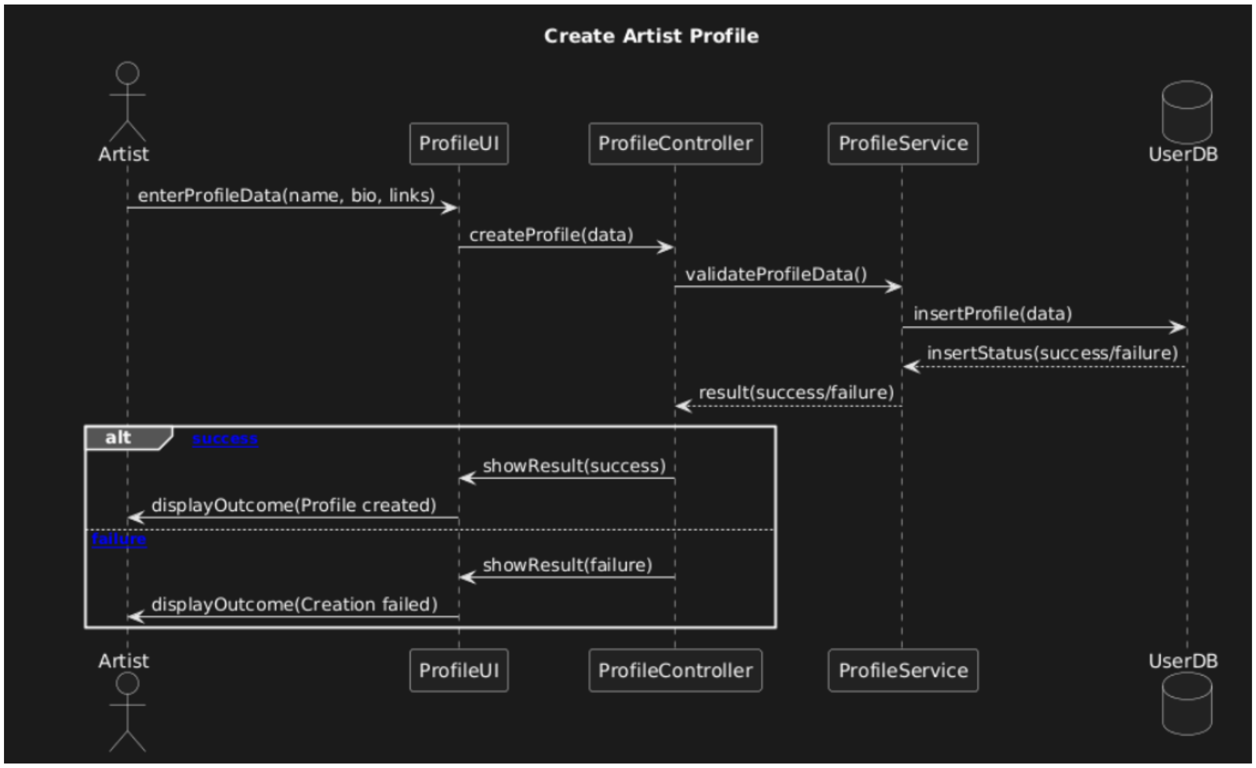
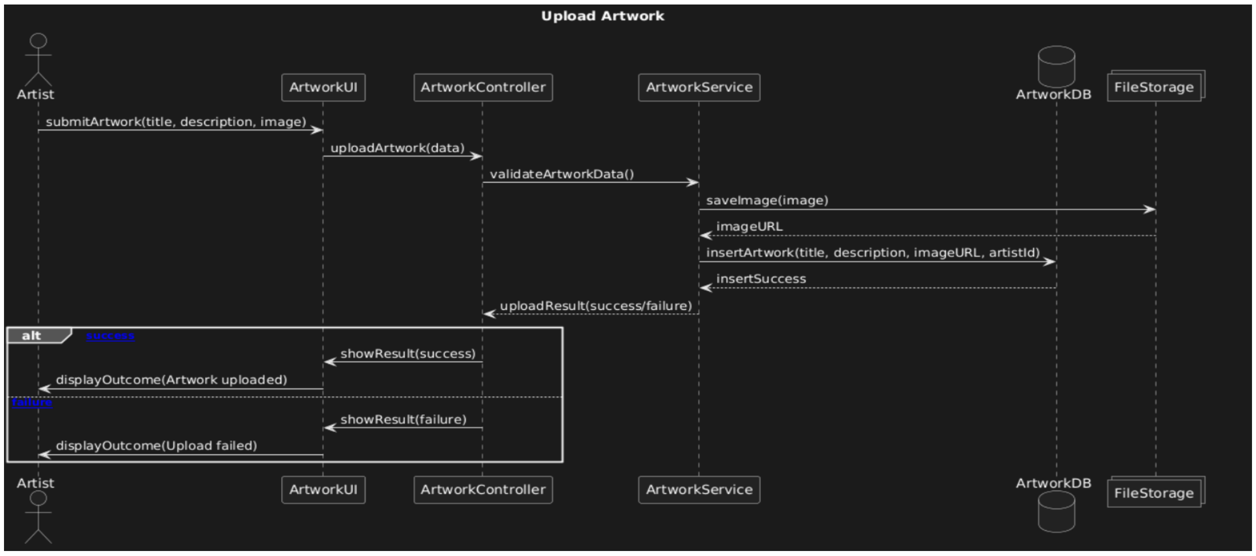
UML Sequence Diagrams:

Case 1: Create Artist Profile(Artist)



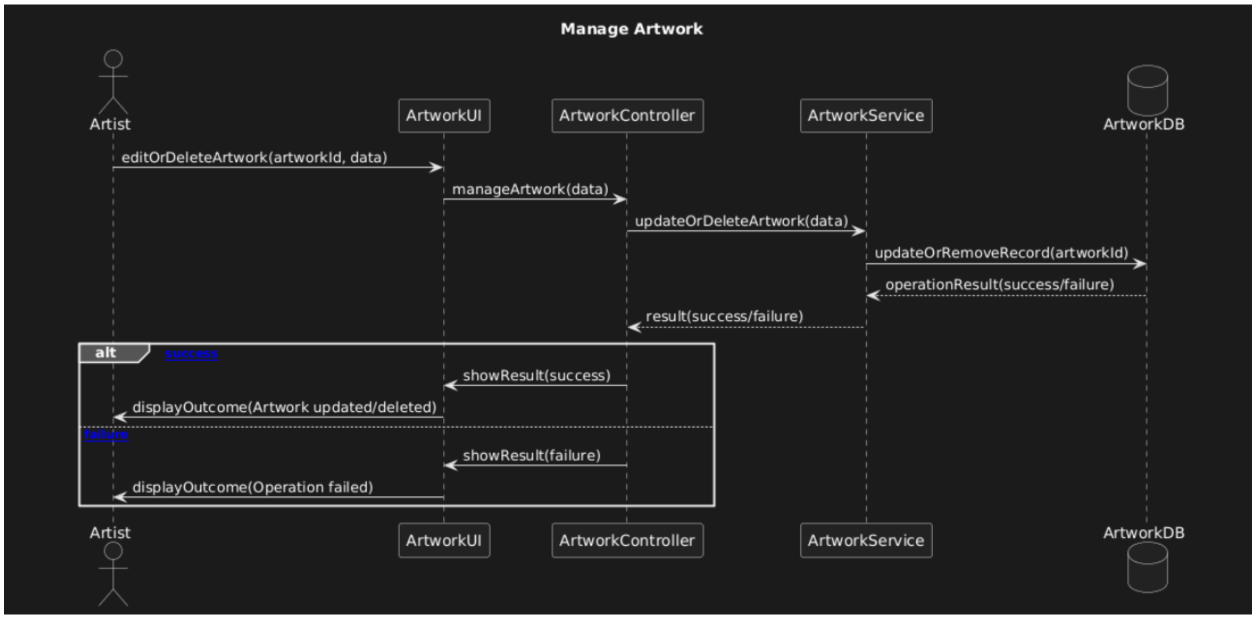
As a local artist, I want to create and edit my profile so that I can share information about myself and my artwork with visitors.The artist opens the ProfileUI and enters details such as name, bio, and contact information. The ProfileController receives the data and asks ProfileService to validate it before storing it in UserDB. If valid, the service returns success, the controller calls showResult(success), and the UI displays “Profile created.” If invalid, the alternate flow returns failure, and the UI displays “Profile creation failed.”

Case 2: Upload Artwork(Artist)



As a local artist, I want to upload images and descriptions of my artwork so that people can see my creations online. The artist submits the artwork title, description, and image through the ArtworkUI. The ArtworkController sends the data to the ArtworkService, which validates the input and saves the image in FileStorage. The ArtworkService then stores the details in ArtworkDB. If successful, it returns success to the controller, which calls showResult(success), and the UI displays “Artwork uploaded.” If validation or saving fails, the alternate flow calls showResult(failure) and the UI displays an error.

