**Group 1 Use Cases (Iteration 2)**

**Darren Ross**

**Anthony Jack**

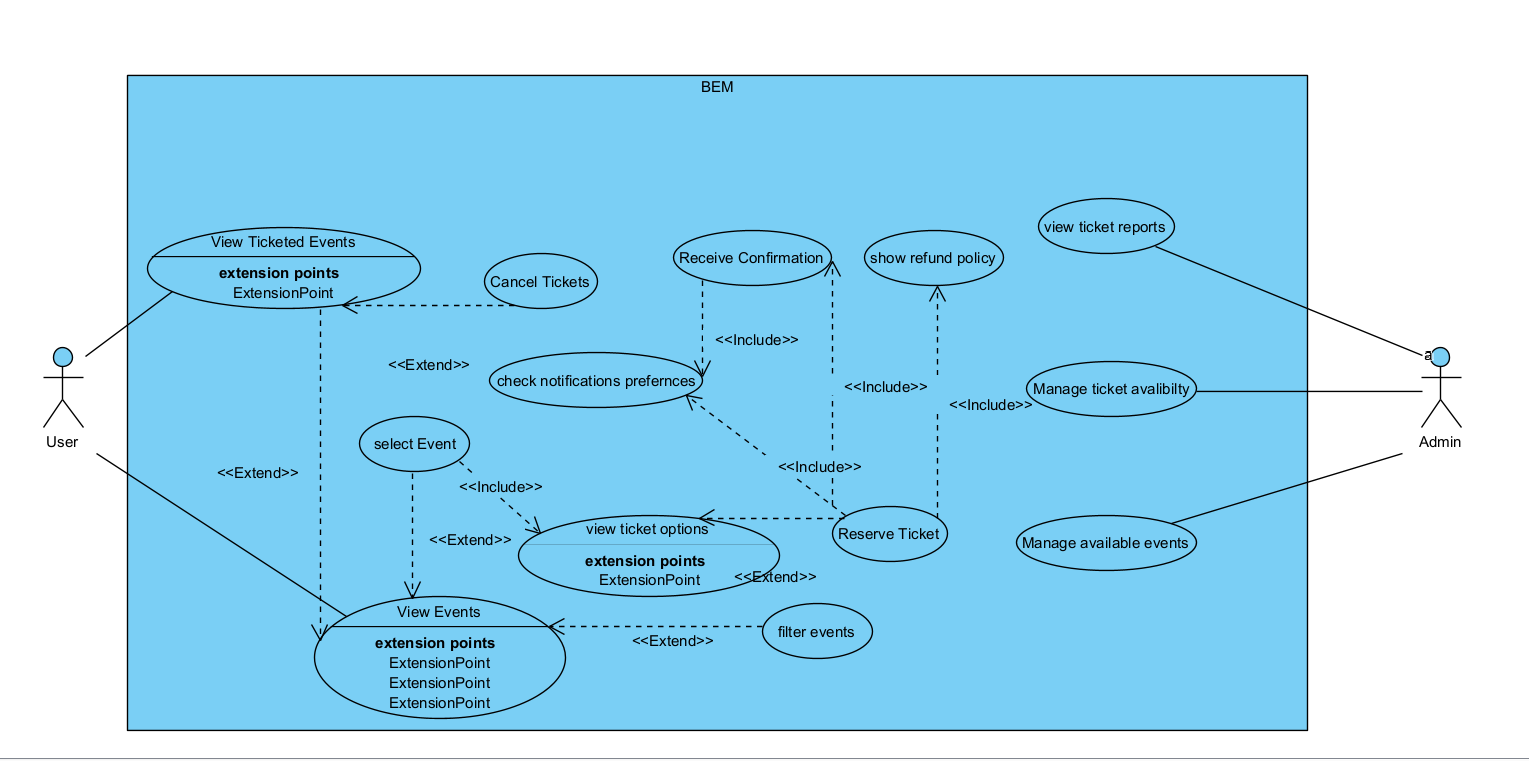
**Manikala Chevitipalli**

**Sai Achyuth Konda**

**GitHub Wiki Link:** <https://github.com/djgamekid/GDP-Group-I-bearcatmanager/wiki/Use-Cases-(Iteration-2)>

**Use Cases for Bearcat Manager**

**Use Case 1: Ticketing System for Event Management**

**Use Case Diagram:**  


**Primary Actor:** User (Student or Staff)  
**Scope:** Event Ticketing  
**Brief:** A User (Student or Staff) can purchase or reserve tickets for an event through the system. The system manages ticket availability, processes payments if applicable, and sends a confirmation and digital ticket to the User.

**Stakeholders:** Users (Students and Staff), Admin, University Administration, Payment Processing Service.

**Postconditions:**

* **Minimal Guarantees:**
  + The system will update the ticket availability for the event.
  + The User will receive a confirmation notification and a digital ticket.
* **Success Guarantees:**
  + The User successfully purchases or reserves a ticket for the event.
  + The User receives a digital ticket with all necessary event information.

**Preconditions:**

* The system displays a list of events with available tickets.
* The User is logged into the system.
* The User has a valid payment method.

