# Super Slice Online Ordering System API Docs

## User Services

These are all the services available on the customer page. Employees can also use these services, so this was left as user services.

### Log-in

Queries a user based on user and password. These are passed as parameters.

If there is a match for username & password, it will return the user object.

Otherwise, this will return null and should be handled in the front end.

|  |
| --- |
| **Request method**: GET  **URI**: /query/user?user=*jojo*&password=*password* |
| **Successful Login Return**  {  "userId": 36,  "username": "jojo",  "password": "cpassword",  "userRole": {  "roleId": 2,  "roleTitle": "Customer"  },  "tickets": []  } |
| **Unsuccessful Login Return**  null |

### Register

Submits a username & password to attempt a registration. These should be passed in the body.

If registration is successful, will return the newly created user object.

Otherwise, will return a new user with id of 0 and null values for all fields.

|  |
| --- |
| **Request method**: POST  **URI**: /register |
| **BODY FORMAT JSON**  {"username":"jojo","password":"cpassword"} |
| **Successful Register Return**  {  "userId": 37,  "username": "jojo",  "password": "cpassword",  "userRole": {  "roleId": 2,  "roleTitle": "Customer"  },  "tickets": []  } |
| **Unsuccessful Register Return**  {  "userId": 0,  "username": null,  "password": null,  "userRole": null,  "tickets": []  } |

## 

### Make Order

Submits a special OrderForm object to create a ticket & all pizzas within.

OrderForm format was created to simplify the front-end work for placing an order.

OrderForm object format shown below and should be placed in the body as JSON.

See “Index” for a list of possible “type”, size, and “toppingNames.”

(Note, toppingNames should only populate for CustomPizza type, else leave it empty)

|  |
| --- |
| **Request Method**: POST  **URI**: /makeOrder |
| **BODY FORMAT JSON**  {  "username": "StarPlatinum",  "note": "Freeze time before opening fence.",  "pizzaForms": [  { "type": "CustomPizza", "size": "Small", "cost":11.00 , "quantity":2,  "toppingNames":["Chicken","Pineapples"]},  { "type": "AlfredoPizza", "size": "Medium", "quantity":1 ,"cost":15.00}  ]  } |
| **Ticket Return**  {  "ticketId": 53,  "placementTime": "18:49:10",  "status": "Pending",  "note": "Freeze time before opening fence.",  "totalCost": 26.0,  "placementDate": "2020-05-08",  "pizzas": [],  "user": {  "userId": 53,  "username": "StarPlatinum",  "password": "cpassword",  "userRole": {  "roleId": 2,  "roleTitle": "Customer"  }  }  } |

### Get Tickets By User

Gets a list of all tickets, ordered by the ticket ID (therefore ordered by time) based on each user. Simply give the username of the user you wish to get the tickets for, and a list will be returned of the ticket type. Insert the username in the pathvariable after /ticket/ as shown below.

|  |
| --- |
| **Request Method:** GET  **URI:** /ticket/*mohammed25* |
| **Tickets Return: (Pizzas after first truncated)**  [  {  "ticketId": 65,  "placementTime": "19:43:49",  "status": "Pending",  "note": "Deliver at 7PM please.",  "totalCost": 48.0,  "placementDate": "2020-05-09",  "pizzas": [  {  "pizzaId": 115,  "type": "MediterraneanPizza",  "toppings": [  {  "toppingId": 2,  "toppingName": "Medium",  "cost": 10.0  },  {  "toppingId": 14,  "toppingName": "Black Olives",  "cost": 1.0  },  {  "toppingId": 16,  "toppingName": "Tomatoes",  "cost": 1.0  }  ]  }..],  "user": {  "userId": 8,  "username": "mohammed25",  "password": "cpassword",  "userRole": {  "roleId": 2,  "roleTitle": "Customer"  }  }  }  ] |

### Contact Us: Email

Submit a body that includes username, emailAddress, and comments as shown below.

An email will be sent to the Super Slice official email address, which may then be viewed and replied to by an employee who has access to the email. There is no HTTP return.

|  |
| --- |
| **Request Method:** POST  **URI:** /sendEmail |
| **BODY FORMAT JSON:**  {  "username":"jojo",  "emailAddress":"starplatinum@gmail.com",  "comments":"Hey, I ordered 20 pizzas and they haven't showed up yet!!!!"  } |
| **NO RETURN** |
| **Email sent to SS Official Email Address:** |

## Employee Services

These are used in the employee page exclusively. Employee login is handled by user login.

### Get Ticket By ID

Pass in a ticket ID as a path variable. Returns ticket with filled out pizza objects.

|  |
| --- |
| **Request Method:** GET  **URI**:/ticket/*35* |
| **Successful Ticket Return**  {  "ticketId": 36,  "placementTime": "15:38:31",  "status": "Pending",  "note": "Punch door down on arrival please.",  "totalCost": 26.0,  "placementDate": "2020-05-08",  "pizzas": [  {  "pizzaId": 87,  "toppings": [  {  "toppingId": 30,  "toppingName": "Small",  "cost": 8.0  },  {  "toppingId": 11,  "toppingName": "Onions",  "cost": 1.0  },  {  "toppingId": 10,  "toppingName": "Mushrooms",  "cost": 1.0  }  ]  }  ],  "user": {  "userId": 36,  "username": "jojo",  "password": "cpassword",  "userRole": {  "roleId": 2,  "roleTitle": "Customer"  }  }  } |
| **Unsuccessful Ticket Return**  404 |

### Get Ticket By Status

Pass in one of a number of status options as parameter: Pending, Incomplete, Complete.

