Team Number: 21
Team Members: Matthew McManness, Shravya Matta, Manvir Kaur, Mariam Oraby, and Magaly Camacho
Project Name: BusyBee

Project Synopsis:

A mobile-friendly calendar and to-do list app (BusyBee) built with Kivy and KivyMD, enabling users to efficiently manage schedules and tasks with a modern, intuitive interface.

Architecture:

Overview:

The BusyBee App is built using Kivy and KivyMD, two Python frameworks designed for building user-friendly interfaces across multiple platforms, particularly for mobile devices. The app is designed to offer an intuitive experience for managing schedules and tasks through two key components: the Calendar View and the To-Do List View. Both views are accessible via a bottom navigation bar that ensures quick, seamless navigation.

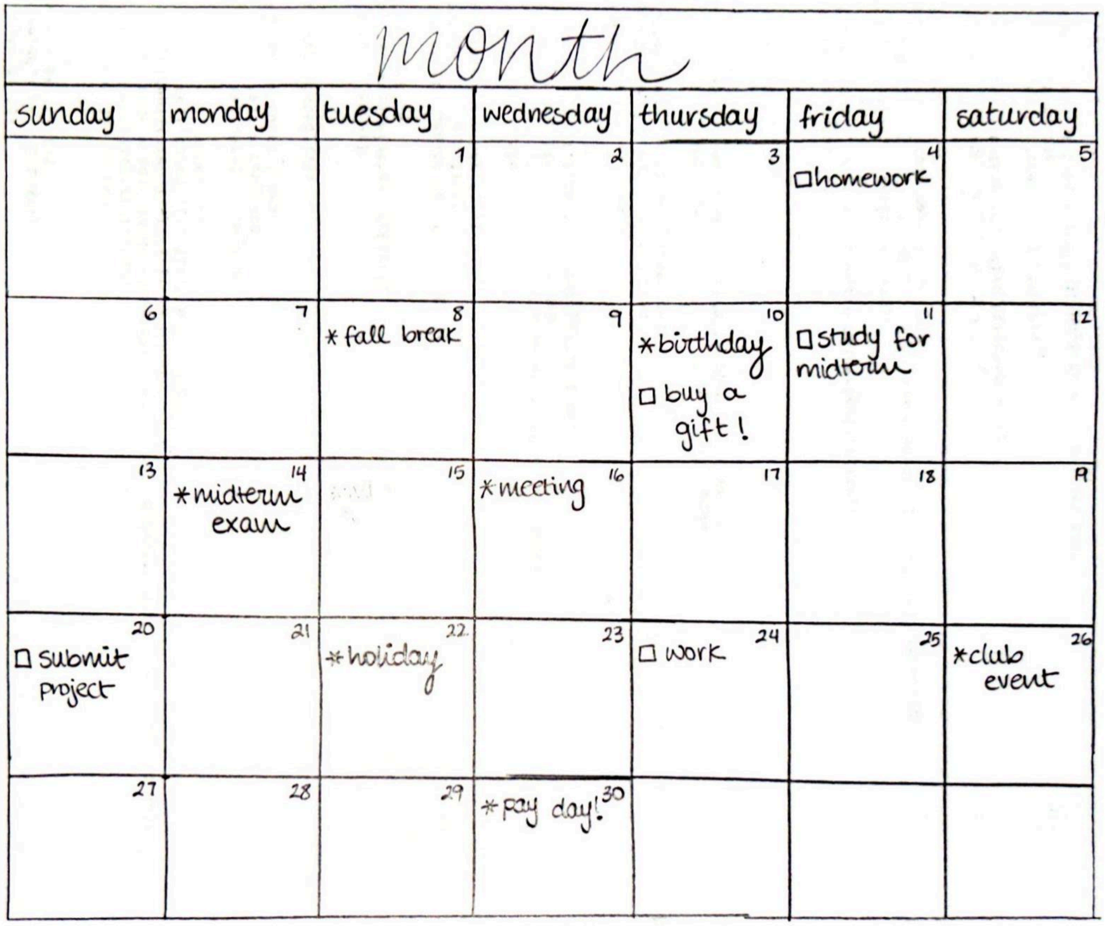
This architecture focuses on responsiveness and scalability by using Kivy's dynamic layout features, making the app adaptable to various screen sizes. KivyMD's material design components provide a modern look, aligning with current UI trends used in popular productivity tools. Below, we break down the components and workflows involved in the app's architecture.

Core Components:

1. App Entry Point (KivyMD App):
    - The MDApp class serves as the entry point for the application. It initializes the UI and controls the overall app life cycle, including launching the main window, managing user interactions, and handling the screen transitions.
    - This class contains the logic that connects the navigation bar buttons with the appropriate screens. When a user clicks the "Calendar" or "To-Do List" button, it triggers a function to switch views via the ScreenManager.
    - The build() method inside the MDApp class loads the KivyMD interface, setting up the layouts, and ensuring everything is initialized properly.

2. Screen Manager:
    - The ScreenManager plays a critical role in the app by managing the transitions between the Calendar Screen and To-Do List Screen.
    - This structure allows the app to simulate a tabbed view, where each tab is represented by a separate screen. Users can easily switch between these screens using the bottom navigation bar.
    - The NoTransition class ensures that screen transitions happen instantly without animations, making it suitable for fast interactions on mobile devices.