**Triggers:** The User clicks on an event they are interested in and selects the option to purchase or reserve a ticket.

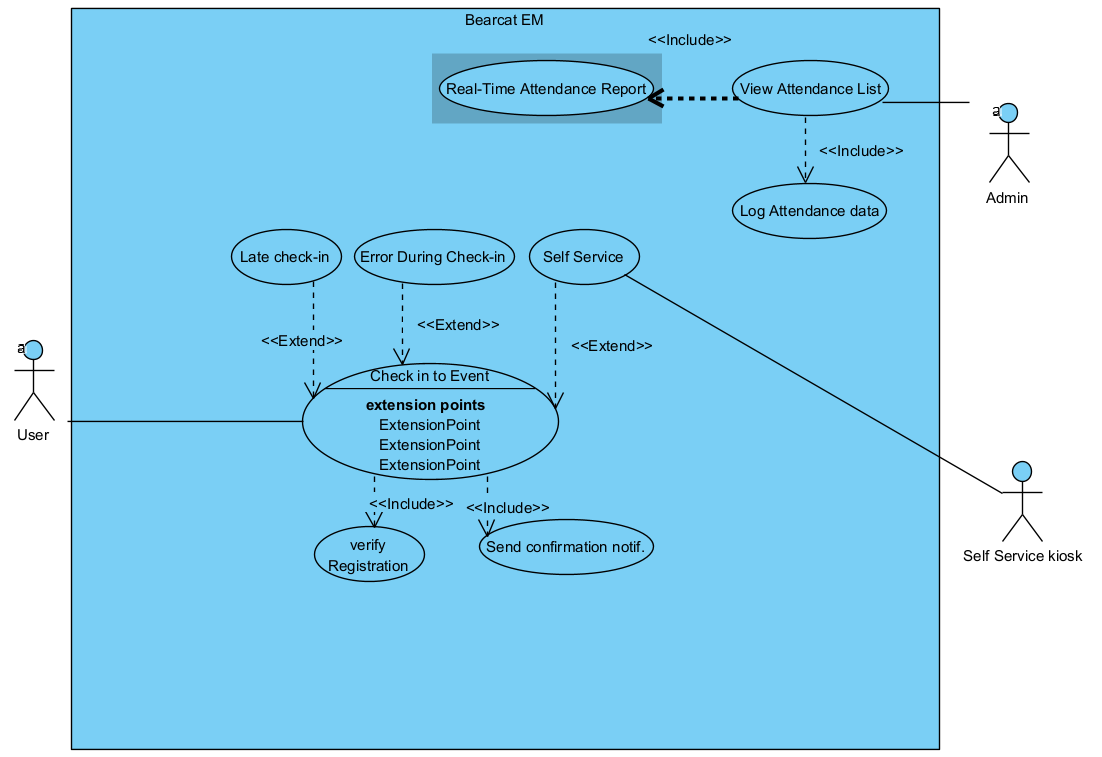
**Basic Flow:**

1. The system presents the User with a list of events that have available tickets.
2. The User selects an event and clicks the "Buy Ticket" or "Reserve Ticket" button.
3. The system displays the event details and ticket options (e.g., general admission, VIP).
4. The User selects the desired ticket option and quantity.
5. The system prompts the User to confirm their selection and proceed to payment (if applicable).
6. The User provides payment information (if applicable) and confirms the purchase or reservation.
7. The system processes the payment (if applicable) and updates the ticket availability.
8. The system sends a confirmation notification and a digital ticket to the User.
9. The system returns the User to the list of events, where they can see their ticketed events.

**Extensions:**

* Event Filtering and Sorting:
  + The system allows the User to filter and sort events by categories such as date, type, or location.
  + The system displays the filtered and sorted list of events based on the User's selection.
* Payment Processing:
  + The system integrates with a secure payment processing service to handle payments.
  + The system displays a payment confirmation page after successful payment processing.
* View Ticketed Events:
  + The system provides a section or filter on the dashboard where Users can view their ticketed events.
  + The system allows the User to cancel their ticket if they can no longer attend (subject to event policy).
  + Refund and Cancellation Policy: The system displays the refund and cancellation policy to the User before confirming the purchase or reservation.

**Use Case 2: Event Check-In for Registered Users**

**Use Case Diagram:**  


**Primary Actor:** User (Student or Staff)  
**Scope:** Event Check-In  
**Brief:** A registered User checks into an event on the day of the event. The system verifies the User’s registration, marks them as "checked in," and updates the attendance list. The system can also allow the User to check in via a mobile app or through a self-service system at the event venue.

**Stakeholders:** Users (Students and Staff), Admin, Event Organizers

**Postconditions:**

* **Minimal Guarantees:**
  + The system will mark the User as "checked in" for the event.
  + The system provides a real-time attendance report for event organizers, showing all checked-in attendees.
* **Success Guarantees:**
  + The User is successfully checked into the event and their attendance is recorded.
  + The system logs the attendance data for further analytics and reporting.

**Preconditions:**

* The User is registered for the event.
* The event is active (ongoing or about to start).
* The system has check-in options available (via app or self-service system).

**Triggers:** The User arrives at the event venue and clicks "Check-In" from the mobile app or checks in using a self-service system at the event location.

