Model-View-Controller (MVC) Breakdown

1. Model

The Model represents the data and business logic of the application. For Grid Social, this includes user data, group information, calendars, events, and scrapbooks.

Data Models:

- **User**:
  - Attributes: `userID`, `username`, `email`, `password`, `profileInfo`, `calendarPreferences`
  - Methods: `createUser()`, `modifyUser()`, `deleteUser()`
- **Group**:
  - Attributes: `groupID`, `groupName`, `groupDescription`, `members[]`, `events[]`
  - Methods: `createGroup()`, `addMember()`, `removeMember()`, `deleteGroup()`
- **Event**:
  - Attributes: `eventID`, `eventName`, `eventDescription`, `date`, `time`, `location`, `groupID`, `attendees[]`
  - Methods: `createEvent()`, `modifyEvent()`, `deleteEvent()`, `addAttendee()`, `removeAttendee()`
- **Calendar**:
  - Attributes: `calendarID`, `userID`, `events[]`, `public`
  - Methods: `addEvent()`, `removeEvent()`, `updateEvent()`
- **Scrapbook**:
  - Attributes: `scrapbookID`, `eventID`, `photos[]`
  - Methods: `addPhoto()`, `removePhoto()`, `viewPhotos()`

2. View

The View is responsible for displaying the data provided by the Model. For Grid Social, this includes user interfaces for feeds, profiles, calendars, groups, and scrapbooks.

User Interfaces:

- **Login View**: Interface for user login.
- **Sign-up View**: Interface for user registration.
- **Feed View**:
  - Public Feed: Displays events and updates from all users.
  - Personal Feed: Displays events and updates from followed users and groups.
- **Profile View**: Interface for viewing and editing user profile.
- **Calendar View**: Interface for viewing and managing personal and group calendars.
- **Group View**: Interface for creating, managing, and joining groups.
- **Scrapbook View**: Interface for viewing and adding photos to scrapbooks.

3. Controller

The Controller handles the user input and interacts with the Model to update the View accordingly. For Grid Social, this includes handling requests for user actions like login, sign-up, event creation, group management, and more.

Controllers:

- **AuthController**:
  - Methods: `login()`, `logout()`, `signup()`, `forgotPassword()`, `resetPassword()`
- **UserController**:
  - Methods: `viewProfile()`, `editProfile()`, `followUser()`, `unfollowUser()`, `viewCalendar()`, `addEventToCalendar()`, `removeEventFromCalendar()`
- **FeedController**:
  - Methods: `viewPublicFeed()`, `viewPersonalFeed()`, `refreshFeed()`
- **GroupController**:
  - Methods: `createGroup()`, `joinGroup()`, `leaveGroup()`, `inviteToGroup()`, `removeFromGroup()`, `createEventInGroup()`, `viewGroupCalendar()`
- **EventController**:
  - Methods: `createEvent()`, `modifyEvent()`, `deleteEvent()`, `viewEventDetails()`, `addAttendee()`, `removeAttendee()`
- **ScrapbookController**:
  - Methods: `addPhotoToScrapbook()`, `removePhotoFromScrapbook()`, `viewScrapbook()`

Example Flow: Creating a Group Event

1. **User Input (View)**:
   - User navigates to the group view and fills out a form to create a new event.
2. **Controller Action**:
   - The `GroupController` receives the input via `createEventInGroup()` method.
   - The controller validates the input data.
3. **Model Update**:
   - If valid, the controller calls `Event.createEvent()` method on the `Event` model.
   - The event is saved in the database and associated with the specific group.
4. **View Update**:
   - The `GroupController` then updates the group calendar view to display the new event.
   - The updated group calendar is rendered and presented to the user.

Summary

This MVC model ensures a clear separation of concerns, making the Grid Social application scalable, maintainable, and efficient. The provided documents give a detailed breakdown of the requirements and design, which we have translated into a structured MVC architecture.