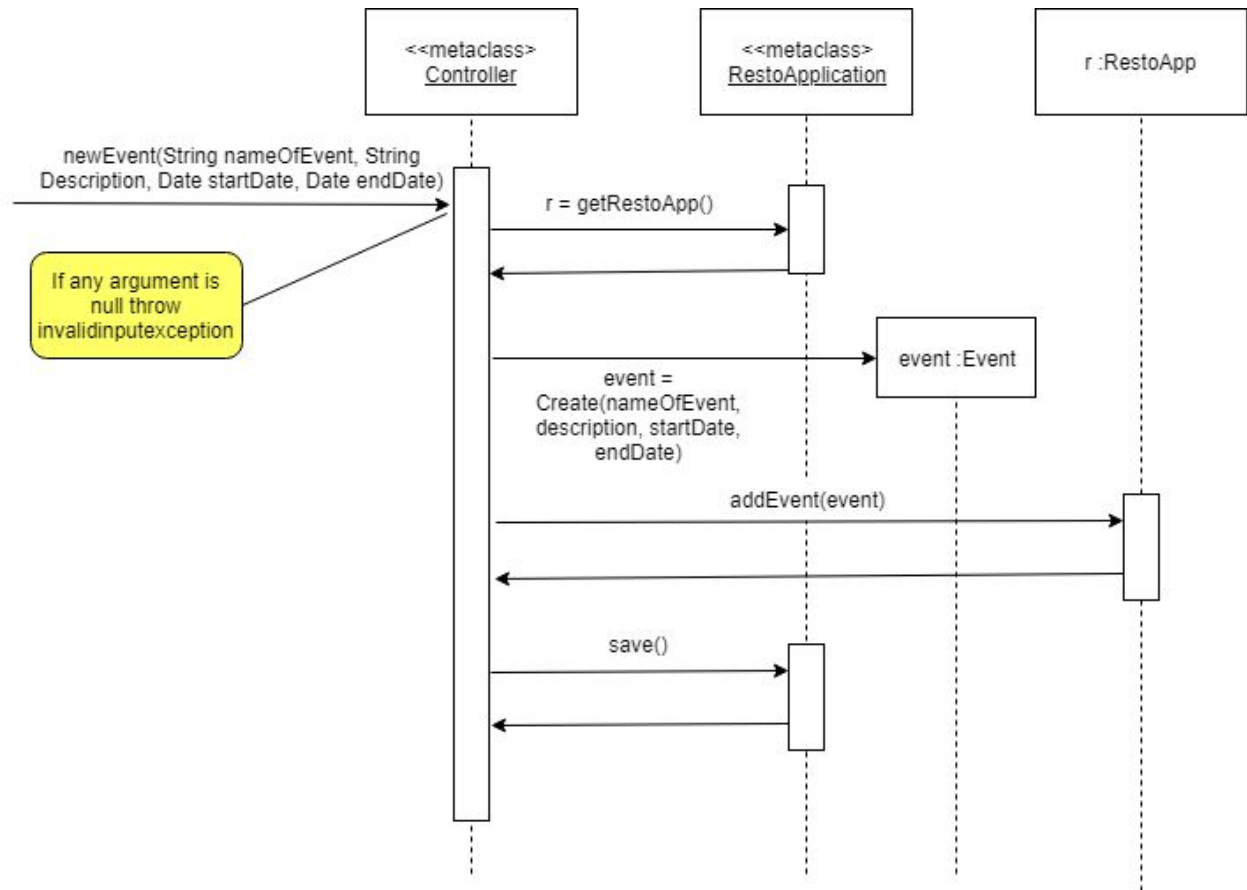


# Iteration 6

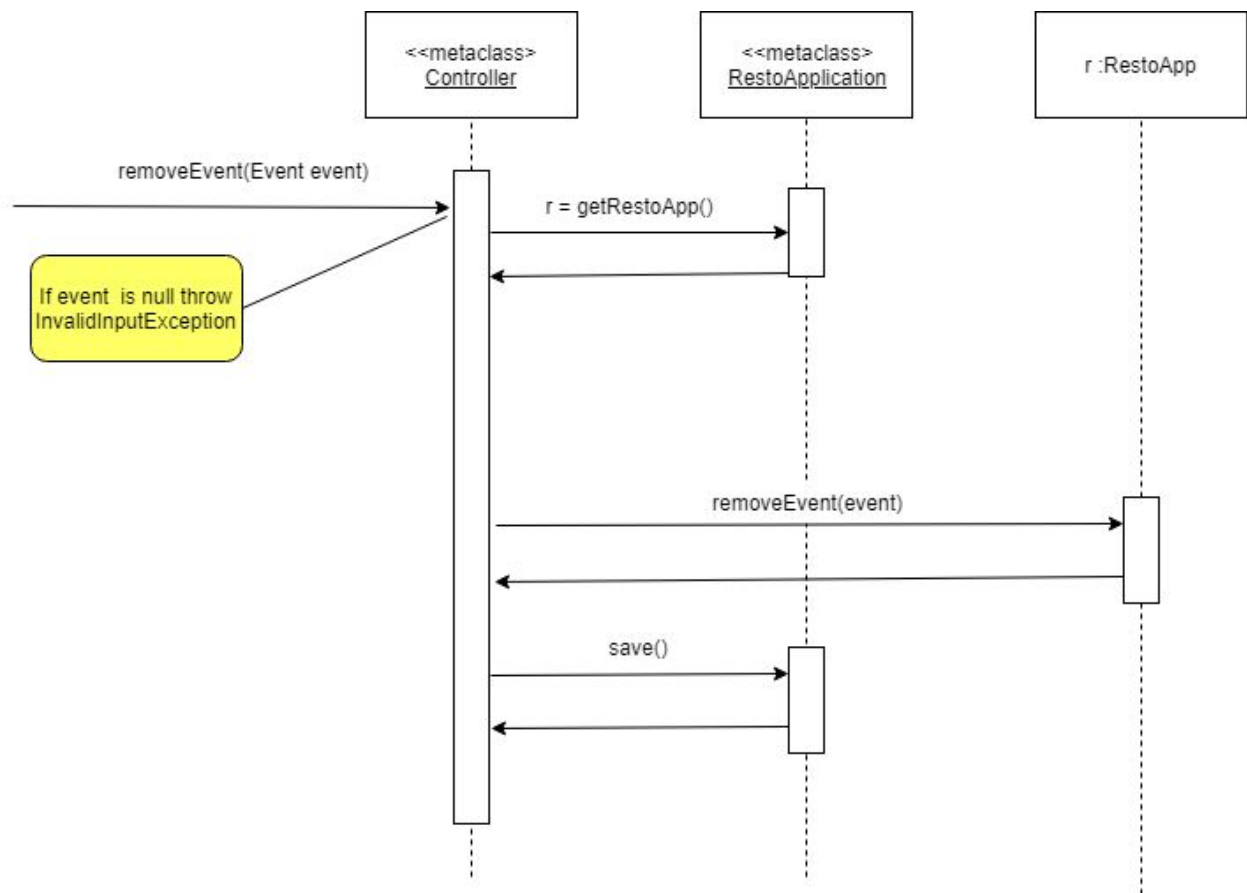
The new feature will have the ability for the owner to create events for the restaurant. For example, special evenings, seasonal events like Halloween or Christmas. The owner will input the following information: NameOfEvent, Description, StartDate, EndDate. They will then click a button labeled “Create Event” which will add it to a JTable which displays all events. The umple domain model will expand towards including a new class Event.



The following Controller interface will create a new Event:



The controller interface to remove/delete an event is as follows:



The UI-mockup for this new feature:

Resto App

Menu Categories

Name :

Number of person :

Date :

Time :

18:58

Tables :

1,2,3,4

Reservation

Event Name:

Start Date :

Description :

End Date :

Add Event

Name	Description	Start Date	End Date
------	-------------	------------	----------

Table Number:

X position:

Y position:

Table width :

Table height :

