The key entities in this document are:

1. **Pro Calendar** - The main application being described.
2. **Users** - The individuals who will interact with the application (e.g., creating accounts, managing events).
3. **Database** - Used to store user information and event data.
4. **Events** - The core functionality of the app involves creating and managing events.
5. **Google Account** - A potential method for user identification and integration.
6. **Account Creation** - The process by which users create their accounts in the app.
7. **Reminders** - Notifications related to events created in the calendar.
8. **User Interface (UI)** - The visual design and interaction flow of the application.
9. **Functionality Requirements** - Features like event creation, reminders, and customization options for users.
10. **APIs** - Kivy (for app development) and SQLite3 (for database management).
11. **Test Cases** - Scenarios to test the application's performance and functionality.
12. **Verification and Validation Testing** - Methods for ensuring the application functions as expected.

#### **1.1: Application will store user information in a database (Must Have)**

The database will be used to store user-related data, including the user’s email address, username, password, and any other personal information required for account management. The data will be securely stored and used for identification, authentication, and event management.

#### **1.2: Application will identify users by unique e-mail address or username (Must Have)**

Each user must have a unique identifier, either an email address or a username. This will allow the application to distinguish between users and ensure that each user has access to their own account, events, and settings.

#### **1.3: Application will identify users by Google login (Must Have)**

For convenience, users will have the option to log in using their Google account. This will streamline the login process, and the user's Google credentials will be used to authenticate their identity within the application.

### **Explanation of Database Relationships**

1. **Users and Accounts**
   * **One-to-Many Relationship**: A user can have one account, but each account is linked to a unique user. The **users** table will store general user information like email, username, and login credentials, while each **account** can contain event data, notifications, and customized calendar settings.
   * **Attributes**: User ID, email, username, password hash, Google login status, and date of creation.
2. **Users and Events**
   * **One-to-Many Relationship**: A user can create multiple events, but each event is linked to a specific user. This ensures that the calendar is personalized to the user’s needs, and each event is linked to their account.
   * **Attributes**: Event ID, event name, description, date, time, user ID (foreign key), reminder preferences.
3. **Events and Reminders**
   * **One-to-One Relationship**: Each event can have one reminder associated with it. This ensures that the user is notified before the event occurs. A reminder will contain the time and the notification method (email or push notification).
   * **Attributes**: Reminder ID, event ID (foreign key), reminder time, reminder method.
4. **Users and Customization Settings**
   * **One-to-Many Relationship**: Each user can have multiple customization settings for their calendar (such as color theme, layout preferences, etc.). These settings are tied to the user's account and are personalized.
   * **Attributes**: Setting ID, user ID (foreign key), color scheme, layout preferences, display options (e.g., number of days shown).

### 

### **Data Flow and Integrity**

* **User Registration/Login**: When a new user registers, their information (email, username, password) is stored in the **users** table. If they choose to log in via Google, the application will authenticate using Google's API, and the user's Google credentials will be linked to the **users** table.
* **Event Creation**: When a user creates an event, an entry is added to the **events** table with a reference to the user’s ID. This way, each event is linked to a specific user’s account.
* **Reminder Creation**: If a user sets a reminder for an event, a corresponding entry is made in the **reminders** table, referencing the event ID. This reminder is then linked to the user’s event.
* **Customization**: Each user's individual calendar settings (like theme or layout) are stored in the **customization settings** table, which is referenced by the user’s ID. This allows the calendar to adjust its appearance based on the user’s preferences.

### **Conclusion**

The Pro Calendar application relies on a relational database to store user information, events, reminders, and customization settings. These entities are interconnected through foreign keys, ensuring data integrity and a personalized user experience. By organizing the data in this manner, the application can efficiently manage user-specific data and provide seamless interaction across different features, such as event management, reminders, and customization.