Name- Rutik Maraskolhe.

Summary of App Assignment

The app I've been working on is a sleek iOS application that combines SwiftUI for its interface, MVVM architecture for its design pattern, and URLSession for seamless data fetching from APIs. Let me break down the key features and components for you:

1. **SwiftUI**: We've utilized SwiftUI to craft a modern and responsive user interface, ensuring a smooth and engaging user experience throughout the app.
2. **MVVM Architecture**: Following the Model-View-ViewModel (MVVM) architecture, our codebase is structured in a way that separates concerns and promotes maintainability. This architecture allows for better testability and scalability of the codebase.
3. **URLSession**: We've leveraged URLSession for handling network requests, enabling the app to seamlessly fetch data from the API endpoints. This ensures efficient communication between the app and the server, enhancing overall performance.
4. **Custom Font Manager**: To enhance the visual appeal of the app, we've integrated a custom Font Manager. We've incorporated the 'Figtree' font downloaded from Google Fonts, ensuring a unique and consistent typography throughout the app.
5. **Custom Greeting Message View**: We've implemented a personalized touch by creating a custom greeting message view. Depending on the user's current time, the app dynamically generates a greeting message, fostering a warm and welcoming user experience.
6. **Custom Tab Bar Button View**: Our app features a custom tab bar button view designed meticulously to align with the provided UI specifications. This ensures consistency and harmony within the app's navigation structure.
7. **ViewModel - DashboardViewModel Class**: We've developed a robust ViewModel class, namely DashboardViewModel, tailored for iOS apps. This class harnesses the power of Combine for reactive programming. It effectively manages dashboard data and states, featuring published properties such as dashboardResponse, isLoading, and error. The fetchDashboardData() method orchestrates asynchronous data fetching from the API, handling HTTP requests and response data with precision. Combine operators facilitate seamless data handling and error management, ensuring a smooth user experience. The cancellables set efficiently manages Combine publishers for proper cleanup.
8. **Custom Model**: We've created a custom model based on the JSON data received from the API endpoint. This model ensures structured data representation within the app, facilitating seamless integration and manipulation of data.
9. **Graph Visualization with Dummy Data**: In instances where the API response lacks data related to the graph, we've implemented a solution by generating dummy data for graph visualization within the app. This ensures that users can still interact with and explore graph representations, enhancing the app's functionality and user engagement.