Number of seats :

select a table

▼

Delete

Update table

Move table

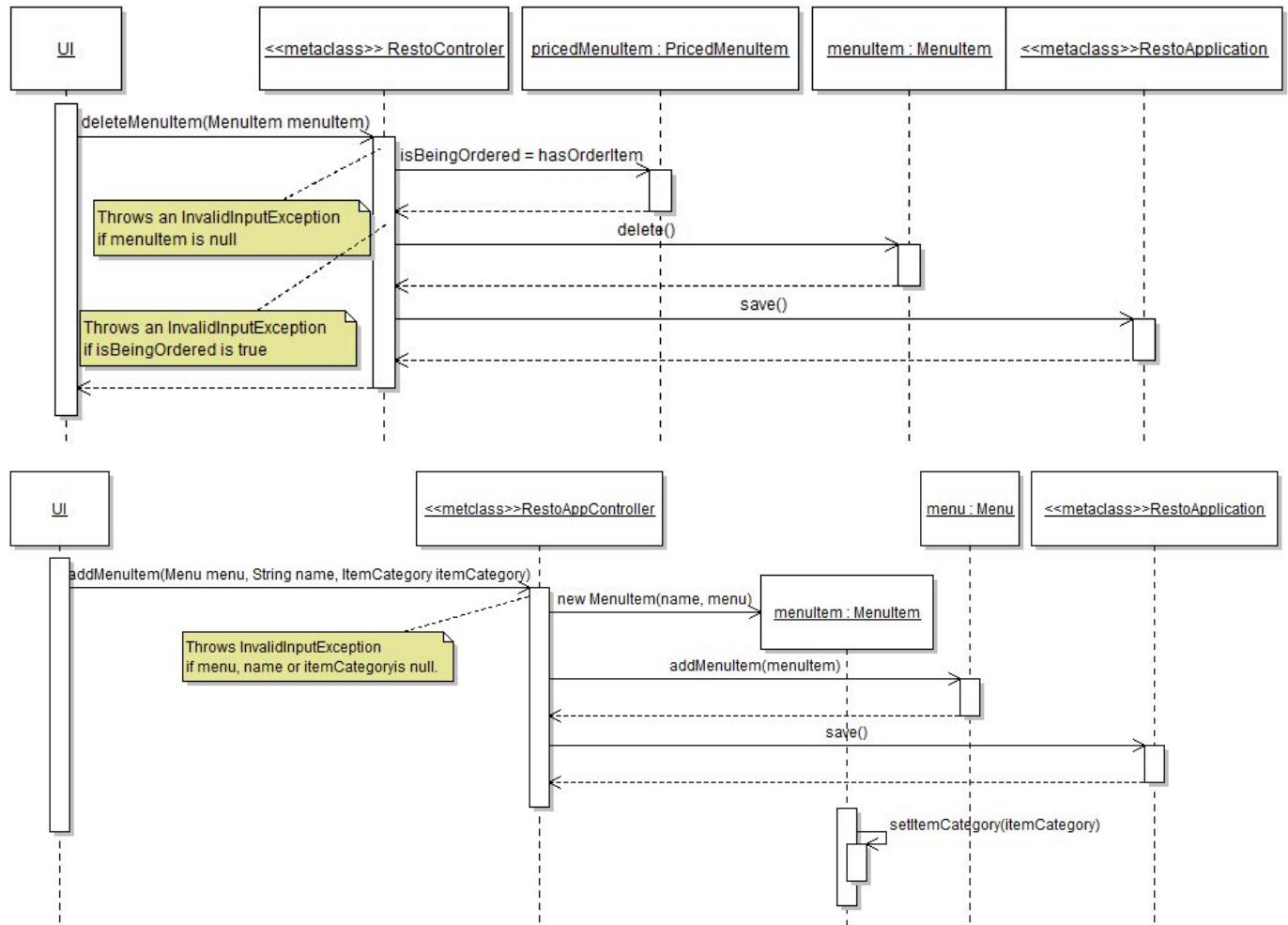
Add table

## Feature 1 : Update a menu item

Hakim Amarouhe(260692167)

Mock-up:

Sequence diagram:



**Note :** For `addMenuItem` `save()` will be called after the `itemCategory` is set, there is just a bug in jetUML preventing from representing it.