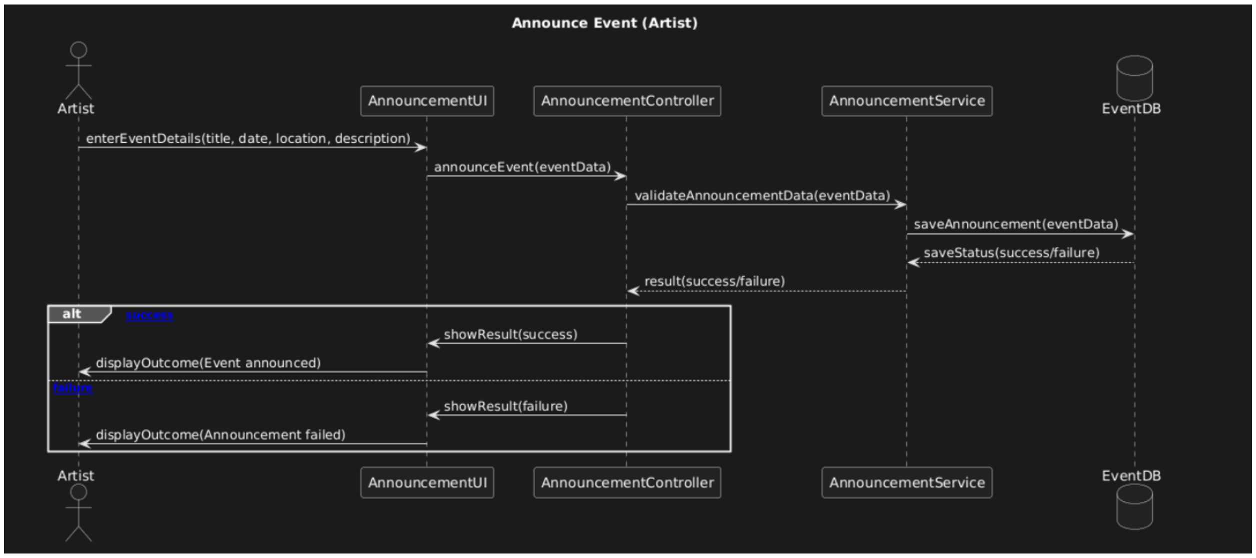
Case 3: Manage Artwork(Artist)


Manage Artwork
Artist
ArtworkUI
ArtworkController
ArtworkService
ArtworkDB
editOrDeleteArtwork(artworkId, data)
manageArtwork(data)
updateOrDeleteArtwork(data)
updateOrRemoveRecord(artworkId)
operationResult(success/failure)
result(success/failure)
alt
success
showResult(success)
displayOutcome(Artwork updated/deleted)
failure
showResult(failure)
displayOutcome(Operation failed)

As a local artist, I want to edit or delete previously uploaded artworks so that my gallery remains current and accurate. The artist selects an existing artwork from the ArtworkUI to modify or delete. The

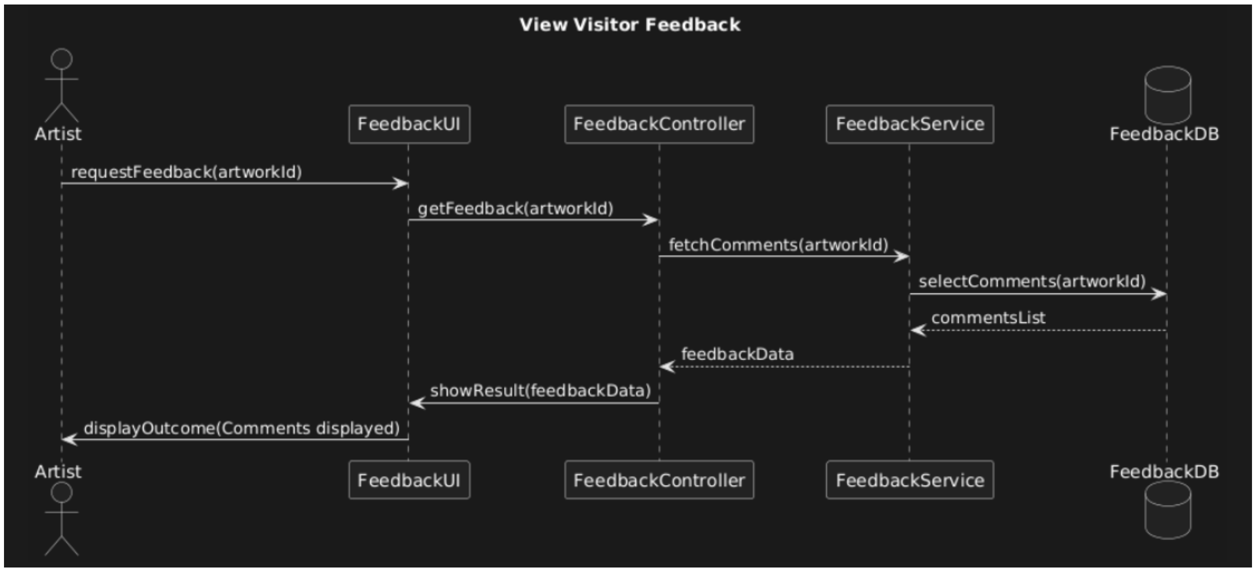
ArtworkController forwards the request to ArtworkService, which updates or removes the record in ArtworkDB. Upon completion, the service returns success or failure. The controller calls showResult(success) if the operation succeeds, and the UI displays “Artwork updated/deleted.” Otherwise, it calls showResult(failure) and displays “Update failed.”

