Layout Manager: Backend Technical Breakdown

***Database***

* Restaurant Class
  + 1. Id (int primary key)
    2. Name (string length > 0)
    3. Password
    4. Width (length of restaurant; number > 0)
    5. Height (breadth of restaurant; number > 0)
    6. Padding (average spacing around table chair sets in restaurant; number > 0)
    7. Tables (one to many)
    8. Chairs (one to many)
    9. Objections (one to many)
    10. Furniture Groups (one to many)
    11. Sitting Areas (one to many)
    12. Reservations (one to many)
* Sitting Area Class
  + 1. Id (primary key)
    2. Width (length of sitting area; number > 0)
    3. Height (breadth of sitting area; number > 0)
    4. X1 (xmin of sitting area with origin at top left of restaurant)
    5. Y1 (ymin of sitting area with origin at top left of restaurant)
    6. Type (custom type of sitting area; nullable)
    7. Restaurant ID (many to one)
    8. Furniture Groups (one to many)
* Table Class
  + 1. Id (primary key)
    2. Width (length of table; number > 0)
    3. Height (breadth of table; number > 0)
    4. Capacity (amount of people table can accommodate; int > 0)
    5. X1 (xmin of table with origin at top left of restaurant)
    6. Y1 (ymin of sitting area with origin at top left of restaurant)
    7. Type (custom type of table; nullable)
    8. Restaurant ID (many to one)
    9. Furniture Group ID (one to one)
* Chair Class
  + 1. Id (primary key)
    2. Width (length of chair; number > 0)
    3. Height (breadth of chair; number > 0)
    4. X1 (xmin of chair with origin at top left of restaurant)
    5. Y1 (ymin of chair with origin at top left of restaurant)
    6. Type (custom type of chair; nullable)
    7. Restaurant ID (many to one)
    8. Furniture Group ID (many to one)
* Objection Class
  + 1. Id (primary key)
    2. Width (length of objection; number > 0)
    3. Height (breadth of objection; number > 0)
    4. X1 (xmin of objection with origin at top left of restaurant)
    5. Y1 (ymin of objection with origin at top left of restaurant)
    6. Type (custom type of objection; nullable)
    7. Restaurant ID (many to one)
* Furniture Group Class
  + 1. Id (primary key)
    2. Orientation
    3. Padding
    4. Table (one to one)
    5. Chairs (one to many)
    6. X1 (xmin of group with origin at top left of restaurant)
    7. Y1 (ymin of group with origin at top left of restaurant)
    8. Type (custom type of group; nullable)
    9. Restaurant ID (many to one)
    10. Sitting Area ID (many to one)
* Reservation Class
  + 1. Id (primary key)
    2. Number (number of people this objection is for; number > 0)
    3. Last Name (last name of person making reservation; string of length > 0))
    4. Mobile Number (valid mobile number)
    5. Date (date string in yyyy-mm-dd format)
    6. Time (time string in hh-mm 24-hour format)
    7. Duration (duration of reservation in minutes; int > 0)
    8. Restaurant ID (many to one)
    9. Group ID (one to one)
    10. Group2 ID (one to one)

***Backend Operations and API***

* addChairToRestaurant(restID, width, height, number, type)
  1. URL: */restaurants/<int:rest\_id>/chairs/*
  2. Method:POST
  3. Body:
     1. **width**: length of chair; number > 0.
     2. **height**: breadth of chair; number > 0.
     3. **number**: number of chairs to add; int > 0; default 1.
     4. **type**: custom chair type; string of length > 0; default none.
* addGroupToRestaurant(restID, tableID, chairsID, coords, padding, orientation, sitID, topLeft)
  1. URL: */restaurants/<int:rest\_id>/groups/*
  2. Method:POST
  3. Body:
     1. **table**: ID of the table in this group; must be a valid table ID int.
     2. **chairs**: list of IDs of chairs in this group; must be a list of valid chair ID ints.
     3. **coords**: list representing coordinates of this group with respect to origin at the top left of the restaurant in the format [xmin, ymin, xmax, ymax]; valid list of numbers.
     4. **padding**: extra spacing around the table and chair set-up; number > 0.
     5. **orientation**: the orientation of this table; either “vertical” or “horizontal”.
     6. **sittingArea**: the ID of the sitting area this group is currently in; if not placed i.e. if coords = None or None in coords, this is None; else it must be a valid sittingArea ID int.
* addObjectionToRestaurant(restID, coords, type)
  1. URL: */restaurants/<int:rest\_id>/objections/*
  2. Method:POST
  3. Body:
     1. **coords**: list representing coordinates of this objection with respect to origin at the top left of the restaurant in the format [xmin, ymin, xmax, ymax]; valid list of numbers.
     2. **type**: custom objection type; string of length > 0; default none.
* addReservationToRestaurant(restID, number, lastName, mobile, date, time, duration, grpID, grp2ID)
  1. URL: */restaurants/<int:rest\_id>/reservations/*
  2. Method:POST
  3. Body:
     1. **number**: number of people this reservation is for; int > 0.
     2. **lastName**: last name of the person making this reservation. String of length > 0.
     3. **mobileNumber**: contact number of the person making this reservation. Valid mobile number.
     4. **date**: the date for this reservation; sent from frontend in “yyyy-mm-dd” format.
     5. **time**: the time for this reservation; sent from frontend in “hh-mm” 24 hour format.
     6. **duration**: the duration in minutes of this reservation; int > 0; default 60.
     7. **grpID**: the first group that is reserved for this reservation; must be a valid group ID int; default None.
     8. **grp2ID**: the second group that is reserved for this reservation; must be a valid group ID int; default None; not None only if first group is not None and first group capacity < number of people for this reservation.
* addSittingAreaToRestaurant(restID, coords, padding, type)
  1. URL: */restaurants/<int:rest\_id>/sittingAreas/*
  2. Method:POST
  3. Body:
     1. **coords**: list representing coordinates of this sittingArea with respect to origin at the top left of the restaurant in the format [xmin, ymin, xmax, ymax]; valid list of numbers.
     2. **padding**: spacing around all the table and chair set-ups in this sittingArea; number > 0.
     3. **type**: custom sittingArea type; string of length > 0; default none.
* addTableToRestaurant(restID, width, height, capacity, number, type)
  1. URL: */restaurants/<int:rest\_id>/tables/*
  2. Method:POST
  3. Body:
     1. **width**: length of table; number > 0.
     2. **height**: breadth of table; number > 0.
     3. **capacity**: the amount of people this table can accommodate; number > 0.
     4. **number**: number of chairs to add; int > 0; default 1.
     5. **type**: custom table type; string of length > 0; default none.
* createRestaurant(name, password, width, height, padding)
  1. URL: */restaurants/*
  2. Method:POST
  3. Body:
     1. **name**: the name of this restaurant; string of length > 0.
     2. **password**: the password to access this restaurant.
     3. **width**: length of restaurant; number > 0.
     4. **height**: breadth of restaurant; number > 0.

