Here's the **Use Case Description** for the system, following the same format as the previous use cases:

**Use Case 1: Conclude Event**

**Primary Actor:**

* Admin (Event Organizer)

**Stakeholders and Interests:**

* **Student:**
  + Wants to see event outcomes, news, and photos, and provide feedback.
* **Department:**
  + Wants a record of completed events for documentation and future planning.
* **Event Organizer (Admin):**
  + Wants to officially mark the event as completed and share updates.

**Preconditions:**

* The event must have taken place.
* The admin must be logged into the system.

**Success Guarantee (Postconditions):**

* The event status is updated to "Completed."
* Photos and news are uploaded for students and teachers to view.
* The system allows students and teachers to provide ratings and feedback.

**Inputs:**

* Event ID
* Post-event summary
* Key highlights
* Photos/images
* Event news updates

**Outputs:**

* Event marked as "Completed."
* Uploaded images and news displayed to students and teachers.
* Rating and feedback section enabled for users.

**Main Success Scenario (Basic Flow):**

1. The admin selects the event to mark as completed.
2. The system prompts for post-event details (summary, key highlights, photos).
3. The admin uploads event news, images, and key outcomes.
4. The system updates the event status to "Completed."
5. The system enables the rating and feedback feature for students and teachers.

**Alternate Scenario (Extensions):**

* **a. System Failure at Any Time:**
  + The software saves existing data automatically.
  + When the system is reopened, it resumes from the last saved step.
* **3a. The admin does not have all post-event details:**
  + The system allows saving progress and updating later.
* **4a. Upload fails due to file size or format issues:**
  + The system notifies the admin and prompts for a compatible file.

**Special Requirements:**

* The application should be developed in Java.
* Student registration should be easily visible, using a readable font size (e.g., 12pt).

**Frequency of Occurrence:**

* Nearly continuous.

**Open Issues:**

* Open for feedback to improve the use case text.
* Consider additional input and output details for better usability.

**Use Case 2: Register for Event**

**Primary Actor:**

* Student

**Stakeholders and Interests:**

* **Student:**
  + Wants to register easily for eligible events and view registration details.
* **Department:**
  + Wants an organized system to manage student participation.
* **Event Organizer:**
  + Wants to monitor student registrations and ensure participant limits are maintained.

**Preconditions:**

* The student must be logged into the system.
* The event must be open for registration.

**Success Guarantee (Postconditions):**

* The student is successfully registered for the event.
* The system updates the event's participant list.

**Inputs:**

* Event ID
* Student ID
* Eligibility verification data
* Registration confirmation request

**Outputs:**

* Confirmation message of successful registration
* Updated participant list
* Error message (if registration fails due to eligibility or full capacity)

**Main Success Scenario (Basic Flow):**

1. The student navigates the events section and selects an event.
2. The system displays event details and eligibility criteria.
3. The student confirms eligibility and proceeds with registration.
4. The system registers the student, updates the participant list, and sends a confirmation.
5. The student can view or cancel their registration.
6. If canceled, the system updates the participant list and confirms cancellation.

**Alternate Scenario (Extensions):**

* **a. System Failure at Any Time:**
  + The software saves existing registration data automatically.
  + When the system is reopened, it resumes from the last saved step.
* **3a. Student is not eligible for the event:**
  + The system displays an error message and does not allow registration.
* **4a. Event registration limit is reached:**
  + The system notifies the student that registration is full and prevents further registration.

**Special Requirements:**

* The application should be developed in Java.
* Student registration should be easily visible, using a readable font size (e.g., 12pt).

**Frequency of Occurrence:**

* Nearly continuous.

**Open Issues:**

* Open for feedback to improve the use case text.
* Consider additional input and output details for better usability.

**Use Case 3: Manage Event**

**Primary Actor:**

* Admin

**Stakeholders and Interests:**

* **Admin:**
  + Needs to create, update, delete events, and set event requirements efficiently.

