **Support OAuth providers (e.g., Google, GitHub) and email/password authentication.**
    - **Protect sensitive routes and data, ensuring only authenticated users can access their personalized dashboards.**
- **Real-Time Data Updates:**
    - **Integrate WebSockets or Server-Sent Events (SSE) to display real-time updates such as:**
        - **Live stock prices.**
        - **Breaking news alerts.**
        - **Real-time weather updates.**

    - **Implement real-time notifications and in-app alerts for significant events.**
- **Localization:**
    - **Implement multi-language support using react-i18next.**
    - **Allow users to switch between at least two languages (e.g., English and Spanish).**
    - **Ensure all static and dynamic text is translatable, including date and number formats.**
- **Performance Optimization:**
    - **Optimize the application for performance using techniques like:**
        - **Code Splitting: Break down the application into smaller chunks for faster load times.**
        - **Tree Shaking: Remove unused code during the build process.**
        - **Image Optimization: Use Next.js Image component for optimized image loading.**
        - **Lazy Loading: Defer loading of non-critical resources until they are needed.**
    - **Analyze and improve the application's performance using tools like Lighthouse, Webpack Bundle Analyzer, and React Profiler.**
    - **Implement Caching Strategies for API data to reduce redundant network requests.**
- **Testing:**
    - **Unit Tests:**
        - **Write comprehensive unit tests for all components using Jest and React Testing Library.**
        - **Mock API calls and test component rendering based on different states (loading, success, error).**
    - **Integration Tests:**
        - **Test interactions between components and state management using React Testing Library.**
        - **Ensure that data flows correctly through the application.**
    - **End-to-End Tests:**
        - **Write end-to-end tests using Cypress or Playwright to validate critical user flows such as:**
            - **User authentication (sign up, log in, log out).**
            - **Data fetching and display for each API integration.**
            - **Widget customization and drag-and-drop functionality.**
        - **Ensure tests cover various scenarios, including edge cases and error conditions.**
    - **Test Coverage:**
        - **Achieve at least 80% test coverage across the codebase.**
        - **Include coverage reports in the project.**

7. Code Quality & Best Practices

- **TypeScript:**

    - **Utilize TypeScript effectively for type safety, leveraging interfaces, types, generics, and enums.**
    - **Avoid using the `any` type; ensure strict type definitions across the application.**
    - **Use TypeScript's advanced features like Union Types, Intersection Types, and Type Guards where appropriate.**
- **Linting & Formatting:**
    - **Ensure the codebase is free of linting errors by configuring ESLint with appropriate rules.**
    - **Use Prettier for consistent code formatting, integrated with ESLint.**
    - **Set up husky and lint-staged to enforce linting and formatting on pre-commit hooks.**
- **Documentation:**
    - **Write clear and concise comments where necessary.**
    - **Document components, functions, and modules using JSDoc or similar conventions for enhanced readability.**
    - **Create a Component Library Documentation detailing the usage, props, and examples of reusable components.**
- **Version Control:**
    - **Use meaningful commit messages following the Conventional Commits specification (e.g., `feat: add weather API integration`).**
    - **Maintain a clean Git history with well-structured branches and pull requests.**
    - **Implement branching strategies like Git Flow or GitHub Flow to manage feature development and releases.**
- **Security Best Practices:**
    - **Securely manage environment variables for API keys and sensitive data using `.env` files and Next.js environment configuration.**
    - **Avoid exposing sensitive information in the client-side code.**
    - **Implement input validation and sanitize user inputs to prevent security vulnerabilities like XSS or injection attacks.**

8. Deployment

- **Continuous Deployment:**
    - **Set up CI/CD pipelines using GitHub Actions to automate testing, linting, and deployment processes.**
    - **Configure automated deployments to platforms like Vercel or Netlify upon successful builds and tests.**
- **Optimized Build:**
    - **Ensure the production build is optimized for performance, security, and scalability.**
    - **Use Next.js optimizations like Image Optimization, Incremental Static Regeneration (ISR), and API Routes for serverless functions.**

- **Environment Variables:**
  - **Securely manage environment variables for API keys and sensitive data using `.env` files and Next.js environment configuration.**
  - **Ensure environment variables are not exposed in the client-side bundle.**
- **Monitoring & Analytics:**
  - **Integrate monitoring tools like Google Analytics, Sentry, or LogRocket to track user interactions, performance metrics, and error reporting.**
  - **Set up dashboards to monitor application health and usage statistics.**

Evaluation Criteria

Your submission will be evaluated based on the following:

1. **Functionality: Completeness and correctness of the required and advanced features.**
2. **Code Quality: Cleanliness, organization, adherence to best practices, and effective use of TypeScript.**
3. **UI/UX Design: Aesthetic appeal, responsiveness, accessibility, and overall user experience.**
4. **Animations: Creativity, complexity, and performance of animations integrated into the application.**
5. **Technical Proficiency: Effective use of Next.js, React, TypeScript, SCSS/CSS, Tailwind CSS, and state management libraries.**
6. **Performance Optimization: Implementation of performance best practices and optimizations.**
7. **Testing: Coverage and quality of unit, integration, and end-to-end tests.**
8. **Documentation: Clarity and comprehensiveness of the README file and in-code documentation.**
9. **Innovation (Bonus): Any additional features, creative solutions, or exceptional implementations beyond the requirements.**
10. **Security: Implementation of security best practices to protect user data and application integrity.**

Submission Guidelines

1. **Repository:**
   - **Push your code to a public GitHub repository. Ensure it's accessible and includes all necessary files.**
   - **Use a clear and descriptive repository name (e.g., `pgagi-analytics-dashboard`).**

2. **README:**
    - **Include a comprehensive README file that covers:**
        - **Project Overview: Brief description of the application and its features.**
        - **Technologies Used: List of technologies, libraries, and tools utilized.**
        - **Installation Instructions: Step-by-step guide to set up the project locally, including how to configure environment variables.**
        - **How to Run the Project: Instructions to start the development server and build the production version.**
        - **Testing Instructions: How to run tests and view coverage reports.**
        - **Deployment Details: If deployed, include the live application link and any deployment notes.**
        - **Environment Variables: Detail the required environment variables and their purposes.**
        - **API Setup: Instructions on obtaining and configuring API keys for the integrated services.**
        - **Additional Notes or Features: Highlight any additional implementations or noteworthy aspects of your project.**
        - **Screenshots: Include screenshots or GIFs showcasing different parts of the application.**
3. **Live Demo (Optional):**
    - **If deployed, include the live application link in the README.**
    - **Ensure the deployed version is fully functional and mirrors the GitHub repository.**
    - **Provide instructions on how to access authenticated sections if authentication is implemented.**
4. **Deadline:**
    - **Ensure your submission is made within 48 hours from receiving this assignment.**

Additional Resources

- **Next.js Documentation: https://nextjs.org/docs**
- **React Documentation: https://reactjs.org/docs/getting-started.html**
- **TypeScript Documentation: https://www.typescriptlang.org/docs/**
- **Tailwind CSS Documentation: https://tailwindcss.com/docs**
- **3.js Documentation: https://threejs.org/docs/**
- **Lottie Documentation: https://airbnb.io/lottie/**
- **Redux Toolkit Documentation: https://redux-toolkit.js.org/**
- **React Testing Library: https://testing-library.com/docs/react-testing-library/intro/**
- **Jest Documentation: https://jestjs.io/docs/getting-started**
- **Cypress Documentation: https://www.cypress.io/**

- **Accessibility Guidelines (WCAG 2.1): https://www.w3.org/TR/WCAG21/**
- **NextAuth.js Documentation: https://next-auth.js.org/getting-started/introduction**
- **React DnD Documentation: https://react-dnd.github.io/react-dnd/about**
- **Framer Motion Documentation: https://www.framer.com/motion/**
- **React Query Documentation: https://react-query.tanstack.com/**
- **Recharts Documentation: https://recharts.org/en-US/**
- **D3.js Documentation: https://d3js.org/**
- **Chart.js Documentation: https://www.chartjs.org/docs/latest/**
- **Victory Documentation: https://formidable.com/open-source/victory/docs/**
- **React-i18next Documentation: https://react.i18next.com/**
- **Sentry Documentation: https://docs.sentry.io/platforms/javascript/guides/react/**
- **LogRocket Documentation: https://docs.logrocket.com/docs**
- **GitHub Actions Documentation: https://docs.github.com/en/actions**
- **Vercel Documentation: https://vercel.com/docs**
- **Netlify Documentation: https://docs.netlify.com/**
- **Husky Documentation: https://typicode.github.io/husky/#/**
- **Lint-Staged Documentation: https://github.com/okonet/lint-staged**

Good Luck!

We are excited to see your innovative solutions, creative designs, and technical prowess. This assignment is an opportunity to showcase your ability to handle complex projects, write clean and maintainable code, and deliver a polished, high-quality application. If you have any questions or need clarifications, feel free to reach out within the assignment timeframe.

PGAGI Recruitment Team