\* initial padding is automatically calculated as 2% of max(width, height)

* getAllRestaurants()
  1. URL: */restaurants/*
  2. Method:GET
* getRestaurantByID(restID)
  1. URL: */restaurants/<int:rest\_id>/*
  2. Method:GET
* getReservationsByRestaurant(restID)
  1. URL: */restaurants/<int:rest\_id>/reservations/*
  2. Method:GET
* getChairByID(chrID)
* getGroupByID(grpID)
* getObjectionByID(objID)
* getSittingAreaByID(saID)
* getTableByID(tbID)
* deleteRestaurantByID(restID)
  1. URL: */restaurants/<int:rest\_id>/*
  2. Method:DELETE
* removeChairFromRestaurant(chrID)
  1. URL: */restaurants/<int:rest\_id>/chairs/*
  2. Method:DELETE
  3. Body:
     1. **ids**: list of ids of chairs to delete; must be a valid list of chair ID ints.
* removeGroupFromRestaurant(groupID)
  1. URL: */restaurants/<int:rest\_id>/groups/<int:grp\_id>/*
  2. Method:DELETE
* removeObjectionFromRestaurant(objID)
  1. URL: */restaurants/<int:rest\_id>/objections/<int:obj\_id>/*
  2. Method:DELETE
* removeReservationFromRestaurant(resID)
  1. URL: */restaurants/<int:rest\_id>/reservations/<int:res\_id>/*
  2. Method:DELETE
* removeSeatingAreaFromRestaurant(saID)
  1. URL: */restaurants/<int:rest\_id>/sittingAreas/<int:sa\_id>/*
  2. Method:DELETE
* removeTableFromRestaurant(tbID)
  1. URL: */restaurants/<int:rest\_id>/tables/*
  2. Method:DELETE
  3. Body:
     1. **ids**: list of ids of table to delete; must be a valid list of table ID ints.
* updateGroupByID(grpID, topLeft, padding, orientation, saID)
  1. URL: */restaurants/<int:rest\_id>/groups/<int:grp\_id>/*
  2. Method:POST
  3. Body:
     1. **topLeft**: list representing coordinates of this group with respect to origin at the top left of the restaurant in the format [xmin, ymin, xmax, ymax]; valid list of numbers; default None.
     2. **padding**: extra spacing around the table and chair set-up; number > 0; default None.
     3. **orientation**: the orientation of this table; either “vertical” or “horizontal”; default None.
     4. **sittingArea**: the ID of the sitting area this group is currently in; if not placed i.e. if coords = None or None in coords, this is None; else it must be a valid sittingArea ID int; default None.
* updateObjectionByID(objID, coords)
  1. URL: */restaurants/<int:rest\_id>/objections/<int:obj\_id>/*
  2. Method:POST
  3. Body:
     1. **coords**: list representing coordinates of this objection with respect to origin at the top left of the restaurant in the format [xmin, ymin, xmax, ymax]; valid list of numbers; default None.
* updateReservationByID(resID, duration)
  1. URL: */restaurants/<int:rest\_id>/reservations/<int:res\_id>/*
  2. Method:POST
  3. Body:
     1. **duration**: the duration in minutes of this reservation; int > 0; default None.
* updateRestByID(restID, name, password, width, height, padding)
  1. URL: */restaurants/<int:rest\_id>/*
  2. Method:POST
  3. Body:
     1. **name**: the name of this restaurant; string of length > 0; default old name.
     2. **password**: the password to access this restaurant; default old password.
     3. **width**: length of restaurant; number > 0; default old width.
     4. **height**: breadth of restaurant; number > 0; default old height.
     5. **padding**: average spacing around table and chair settings in this restaurant; number > 0; default old padding.
* updateSittingAreaByID(saID, coords)
  1. URL: */restaurants/<int:rest\_id>/sittingAreas/<int:sa\_id>/*
  2. Method:POST
  3. Body:
     1. **coords**: list representing coordinates of this sittingArea with respect to origin at the top left of the restaurant in the format [xmin, ymin, xmax, ymax]; valid list of numbers; default None.