Case 4: Announce Event(Artist)


Announce Event (Artist)
Artist
AnnouncementUI
AnnouncementController
AnnouncementService
EventDB
enterEventDetails(title, date, location, description)
announceEvent(eventData)
validateAnnouncementData(eventData)
saveAnnouncement(eventData)
saveStatus(success/failure)
result(success/failure)
alt
success
showResult(success)
displayOutcome(Event announced)
failure
showResult(failure)
displayOutcome(Announcement failed)

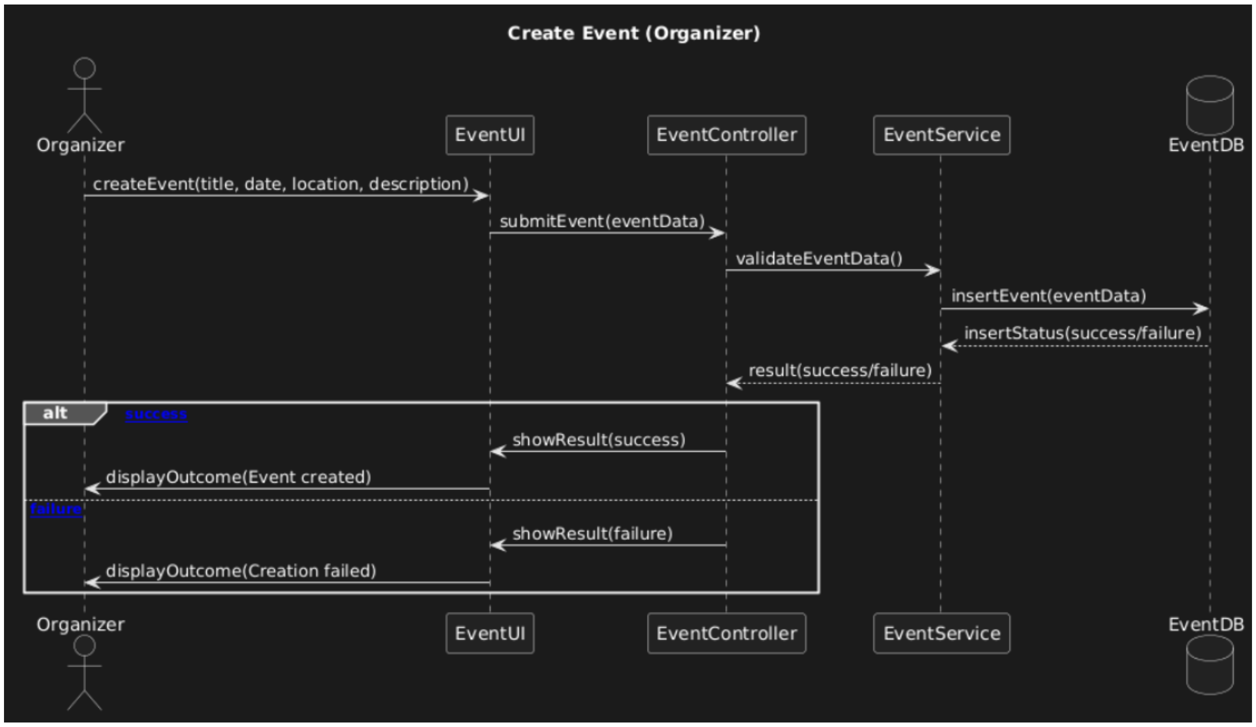
As a local artist, I want to announce upcoming exhibitions or art events so that visitors know where and when they can see my work. The artist enters event details such as title, date, location, and description in the AnnouncementUI. The AnnouncementController receives the event data and forwards it to the AnnouncementService, which validates the information and stores the announcement in the EventDB. Once the database confirms successful saving, the service returns success to the controller. The controller then calls showResult(success), and the UI displays “Event announced.” If any validation or saving error occurs, the alternate flow triggers showResult(failure), and the UI displays “Announcement failed.”