**Specification of controller:**

**`addMenuItem(Menu menu, String name, ItemCategory itemCategory)` throws `InvalidInputException`**

**`deleteMenuItem(MenuItem menuItem)` throws `InvalidInputException`**

**Feature 2 : Order item from menu for a customer at a table (it is possible to split a menu item between several customers)**

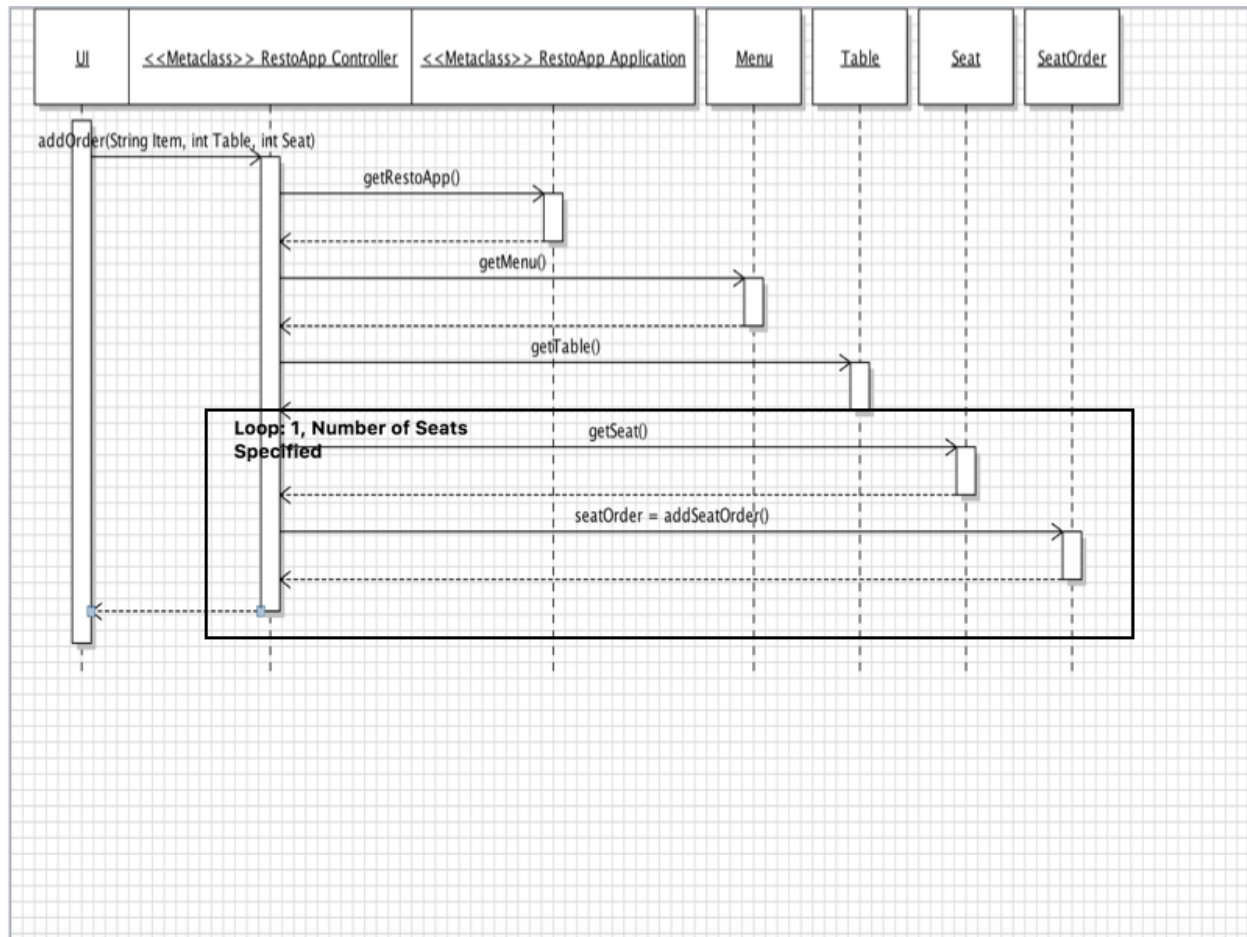
Ishraq Hossain (260720764)

