## Grant Organizer Role

### SOS02-System-\*\*-Sunny\*\*

**Name**: SOS02-System-\*\*-Sunny\*\*

**Purpose**: Investigate normal execution of the Use Case with inputs which are similar to those expected by the system.

**Test set up**: The SOS system is set up and working. The Organizer is using Chrome as their browser. An Organizer is logged in and belongs to the chosen Organization and has rights to give privileges to other Members in the organization. This other Member exists in the system with ID “Johns01”. There is a Privilege in the system called “Event Manager” which can be assigned.

**Input**: The following sequence is done:

1. Organizer clicks on Add Organizer.
2. Organizer adds the following data:
   1. Member ID = “Johns01”
   2. Organizer Title = “Event Organizer”
   3. Powers and Privileges = “Event Manager”
3. Organizer clicks on Complete.

**Expected Output**: The system completes the request without exceptions or errors. A new Role object is saved to the database reflecting the relation between the Member and the Privilege. A new Request object is saved to the database reflecting the whole transaction. The Member is able to create events on the Organization.

### SOS02-System-\*\*-Sunny\*\*

**Name**: SOS02-System-\*\*-Sunny\*\*

**Purpose**: Investigate normal execution of the Use Case with inputs which are similar to those expected by the system.

**Test set up**: The SOS system is set up and working. The Organizer is using Firefox as their browser. An Organizer is logged in, is the current Owner of the chosen Organization and has rights to give privileges to other Members in the organization. This other Member exists in the system with ID “Jdoe001”. There is a Privilege in the system called “Owner” which can be assigned.

**Input**: The following sequence is done:

1. Organizer clicks on Add Organizer.
2. Organizer adds the following data:
   1. Member ID = “Jdoe001”
   2. Organizer Title = “President”
   3. Powers and Privileges = “Owner”
3. Organizer clicks on Complete.

**Expected Output**: The system completes the request without exceptions or errors. A new Role object is saved to the database reflecting the relation between the Member and the Privilege. A new Request object is saved to the database reflecting the whole transaction. The Member is now the owner of the Organization. The previous owner is no longer the owner of the Organization.

### SOS02-System-\*\*-Rainy\*\*

**Name**: SOS02-System-\*\*-Rainy\*\*

**Purpose**: Investigate the execution of the Use Case with inputs which are expected to break the system.

**Test set up**: The SOS system is set up and working. The Organizer is using Firefox as their browser. An Organizer is logged in and belongs to the chosen Organization and has rights to give privileges to other Members in the Organization. The target Member does not exist in the system. There is a Privilege in the system called “Event Manager” which can be assigned.

**Input**: The following sequence is done:

1. Organizer clicks on Add Organizer.
2. Organizer adds the following data:
   1. Member ID = “Jdoe001”
   2. Organizer Title = “Event Organizer”
   3. Powers and Privileges = “Event Manager”
3. Organizer clicks on Complete.

**Expected Output**: The system does not complete the request as it finds that the member ID cannot be found. It displays an error message explaining precisely that. A new Request object is saved to the database reflecting the whole transaction, noting the erroneous result.

### SOS02-System-\*\*-Rainy\*\*

**Name**: SOS02-System-\*\*-Rainy\*\*

**Purpose**: Investigate the execution of the Use Case with inputs which are expected to break the system.

**Test set up**: The SOS system is set up and working. The Organizer is using Firefox as their browser. An Organizer is logged in and belongs to the chosen but is missing the rights to give other Members roles. The target Member exists in the System and has the Member ID “Jdoe001”. There is a Privilege in the system called “Event Manager” which can be assigned.

**Input**: The following sequence is done:

1. Organizer clicks on Add Organizer.
2. Organizer adds the following data:
   1. Member ID = “Jdoe001”
   2. Organizer Title = “Event Organizer”
   3. Powers and Privileges = “Event Manager”
3. Organizer clicks on Complete.

**Expected Output**: The system does not complete the request as it finds that the Organizer does not have the rights to give other privileges. It displays an error message explaining precisely that. A new Request object is saved to the database reflecting the whole transaction, noting the erroneous result.

## Attending an Event

### SOS04-System-\*\*-Sunny\*\*

**Name**: SOS04-System-\*\*-Sunny\*\*

**Purpose**: Investigate normal execution of the Use Case with inputs which are similar to those expected by the system.

**Test set up**: The SOS system is set up and working. The Organizer is using Chrome as their browser. A Member, “jdoe001”, is logged in. This Member belongs to an Organization which is hosting an Event, “General Meeting”, and is currently at the physical location of that Event.

**Input**: The following sequence is done:

1. Member clicks on Events.
2. Member clicks on the “General Meeting” Event.
3. Member clicks on the “I’m Here!” button on the Event page.

**Expected Output**: The system completes the request without exceptions or errors. A new Attendance object is created and added to the Database reflecting the relationship between the User and the Event. It includes the correct real-time of attendance. A Request object is created reflecting the while transaction.

### SOS04-System-\*\*-Sunny\*\*

**Name**: SOS04-System-\*\*-Sunny\*\*

**Purpose**: Investigate normal execution of the Use Case with inputs which are similar to those expected by the system.

**Test set up**: The SOS system is set up and working. The Organizer is using Firefox as their browser. A Member, “johns01”, is logged in. This Member belongs to an Organization which is hosting an Event, “General Meeting”, and is currently at the physical location of that Event.

**Input**: The following sequence is done:

1. Member clicks on Events.
2. Member clicks on the “Fishing for Dummies” Event.
3. Member clicks on the “I’m Here!” button on the Event page.

**Expected Output**: The system completes the request without exceptions or errors. A new Attendance object is created and added to the Database reflecting the relationship between the User and the Event. It includes the correct real-time of attendance. A Request object is created reflecting the while transaction.

### SOS04-System-\*\*-Rainy\*\*

**Name**: SOS04-System-\*\*- Rainy\*\*

**Purpose**: Investigate the execution of the Use Case with inputs which are expected to create exceptions on the system.

**Test set up**: The SOS system is set up and working. The Organizer is using Chrome as their browser. A Member, “johns01”, is logged in. This Member belongs to an Organization which is hosting an Event, “General Meeting” but isn’t currently at the physical location specified by the Event.

**Input**: The following sequence is done:

1. Member clicks on Events.
2. Member clicks on the “Fishing for Dummies” Event.
3. Member clicks on the “I’m Here!” button on the Event page.

**Expected Output**: The system does not complete the request because it detects that the Member is not at the location specified by the Event. A message is communicated to the Member explaining precisely this. A Request object is created reflecting the while transaction, including its erroneous return.

### SOS04-System-\*\*-Rainy\*\*

**Name**: SOS04-System-\*\*-Rainy\*\*

**Purpose**: Investigate the execution of the Use Case with inputs which are expected to create exceptions on the system.

**Test set up**: The SOS system is set up and working. The Organizer is using Chrome as their browser. A Member, “johns01”, is logged in. This Member belongs to an Organization which is hosting an Event, “General Meeting” but their currently location is not available to the system.

**Input**: The following sequence is done:

1. Member clicks on Events.
2. Member clicks on the “Fishing for Dummies” Event.
3. Member clicks on the “I’m Here!” button on the Event page.

**Expected Output**: The system does not complete the request because the Member’s current location is not defined. A message is communicated to the Member explaining precisely this. A Request object is created reflecting the while transaction, including its erroneous return.