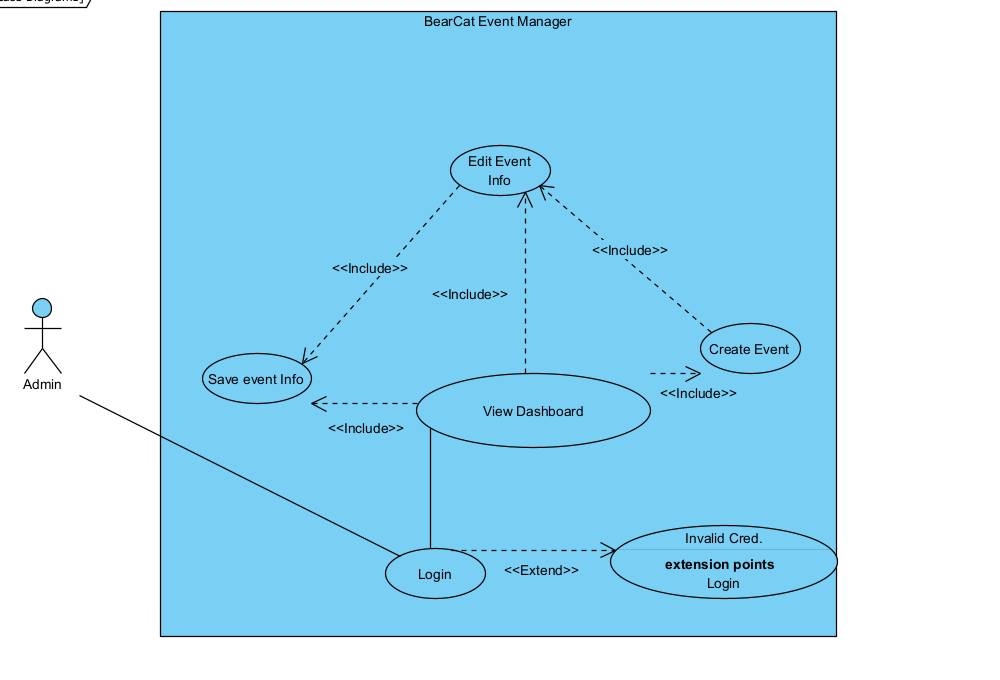
**Basic Flow:**

1. The system displays a "Check-In" button for the event on the User's dashboard or in the mobile app.
2. The User selects the event they want to check into and clicks "Check-In."
3. The system verifies the User’s registration status for the event.
4. Upon successful verification, the system marks the User as "checked in."
5. The system updates the event's attendance list to reflect the User's check-in.
6. The system sends a confirmation notification to the User, confirming the successful check-in.
7. The system updates the event organizer's dashboard to show the User's attendance.

**Extensions:**

* Self-service Check-In:
  + The User scans their event QR code or taps their ID card at the event.
  + The self-service system displays a confirmation screen, marking the User as "checked in."
* Late Check-In:
  + The system allows for a grace period after the event start time for late check-ins.
  + If the User checks in after the grace period, the system sends a notification indicating they are late.

## **Use Case 3: Creating Events for Students & Staff**

**Use Case Diagram:**  


**Primary Actor:** Admin (Administrative Director)  
**Scope:** Admin

**Brief:** The admin inputs details about an event, including the idea, time, location, and date. This event is then monitored with analytics and a list of accepted invites from students and staff.

**Stakeholders:** Admin, University Administration

**Postconditions:**

* **Minimal Guarantees:**
  + The admin will have a monitorable event, with notifications sent to students and staff after someone signs up.
* **Success Guarantees:**
  + An event card is created that is viewable by Users and updatable by the Admin.

**Preconditions:**

* The system lists available event spaces to the Admin.

**Triggers:**

* The admin clicks on an interface button labeled "Create a new Event."

**Basic Flow:**

1. The system presents a dashboard with analytics regarding any previously created events or a prompt to create a new event if none exist.
2. The system displays a button that links to a form page for the admin to enter event information.
3. The system lists fillable tabs along with a generated list of locations for selection.
4. The admin inputs the required information and clicks "Save," which stores and displays the information.
5. The system returns the admin to the dashboard with a viewable card that presents the event information in a styled layout.

**Extensions:**

1–2. **Welcome Message**

* The system greets the admin upon login.

**Editable from Dashboard**

* The system displays an edit button for each created event, if the event time has not passed.