Case 5: View Visitors Feedback(Artist(Optional))



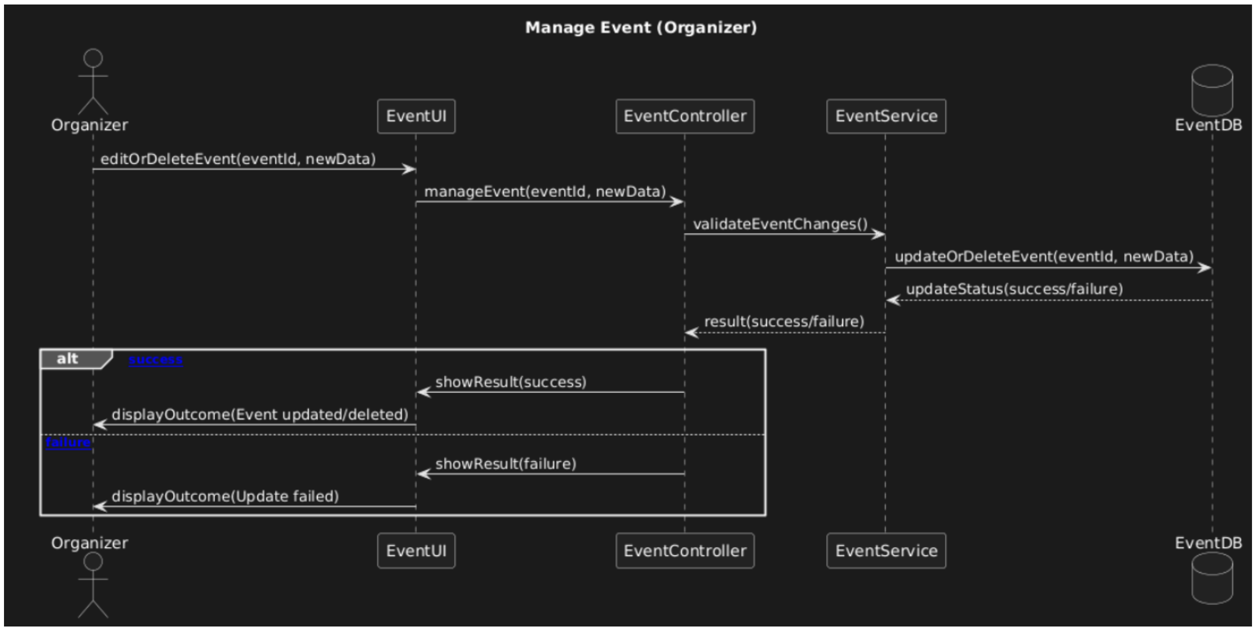
As a local artist, I want to view visitor feedback so that I can understand how people perceive my work and make improvements. The artist requests to view feedback through FeedbackUI. The FeedbackController retrieves data from FeedbackService, which queries FeedbackDB for comments related to the artist's artworks. The feedback list is returned to the controller, which calls showResult(success) and displays the results in the UI. The artist sees all comments via displayOutcome(Feedback displayed).