UI Mock Up:

The UI mockup consists of several components:

- Table Layout:** A grid of tables represented by rectangles with circles around them indicating seats. Tables are numbered 1 through 6. Table 3 and Table 6 are marked as "(Reserved)".
- Form Fields:**
  - Table number:
  - X position:
  - Time:
  - Number of persons:
  - Phone number:
  - Email address:
- Menu Navigation:**
  - Buttons: View Menu, Appitizers, Main, Desserts, AlcoholicBeverage, Non-AlcoholicBeverage.
  - Buttons: Move Table, Menu, table order.
- Ordering Modal:** A central panel titled "Categorie" containing:
  - Item 1
  - Item 2
  - Item 3
  - Item 4
  - Table Number:
  - Seat Number:
  - Buttons: Add Order to Seat
- Reservation Button:** A button labeled "Reservation" at the bottom.

### Sequence Diagram:



### Specification of Controller Interface

addOrder(String item, int Table, int Seat)

Method to add the order to the seats

getRestoApp()

GetMenu()

Displays the menu to show the different items and boxes to select Table and Seats

getTable()

getSeat()

addSeatOrder()

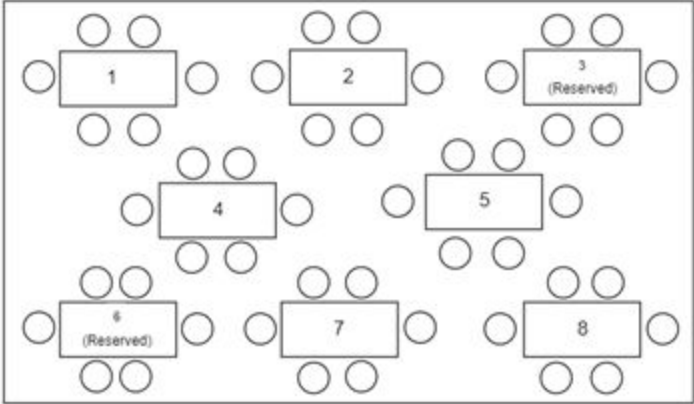
Fetches the Table and for each seat adds the order to the seat,

This method throws an InvalidInputException if Table and Seat is null

## Feature 3 : Cancel an ordered item or a whole order for a customer or table

Farouk Arab(260806502)

### Mock-up:



Name:

Date:

Time:

Number of persons:

Phone number :

Email address:

Reservation

Table number:

X position:

Y position:

Table width:

Table height:

Number of seats:

Select a table ▼

Delete

Update Table

Move Table

Add table

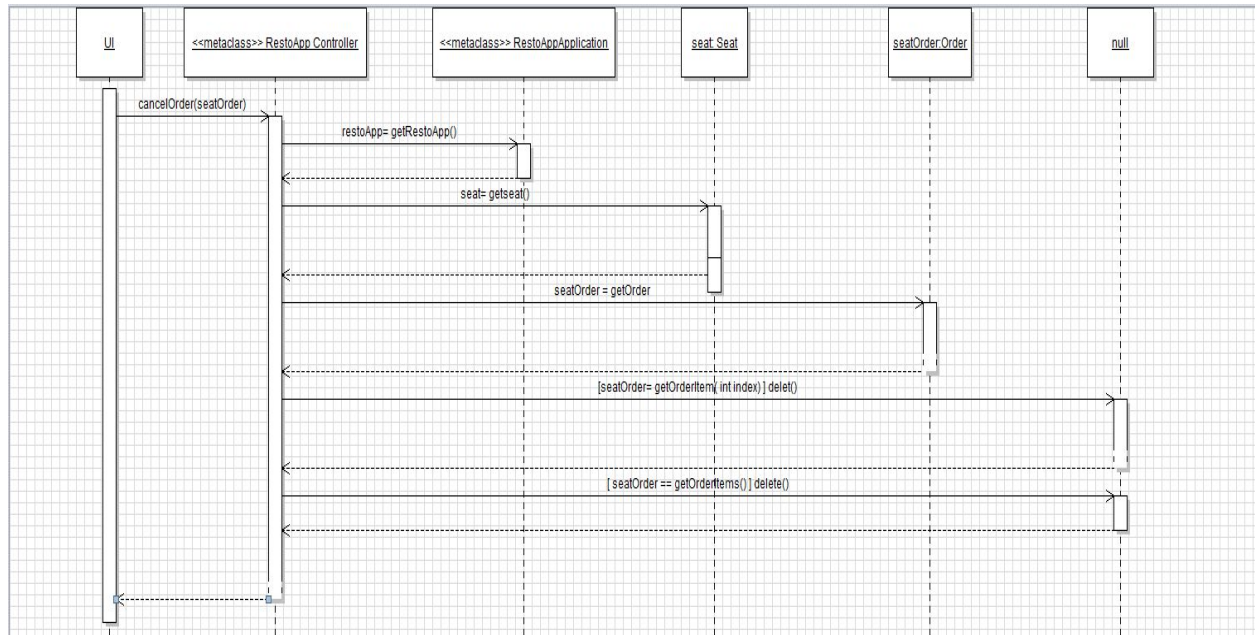
Menu

Cancel customer order

Cancel table order



## Sequence diagram:



## Specification of Controller Interface

CancelSeatOrder( SeatOrder)

getRestoApp()

getSeat()

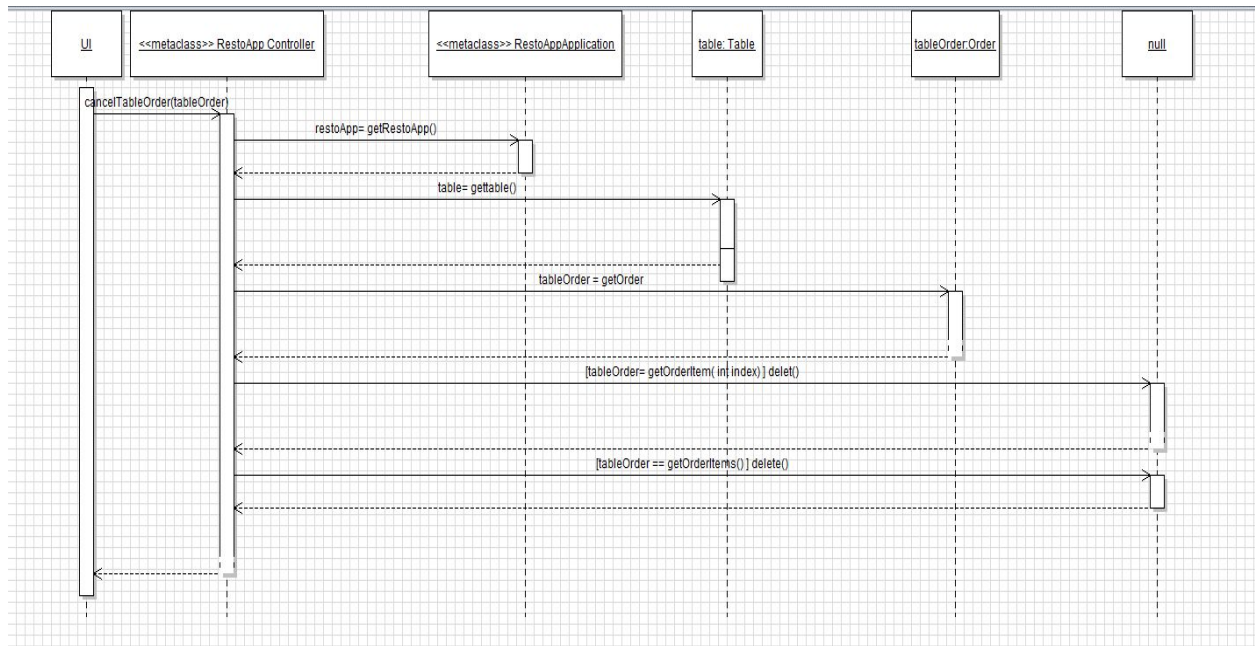
getOrder()

getOrderItem(int index)

getOrderItems()

delete()

## Sequence diagram:



## Specification of Controller Interface

CancelTableOrder( tableOrder)

getRestoApp()

getTable()

getOrder()

getOrderItem(int index)

getOrderItems()

delete()

## **Feature : View order for the customers of a table**

Done by Ziouti Ismail (260807219)

### **UI Mock-Up:**

The UI Mock-Up consists of a table layout diagram and a form for managing tables and viewing orders.

