Quick note

Some system operations are either very trivial, purely UI related, or both. The UI for this project was coded using Java Swing, where UI elements (such as new pages) are coded as Java classes. As such, some system operations are worth including in System Sequence Diagrams (because they relate to how an actor interacts with the system and what the system does) but are very trivial in the way they work. Thus, we decided that **those system operations should not have corresponding sequence diagrams.** Of course, operations with more complexity will have corresponding sequence diagrams. **The operations that have been highlighted have corresponding sequence diagrams.** There are also some "operations" in the system sequence diagrams that are surrounded by *, like *validations* and so on. **The "operations" surrounded by * will not be included in the operation contracts, as we don't consider them system operations, but they make the SSDs clearer.**

See below for the Operation Contracts.

SSD: Enter Organization Offerings

- Contract: AddOfferingPage
  - Operation: AddOfferingPage()
  - Cross References: Use Case: Enter Organization's Offerings
  - Preconditions:
    - Admin has logged into the system.
    - Admin has clicked on "View, add, edit and delete offerings"
    - A new OfferingsPage was created.
  - Postconditions:
    - A new AddOffering() is created (instance creation)
- Contract: enterNewOffering
  - Operation: enterNewOffering(classType: String, location: String, city: String, capacity: int, startTime: Timestamp, endTime: Timestamp)
  - Cross References: Use Case: Enter Organization's Offerings
  - Preconditions:
    - User has gone through AddOffering() operation
    - The inputs for city, location, capacity, startTime, endTime and classType are all valid according to the logic of the requirements and the prompts given to the Admin.
      - All inputs have been entered
      - Capacity is an integer > 1
      - startTime and endTime are in the valid format (yyyy-MM-dd HH:mm:ss)
      - endTime is after startTime
      - The offering is unique (no two offerings at the same time and location)
  - Postconditions:
    - An Offering with the specified parameters (city, etc.) is added to the DB (instance creation)
      - Note: The Offering has no associated Instructor nor Clients yet
    - A confirmation message is shown to the user

SSD: Take Offering

- Contract: takeOffering
  - Operation: takeOffering(offeringId: int)
  - Cross References: Use Case: Take on offerings
  - Preconditions:
    - Instructor has logged in to the system
    - Instructor has clicked on “View all offerings, select an offering to teach”
    - A new OfferingsPage was created.
    - There is at least one Offering.
    - Instructor has clicked on an Offering to take on.
    - The system has validated the following.
      - Offering city matches with instructor's cities
      - Offering is not already taken by an Instructor
      - Offering's class type matches instructor's specialty
  - Postconditions:
    - The Offering is associated to the instructor (association formed)
      - In the DB, the Offering has a column for the Instructor’s ID, which will now contain this Instructor’s ID
    - A confirmation message is shown to the user

SSD: Make Booking

- Contract: makeBooking
  - Operation: makeBooking(offeringId: int)
  - Cross References: Use Case: Make Bookings
  - Preconditions:
    - Client has logged into the system.
    - Admin has clicked on “View available offerings and make bookings”
    - A new OfferingsPage was created.
    - There is at least one Offering.
    - Client has clicked on an Offering to reserve.
    - The system has validated the following:
      - The Offering isn't at capacity
      - The Client hasn't already booked that Offering.
      - The Client hasn't booked another Offering at the same time and day.
  - Postconditions:
    - A new Booking is created in the database with offeringId and the Client’s id (instance creation, association formed)
    - A confirmation message is shown to the user

SSD: Make Booking as Guardian for Minor

- Contract: SelectMinorPage
  - Operation: SelectMinorPage(offeringId: int, startTime: String, endTime: String)
  - Cross References: Use Case: Make Bookings as Guardian for Minor
  - Preconditions:
    - Guardian has logged into the system.
    - Guardian has clicked on “View available offerings and make bookings”
    - A new OfferingsPage was created.
    - There is at least one available Offering.
    - Guardian has clicked on an Offering to reserve.
    - The system has validated the following that the Offering isn't at capacity
  - Postconditions:
  - A new SelectMinorPage(offeringId: int, startTime: String, endTime: String) was created (instance creation)
- Contract: makeBooking
  - Operation: makeBooking(minorId: int, offeringId: int)
  - Cross References: Use Case: Make Bookings as Guardian for Minor
  - Preconditions:
    - Guardian has chosen a Minor to make a Booking for.
    - The system has validated the following that the minor isn’t booked another Offering at the same time and day.
  - Postconditions:
    - An new Booking is created in the database with offeringId and the Minor’s id (instance creation, association formed)
    - A confirmation message is shown to the user