Case 6: Create Event



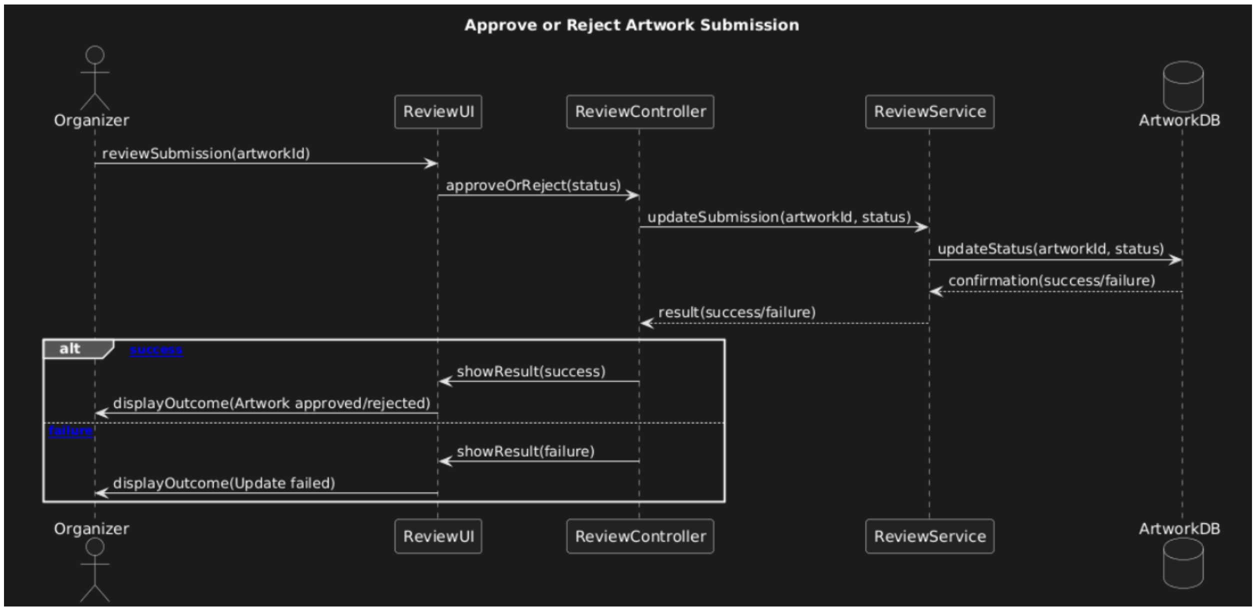
As an event organizer, I want to create new art events so that artists can participate and visitors can attend. The event organizer fills in event information using the EventUI. The EventController receives the input and asks EventService to validate it before saving the record in EventDB. If valid, the service returns success, the controller calls showResult(success), and the UI displays “Event created.” If invalid, the alternate flow returns failure and displays “Creation failed.”

Case 7: Manage Event


Manage Event (Organizer)
Organizer
EventUI
EventController
EventService
EventDB
editOrDeleteEvent(eventId, newData)
manageEvent(eventId, newData)
validateEventChanges()
updateOrDeleteEvent(eventId, newData)
updateStatus(success/failure)
result(success/failure)
alt
success
showResult(success)
displayOutcome(Event updated/deleted)
failure
showResult(failure)
displayOutcome(Update failed)

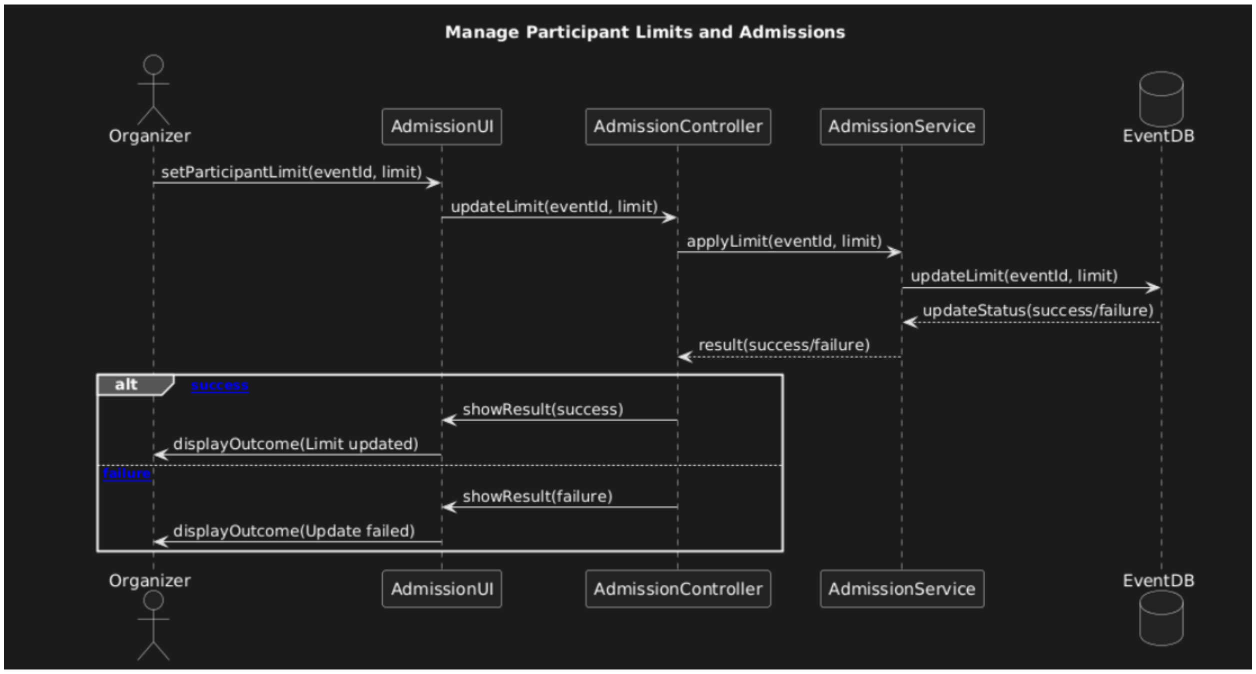
As an event organizer, I want to update, reschedule, or delete events so that the information presented to users remains correct. The organizer selects an event to manage in the EventUI. The EventController forwards the update or delete request to EventService, which modifies or removes the record in EventDB. Once the operation completes, EventService returns success or failure. The controller calls showResult(success) for success, and the UI displays “Event updated/deleted.” On failure, the alternate flow calls showResult(failure) and the UI displays “Update failed.”