3. Calendar Screen:



month						
sunday	monday	tuesday	wednesday	thursday	friday	saturday
		1	2	3	4 □homework	5
6	7	8 * fall break	9	10 *birthday □ buy a gift!	11 □study for midterm	12
13	14 *midterm exam	15	16 *meeting	17	18	19
20 □ submit project	21	22 *holiday	23	24 □ work	25	26 *club event
27	28	29	30 *pay day!			

- The Calendar Screen is one of the primary views of the app, displaying a 7x5 grid layout representing the days of the month. The grid uses Python's built-in calendar library to dynamically generate the correct dates based on the current month and year.
- Each day is represented by a button, allowing users to interact with it. This interaction can later be expanded to include features such as adding events or reminders.
- The screen includes a floating action button (FAB) at the bottom-right corner, which can trigger actions such as adding new calendar events. This design mimics modern calendar apps, where quick actions are accessible via a floating button.
- Additionally, a header at the top of the screen displays the current month and year, giving users context for the calendar they are viewing.

4. To-Do List Screen:
   - The To-Do List Screen provides a simple, scrollable list where users can manage their tasks. The task list is displayed using the MDList component, which is a specialized KivyMD widget designed for lists with material design aesthetics.

- Users can interact with the tasks through checkboxes to mark them as complete, or by adding new items via an Add Task option, which can later be implemented as a dialog or form.
- The tasks are wrapped in a ScrollView, allowing the app to handle long lists gracefully. This ensures the layout remains responsive, even when there are many tasks to display.

5. Navigation Bar (Bottom):
    - The bottom navigation bar provides an easy way to switch between the calendar and to-do list views. This navigation bar contains two buttons: one for the Calendar View and one for the To-Do List View.
    - The use of a bottom navigation bar follows modern design trends, as it is easier to access on mobile devices compared to a top or side navigation bar. The buttons are styled using KivyMD's material design components to maintain a sleek and modern appearance.

6. Database (SQLite):
    - The database will allow the persistent storage and view of events and tasks, as well as their details, across different launches of the application.
    - The application will utilize a data structure class to model the records in order to more smoothly manage and display events and tasks.
    - For more specifics on the structure of both the database and data structure please see the Requirements Artifacts.

How the Software Works:

The app follows a simple, user-driven flow, where the user interacts with the UI elements to navigate between screens or manage tasks and events. Below is a breakdown of the interaction flow and data flow within the app.

User Interaction Flow:

1. App Launch:
    - When the app is launched, the MDApp initializes the ScreenManager and loads the default screen (either the Calendar or To-Do List, depending on the last session).

2. Switching Screens:
    - The user interacts with the bottom navigation bar to switch between the calendar and to-do list views.
    - The switch_to_screen() method in the MDApp class handles this interaction by updating the current screen in the ScreenManager.

3. Calendar Interaction:

- On the Calendar Screen, users can view the current month and interact with individual days by tapping on the buttons. The app dynamically adjusts the calendar to reflect the correct number of days in each month.

4. Task Management:
   - On the To-Do List Screen, users can scroll through tasks, mark them as complete using checkboxes, and add new tasks (this can be expanded with dialog inputs).

Data Flow Diagram:

```
[App Entry Point (MDApp)]
      |
      v
[Screen Manager] <--- [Navigation Bar]
   /                    \
[Calendar]            [To-Do List]
   |                           |
[Calendar Grid]          [Task List]
```

Explanation of Data Flow:

1. App Initialization:
   - The MDApp loads the user interface and sets the ScreenManager as the root widget. It initializes the first screen, which can either be the Calendar or To-Do List.

2. Screen Navigation:
   - The bottom navigation bar sends input to the switch_to_screen() function. This function updates the current screen in the ScreenManager, ensuring the correct view is displayed.

3. Calendar Data Handling:
   - The Calendar Screen uses the datetime and calendar modules to generate the correct dates and display them in a 7x5 grid layout. The calendar grid is populated dynamically, ensuring that it always matches the current month and year.

4. Task Management:
   - The To-Do List Screen maintains a scrollable list of tasks. Each task is displayed as a OneLineListItem, which is part of KivyMD’s material design components.

Component Relationships:

1. Screen Manager ↔ Calendar and To-Do List Screens:
   - The Screen Manager switches between the two views when the user interacts with the bottom navigation bar.

2. FAB ↔ Calendar Screen:

    - On the Calendar Screen, the Floating Action Button (FAB) can be used to trigger event creation or perform specific actions.

3. Task Management ↔ To-Do List:
    - The tasks in the To-Do List are displayed using an MDList component, and users can interact with each task (e.g., add, edit, or delete tasks).

User Interaction Flow Diagram:

User → Bottom Navigation → Screen Manager → Calendar/To-Do List View

Data Flow Between Components Diagram:

MDApp → ScreenManager → (Calendar/To-Do List Screens) → FAB/Task Actions

Summary:

The BusyBee App combines the strengths of Kivy and KivyMD to provide a modern, responsive, and mobile-friendly user experience. Its architecture, centered around the ScreenManager, allows for seamless transitions between the calendar and to-do list views. The use of material design components ensures a sleek and intuitive interface. The bottom navigation bar enhances usability, making it easy for users to switch between views on a mobile device.

With its modular design, the app is ready for future expansion, allowing developers to add advanced features like notifications, cloud synchronization, and persistent storage. This architecture ensures that the app is scalable, maintainable, and adaptable for both personal and professional use cases.