**Inputs:**

* Event details (name, description, location, date, time, participant limit)
* Edited event details
* Confirmation for deletion
* Requirement details (prerequisites, maximum participants)

**Outputs:**

* Confirmation of event creation
* Updated event details
* Deletion confirmation
* Error messages if required fields are missing or unauthorized access is attempted

**Preconditions:**

* The event management system must be available and functional.
* Only Admins can create, edit, delete, and set requirements for events.

**Postconditions:**

* The event is successfully created, updated, or deleted.
* Event requirements are correctly configured and stored.

**Main Success Scenario (Basic Flow):**

1. The admin navigates to the "Manage Event" section.
2. The system displays options for managing events.
3. The admin selects "Create Event."
4. The system prompts for event details (name, description, location, date, time, participant limit, etc.).
5. The admin enters the required details and submits the form.
6. The system validates the input and saves the event.
7. The system confirms event creation and adds it to the event list.
8. The admin selects an existing event to “Edit.”
9. The system displays the event’s details.
10. The admin modifies the necessary fields and submits changes.
11. The system validates and updates the event information.
12. The admin selects an event to “Delete.”
13. The system prompts for confirmation.
14. Upon confirmation, the system removes the event.
15. The admin selects "Set Event Requirements."
16. The system displays available requirement fields (e.g., prerequisites, maximum participants, category).

**Alternate Scenario:**

* **a. System Failure at Any Time:**
  + The software saves existing data automatically.
  + When the system is reopened, it resumes from the last step saved.
* **5a. If required fields are missing, the system prompts the admin to complete them.**
* **10a. If required fields are missing, the system prompts the admin to complete them.**

**Special Requirements:**

* The application should be developed in Java.
* Event details should be stored securely.

**Frequency of Occurrence:**

* Nearly continuous.

**Open Issues:**

* Open for feedback to improve the use case text.
* Consider additional input and output details for better usability.

**Use Case 4: View Event Feed (with Extended Event Rating)**

**Primary Actors:**

* **Student**
* **Teacher**

**Stakeholders & Interests:**

* **Students & Teachers:**
  + Want to see event updates, news, and photos.
  + Provide feedback by rating events.
* **Admin:**
  + Upload event-related news and images.
  + Ensure the event feed is updated with relevant content.

**Preconditions:**

* The user is authenticated (logged in).
* The admin has successfully uploaded event updates (news and/or photos).

**Success Guarantee (Postconditions):**

* The user successfully views the event feed, including photos and news.
* Any rating submitted by the user is stored, and the event’s overall rating is updated accordingly.

**Inputs:**

* Event selection by the user
* Rating submitted by the user (e.g., 1 to 5 stars)
* Feedback comments (optional)

**Outputs:**

* Display of event feed, including images and news updates
* Updated event rating
* Feedback comment (if provided)

**Main Success Scenario (Basic Flow):**

1. The student/teacher logs into the system.
2. The user navigates to the "Event Feed" section.
3. The system displays a list of completed events with images, news, and details.
4. The user selects an event to view.
5. The system displays detailed event information, including images and news updates.
6. The user has the option to rate the event by selecting a star rating (1-5).
7. If the user provides a rating, the system updates the event’s overall rating.
8. The user can also leave a feedback comment (optional).
9. The system stores the feedback and associates it with the event.
10. The event’s updated rating and feedback become visible to other users.

**Alternate Scenarios (Extensions):**

* **a. System Failure at Any Time:**
  + The system automatically saves event ratings and comments.
  + Users can reattempt their actions after the system recovers.
* **3a. No events available to view:**
  + The system displays a message: "No events available."
* **6a. User attempts to submit a rating but is not logged in:**
  + The system prompts the user to log in first.
* **8a. User enters inappropriate content in the feedback:**
  + The system detects inappropriate language and prompts the user to modify the comment.

**Special Requirements:**

* The application should be developed in Java.
* Event details and feedback should be stored securely.
* User ratings should be aggregated to show the overall event rating.

**Frequency of Occurrence:**

* Frequently after events are completed.

**Open Issues:**

* Consider adding a reporting system for inappropriate feedback.
* Implement an option to sort events based on ratings.