Case 8: Approve or Reject Artwork Submission


Approve or Reject Artwork Submission
Organizer
ReviewUI
ReviewController
ReviewService
ArtworkDB
reviewSubmission(artworkId)
approveOrReject(status)
updateSubmission(artworkId, status)
updateStatus(artworkId, status)
confirmation(success/failure)
result(success/failure)
alt
success
showResult(success)
displayOutcome(Artwork approved/rejected)
failure
showResult(failure)
displayOutcome(Update failed)

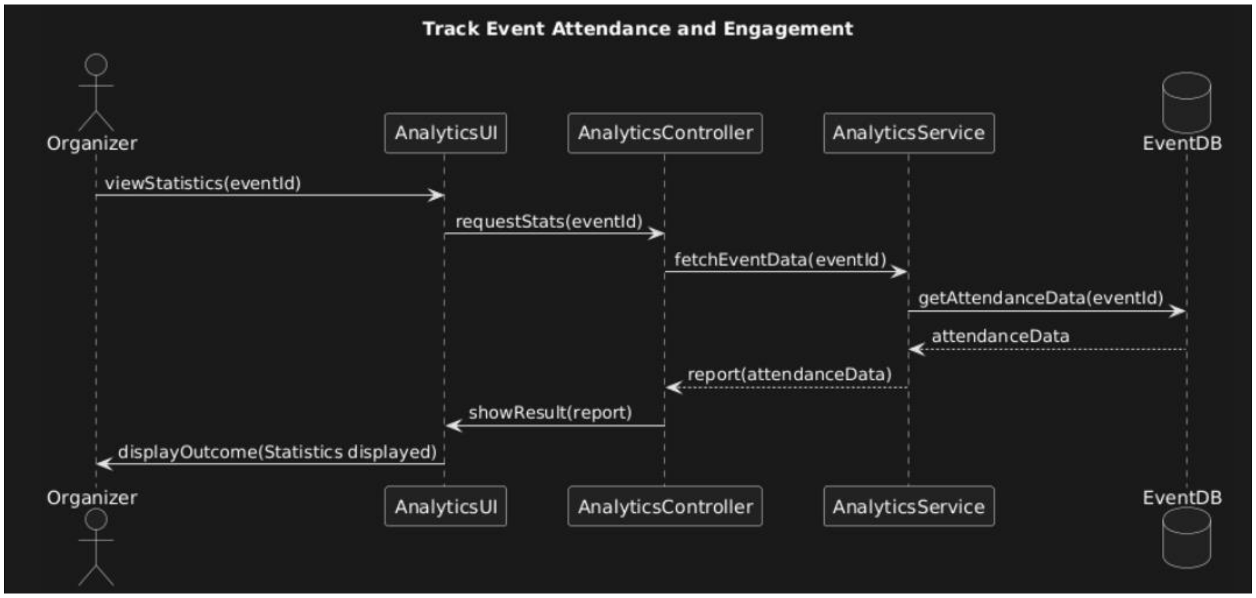
As an event organizer, I want to approve or reject artist submissions so that only approved artworks are displayed in exhibitions. The organizer reviews artist submissions using ReviewUI. The ReviewController sends the decision (approve or reject) to ReviewService, which updates the corresponding record in ArtworkDB. The service returns confirmation to the controller, which calls showResult(success) and the UI displays "Submission approved/rejected." If an error occurs, the alternate flow triggers showResult(failure) and displays "Operation failed."

