WEEK 4

1. Class Diagram with Applied Design Patterns

The class diagram illustrates the core classes, their attributes, methods, and relationships, with the Observer Pattern, Factory Pattern, and Adapter Pattern explicitly applied to support the app's functionality. The diagram focuses on key entities (e.g., users, posts, groups, events, marketplace items) and their interactions.

Design Patterns Applied:

- **Observer Pattern**: Implemented via methods like observeUpdates() and observe() in classes (User, StudyGroup, Event, ChatMessage, Notification) to listen for real-time Firestore updates (e.g., new messages or events).
- **Factory Pattern**: Represented by the PostFactory class, which creates different types of Post objects (e.g., text, image, event) based on input.
- **Adapter Pattern**: Embodied in the FirestoreAdapter class, which converts Firestore documents to app-specific models (e.g., Post, Event) and vice versa.

---

2. Explanation of Responsibilities per Class

Each class has distinct responsibilities to support *CampusConnect*'s functionality, aligned with the app's features (Home Feed, Event Calendar, Group Chats, Marketplace, Profile).

- **User**
    - **Responsibilities**: Manages user authentication, profile data, and real-time updates.
    - **Attributes**: id (unique identifier), email (university email), name, profilePic.
    - **Methods**:
        - login(): Authenticates with Firebase Authentication.
        - updateProfile(): Updates user details in Firestore.

        - observeUpdates(): Subscribes to profile changes (Observer Pattern).
    - **Role**: Represents a student, central to all app interactions.
- **Post**
    - **Responsibilities**: Handles creation, updating, and deletion of feed posts.
    - **Attributes**: id, content, type (e.g., text, image), timestamp.
    - **Methods**:
        - create(): Saves a post to Firestore.
        - update(): Modifies post content.
        - delete(): Removes a post.
    - **Role**: Represents content in the Home Feed, created via the Factory Pattern.
- **StudyGroup**
    - **Responsibilities**: Manages study group creation, membership, and messaging.
    - **Attributes**: id, name, course, members (list of users).
    - **Methods**:
        - addMember(): Adds a user to the group.
        - sendMessage(): Sends a chat message.
        - observeMessages(): Listens for new messages (Observer Pattern).
    - **Role**: Facilitates academic collaboration in group chats.
- **Event**
    - **Responsibilities**: Manages campus events and deadlines in the Event Calendar.
    - **Attributes**: id, title, date, location.
    - **Methods**:
        - create(): Saves an event to Firestore.
        - observeUpdates(): Monitors event changes (Observer Pattern).
    - **Role**: Supports event tracking and notifications.
- **MarketplaceItem**

    - **Responsibilities**: Handles listing, updating, and browsing items for sale.
    - **Attributes**: id, title, price, seller (User reference).
    - **Methods**:
        - listItem(): Publishes an item to the Marketplace.
        - updateItem(): Modifies item details.
    - **Role**: Enables buying and selling within the student community.
- **ChatMessage**
    - **Responsibilities**: Manages group chat messages.
    - **Attributes**: id, content, sender (User), timestamp.
    - **Methods**:
        - send(): Saves a message to Firestore.
        - observe(): Listens for new messages (Observer Pattern).
    - **Role**: Powers real-time communication in study groups.
- **Notification**
    - **Responsibilities**: Delivers alerts for events, messages, or updates.
    - **Attributes**: id, message, timestamp.
    - **Methods**:
        - send(): Triggers a push notification via FCM.
        - observe(): Monitors notification updates (Observer Pattern).
    - **Role**: Keeps users informed in real-time.
- **PostFactory**
    - **Responsibilities**: Creates specific post types (Factory Pattern).
    - **Methods**:
        - createPost(type): Returns a Post instance based on the type (e.g., text, image).
    - **Role**: Simplifies post creation for the Home Feed.
- **FirestoreAdapter**

- **Responsibilities**: Converts Firestore data to app models and vice versa (Adapter Pattern).
- **Attributes**: firestore (Firestore instance).
- **Methods**:
  - toPost(doc): Maps a Firestore document to a Post.
  - toEvent(doc): Maps a Firestore document to an Event.
  - savePost(post): Saves a Post to Firestore.
  - saveEvent(event): Saves an Event to Firestore.
- **Role**: Ensures seamless data integration between Flutter and Firebase.

---

3. Relationships Between Classes

The relationships between classes define how they interact, supporting the app's functionality and design patterns.

- **User ↔ StudyGroup**
  - **Type**: Association (Many-to-Many).
  - **Description**: A User can join multiple StudyGroup instances, and a StudyGroup has multiple User members (stored in members).
  - **Example**: A student joins a group for a biology course.
- **User ↔ Post**
  - **Type**: Association (One-to-Many).
  - **Description**: A User can create multiple Post objects, and each Post is linked to its creator.
  - **Example**: A student posts an announcement in the Home Feed.
- **User ↔ MarketplaceItem**
  - **Type**: Association (One-to-Many).
  - **Description**: A User (as seller) can list multiple MarketplaceItem objects.
  - **Example**: A student lists a textbook for sale.
- **User ↔ ChatMessage**

    - **Type**: Association (One-to-Many).
    - **Description**: A User (as sender) can send multiple ChatMessage objects.
    - **Example**: A student sends a message in a study group chat.
- **StudyGroup ↔ ChatMessage**
    - **Type**: Aggregation (One-to-Many).
    - **Description**: A StudyGroup contains multiple ChatMessage objects, but messages can exist independently (e.g., archived).
    - **Example**: A group chat stores all messages sent by members.
- **PostFactory → Post**
    - **Type**: Creation (Factory Pattern).
    - **Description**: PostFactory creates instances of Post based on the specified type.
    - **Example**: Creating an image post vs. a text post.
- **FirestoreAdapter ↔ Post, Event**
    - **Type**: Dependency (Adapter Pattern).
    - **Description**: FirestoreAdapter converts Firestore data to Post or Event objects and saves them back to Firestore.
    - **Example**: Mapping a Firestore document to a Post object for display.
- **Observer Pattern Relationships**
    - **Type**: Dependency.
    - **Description**: Classes like User, StudyGroup, Event, ChatMessage, and Notification implement observe*() methods to subscribe to Firestore updates via StreamBuilder.
    - **Example**: A StudyGroup observes new ChatMessage objects to update the chat UI.
- **Event ↔ Notification**
    - **Type**: Association (One-to-Many).
    - **Description**: An Event can trigger multiple Notification objects (e.g., reminders).

- **Example**: An event sends a notification to attendees.