**Table Layout Diagram:** A rectangular area containing eight tables, each represented by a rectangle with circles around it indicating seats. The tables are numbered 1 through 8. Table 3 and Table 6 are marked as "(Reserved)".

**Form Fields:**

- Table number:
- X position:
- Y position:
- Table width:
- Table height:
- Number of seats:
- Select a table:

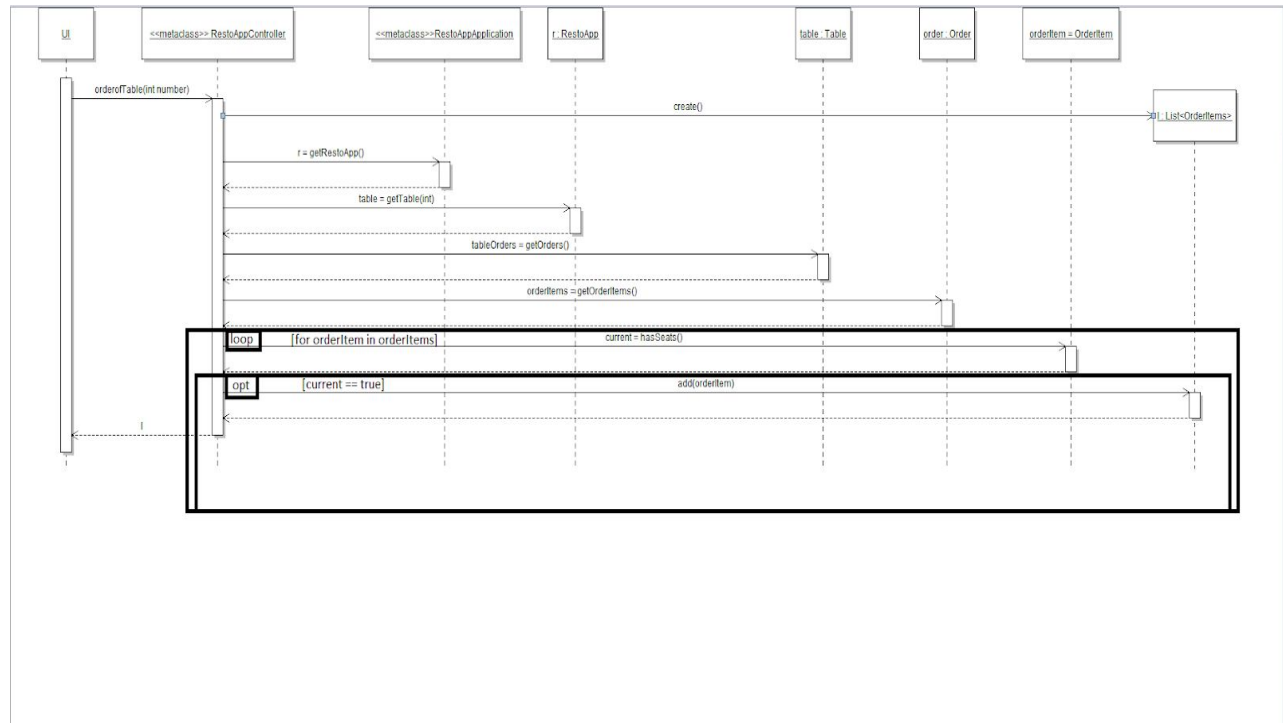
**Buttons:**

- Delete
- Update Table
- Move Table
- Add table
- Menu
- Cancel customer order
- Cancel table order
- View Order :
- Reservation

### **Specification of Controller Interface**

```
public static List<OrderItems> orderofTable(int number)
create()
getRestoApp()
getTable(int)
getOrders()
getOrderItems()
hasSeats()
add()
```