Case 9: Manage Participant Limits and Admissions


Manage Participant Limits and Admissions
Organizer
AdmissionUI
AdmissionController
AdmissionService
EventDB
setParticipantLimit(eventId, limit)
updateLimit(eventId, limit)
applyLimit(eventId, limit)
updateLimit(eventId, limit)
updateStatus(success/failure)
result(success/failure)
alt
success
showResult(success)
displayOutcome(Limit updated)
failure
showResult(failure)
displayOutcome(Update failed)

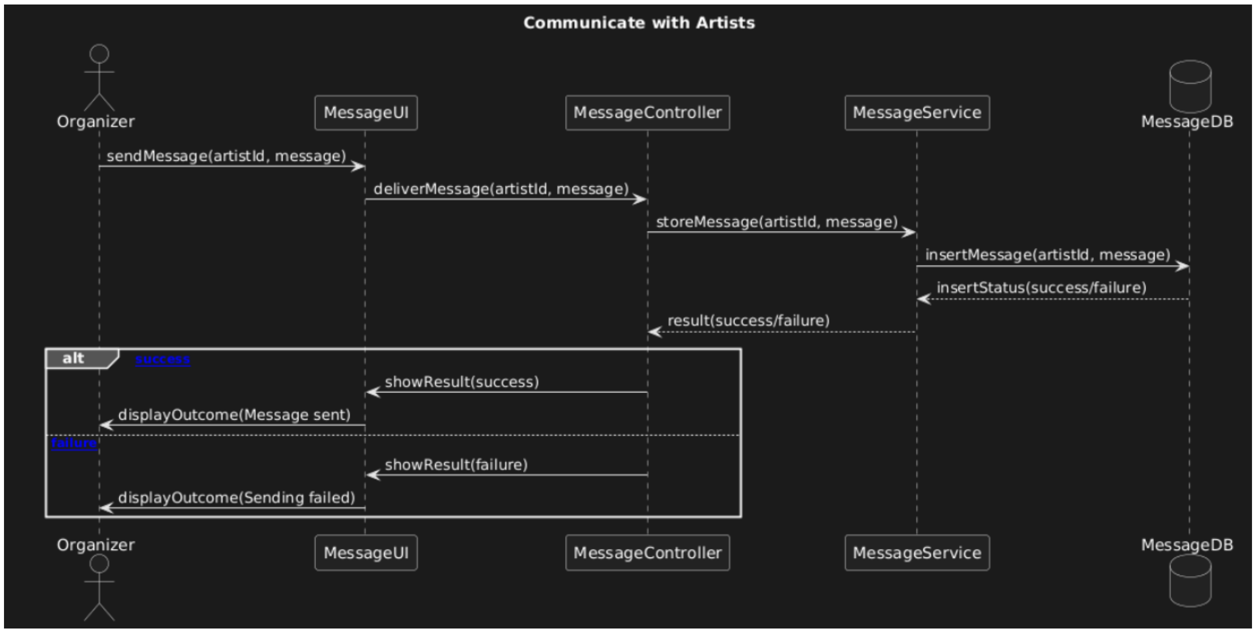
As an event organizer, I want to set participant limits and manage admissions so that event capacity and attendance are properly controlled. The organizer sets the maximum number of participants through the AdmissionUI. The AdmissionController sends the update request to AdmissionService, which modifies the capacity data in EventDB. When successful, the service returns success, the controller calls showResult(success), and the UI displays “Limit updated.” If the operation fails, the alternate flow triggers showResult(failure) and the UI displays “Update failed.”

Case 10: Track Event Attendance



As an event organizer, I want to track attendance and engagement so that I can measure the success of each event and plan improvements. The organizer requests attendance statistics via AnalyticsUI. The AnalyticsController asks AnalyticsService to gather data from EventDB. The service retrieves participation numbers and returns the report to the controller. The controller calls showResult(success), and the UI uses displayOutcome() to show event analytics and engagement metrics.

Case 11: Communicate with Artists



Communicate with Artists
Organizer
MessageUI
MessageController
MessageService
MessageDB
sendMessage(artistId, message)
deliverMessage(artistId, message)
storeMessage(artistId, message)
insertMessage(artistId, message)
insertStatus(success/failure)
result(success/failure)
alt
success
showResult(success)
displayOutcome(Message sent)
failure
showResult(failure)
displayOutcome(Sending failed)

As an event organizer, I want to communicate directly with artists so that I can coordinate event details and respond to inquiries. The organizer sends a message to an artist through MessageUI. The MessageController handles the request and forwards it to MessageService, which saves the message in MessageDB. If successful, MessageService returns confirmation, and the controller calls showResult(success) so the UI displays “Message sent.” If sending fails, the alternate flow returns failure, and the UI displays “Message delivery failed.”