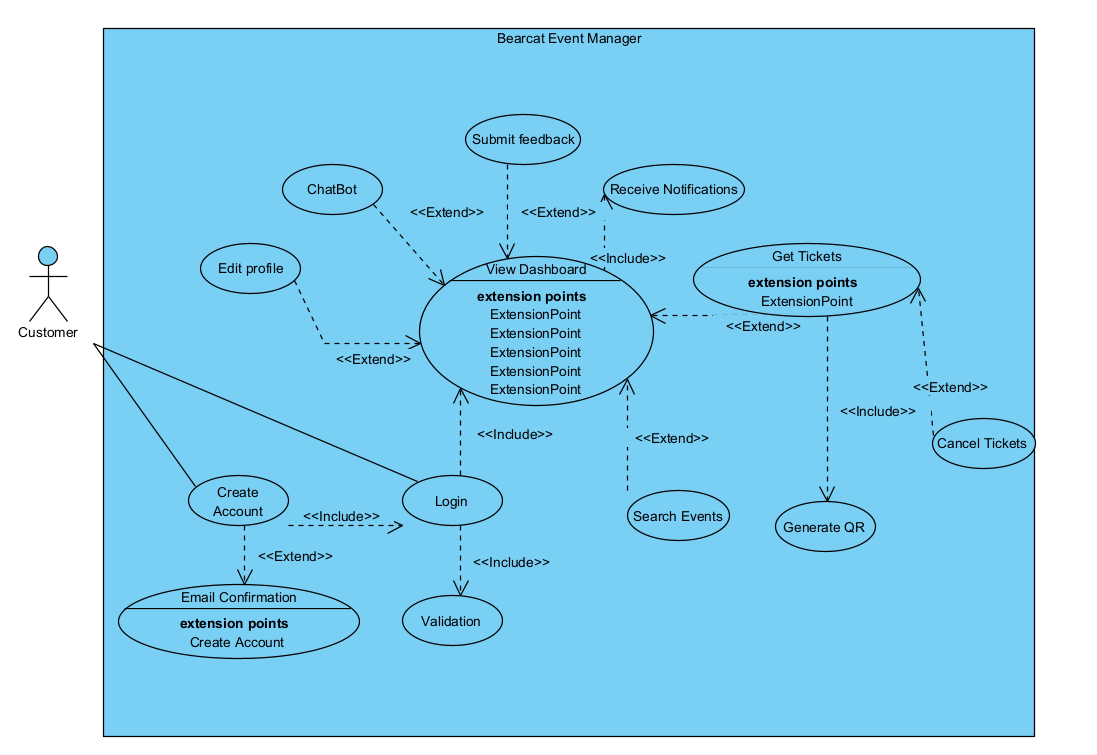
4–5. **Alert Confirming Input**

* The system provides an alert pop-up to confirm that the data entered is correct.

**Event Notification of Success**

* The system sends a notification to the Admin confirming the successful creation of the event.

## **Use Case 4: A User (Student or Staff) Registers for an Event**

**Use Case Diagram:**  


**Primary Actor:** User (Student or Staff)  
**Scope:** Event Registration

**Brief:** A User browses a list of upcoming events, selects one they’re interested in, and registers. The system then updates the event’s attendee list and sends a confirmation notification to the User.

**Stakeholders:** Users (Students and Staff), Admin, University Administration

**Postconditions:**

* **Minimal Guarantees:**
  + The system updates the event’s attendee list.
  + The User receives a confirmation notification of their registration.
* **Success Guarantees:**
  + The User successfully registers for the event.
  + The User receives complete event details in the confirmation notification.

**Preconditions:**

* The system displays a list of upcoming events available for registration.
* The User is logged into the system.

**Triggers:**

* The User clicks on an event of interest from the list of upcoming events.

**Basic Flow:**

1. The system displays a dashboard or list of upcoming events.
2. The User selects an event by clicking on it.
3. The system shows event details, including date, time, location, and description.
4. The User clicks the "Register" button for the event.
5. The system prompts the User to confirm their registration.
6. The User confirms registration.
7. The system updates the event’s attendee list with the User’s information.
8. The system sends a confirmation notification to the User with event details and a confirmation message.
9. The system returns the User to the dashboard or list of events, where they can view their registered events.

**Extensions:**

2–3. **Event Filtering**

* The system allows the User to filter events by categories such as date, type, or location, displaying the filtered list based on the selection.

5–6. **Registration Confirmation Alert**

* The system provides a pop-up to confirm the User’s registration intent, allowing the User to cancel or confirm.

7–8. **Notification Preferences**

* The system checks the User’s notification preferences (email, SMS, or in-app) and sends the confirmation accordingly, logging the notification for auditing.

9–10. **View Registered Events**

* The system provides a section or filter on the dashboard where Users can view their registered events.
* The system allows the User to cancel their registration if they no longer plan to attend.