## Sequence Diagram:



## Feature 5 : Issue a bill

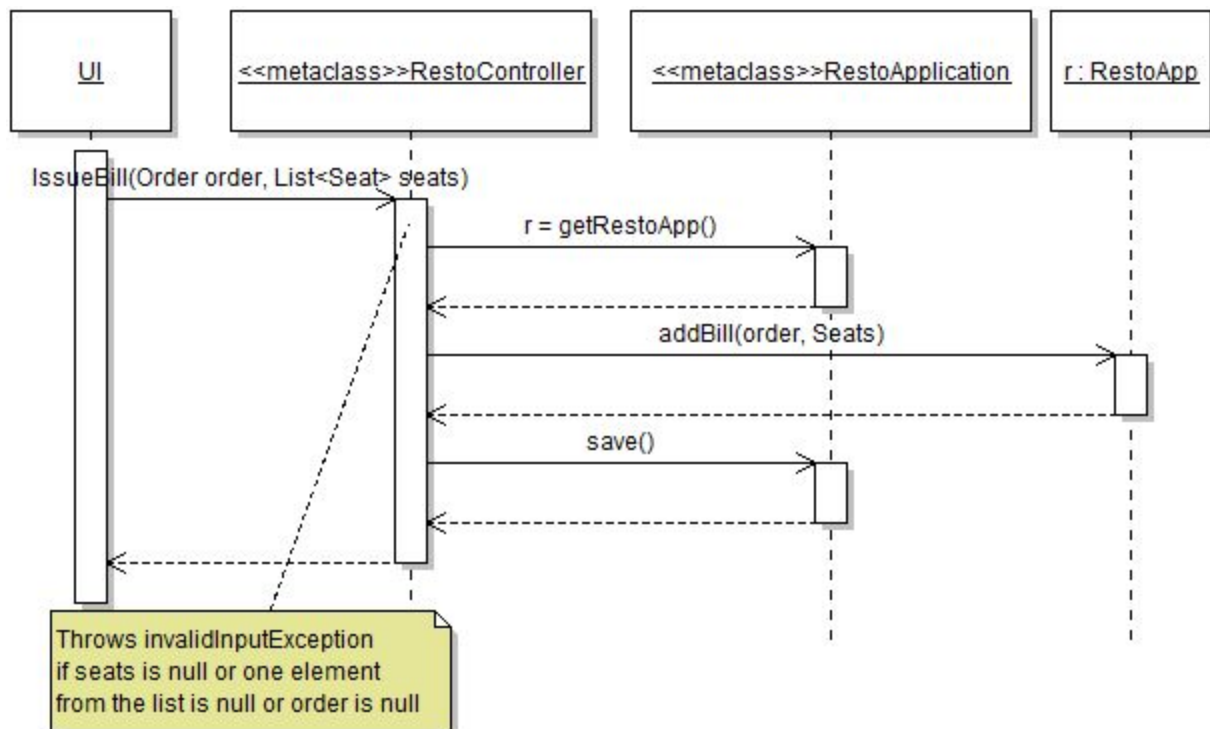
Hakim Amarouhe(260692167)

### Mock-up:

The mock-up shows a web application window titled "Resto App". It contains several sections:

- Menu Categories**: A large empty box on the left.
- Table Management**: Fields for "Table Number:", "X position:", "Y position:", "Table width:", "Table height:", and "Number of seats:". Below these are buttons for "Delete", "Update table", "Move table", and "Add table".
- Reservation Form**: Fields for "Name:", "Number of per...", "Date:", "Time:" (set to 23:28), "Tables:" (set to 1,2,3,4), "phone number:", and "email adress:". There is a "Reservation" button.
- Event Management**: Fields for "Event Name:", "Start Date:", "End Date:", and "Descripti...". There is an "Add Event" button.
- Table List**: A table with columns "Name", "Description", "Start Date", and "End Date".
- Issue Bill**: A button labeled "Issue Bill".

### Sequence diagram:



**Specification of controller:**

**IssueBill(Order order, List<Seat> seats) throws InvalidInputException**