Returns a list of all tickets following status request, along with ALL FULL PIZZA objects.

|  |
| --- |
| **Request Method:** GET  **URI:** /ticket?status=*Pending* |
| **Successful Ticket Returns (Pizzas truncated)**  [  {  "ticketId": 55,  "placementTime": "19:16:40",  "status": "Pending",  "note": "Freeze time before opening fence.",  "totalCost": 26.0,  "placementDate": "2020-05-08",  "pizzas": [..],  "user": {  "userId": 53,  "username": "StarPlatinum",  "password": "cpassword",  "userRole": {  "roleId": 2,  "roleTitle": "Customer"  }  }  }  ] |
| **Sample embedded Pizza object (Toppings truncated)**  "pizzas": [  {  "pizzaId": 85,  "toppings": [..]  },  {  "pizzaId": 86,  "toppings": [..]  }  ] |

### Update Ticket

Pass in the ticket ID and the new status as parameters to update the ticket status.

The return will be the ticket item with pizza objects filled out.

|  |
| --- |
| **Request Method**: PUT  **URI:** /update?id=*36*&status=*Cooking* |
| **Successful Ticket Return**  {  "ticketId": 36,  "placementTime": "15:38:31",  "status": "Cooking",  "note": "Punch door down on arrival please.",  "totalCost": 26.0,  "placementDate": "2020-05-08",  "pizzas": [  {  "pizzaId": 87,  "toppings": [  {  "toppingId": 10,  "toppingName": "Mushrooms",  "cost": 1.0  },  {  "toppingId": 11,  "toppingName": "Onions",  "cost": 1.0  },  {  "toppingId": 30,  "toppingName": "Small",  "cost": 8.0  }  ]  }  ],  "user": {  "userId": 36,  "username": "jojo",  "password": "cpassword",  "userRole": {  "roleId": 2,  "roleTitle": "Customer"  }  }  } |
| **Unsuccessful Ticket Return**  505 |

## Pizza Services

This is a service for the web page to access, getting general information that isn’t necessarily tied to a user or employee.

### Get Meat/Veggie Toppings & Sizes

Returns a list format of all toppings, which includes id, name, cost, and a list of all pizza IDs with said topping. Sizes get stored as a topping at the moment and are handled the same way.

|  |
| --- |
| **Request Method:** GET  **URI:** /getMeatToppings OR /getVeggieToppings OR /getSizes |
| **Successful Toppings Return (Only first two on list, meat)**  [  {  "toppingId": 22,  "toppingName": "Sausage",  "cost": 2.0,  "pizzas": [  {  "pizzaId": 82  },  {  "pizzaId": 14  },  {  "pizzaId": 48  }  ]  },  {  "toppingId": 23,  "toppingName": "Bacon",  "cost": 2.5,  "pizzas": [  {  "pizzaId": 82  },  {  "pizzaId": 14  },  {  "pizzaId": 18  }  ]  }  ] |

### Get Pizza By Type

Gets a blank pizza template for a pizza of a certain type. Used to get a list of toppings for each pizza type. As such, pass pizza names as a parameter of “toppings.”

See index for a list of all possible pizza types to pass.

|  |
| --- |
| **Request Method:** GET  **URI:** /pizza?toppings=*WhitePizza* |
| **Successful Toppings Return**  {  "pizzaId": 0,  "ticket": {  "ticketId": 0,  "placementTime": null,  "status": null,  "note": null,  "totalCost": 0.0,  "placementDate": null,  "pizzas": [],  "user": null  },  "type": "WhitePizza",  "toppings": [  null,  {  "toppingId": 19,  "toppingName": "Roasted Cauliflower",  "cost": 1.0  },  {  "toppingId": 35,  "toppingName": "Parmesan Cheese",  "cost": 1.0  },  {  "toppingId": 36,  "toppingName": "Ricotta Cheese",  "cost": 1.0  },  {  "toppingId": 37,  "toppingName": "White Sauce",  "cost": 1.0  }  ]  } |

## Metrics Service

This is a service used to retrieve the metric data used for the metrics/statistics page. Although it is in the Employee page, it is information used more for management rather than the general employee. As such, these are implemented in its own Service/Controller layer in the backend.

### Get Orders By Time

Gets an array of type int/number with an index for each hour. Each index represents an hour of the day, and the number in the index represents the amount of tickets with an order time of that hour. Simple GET request.

|  |
| --- |
| **Request Method:** GET  **URI:** /metrics/time |
| **Successful Array Return:**  [  0,  0,  1,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  6,  2,  0,  1,  0,  0  ] |

### Get Top 5 Toppings

Gets the top five ordered toppings by pizza. The return is a map with the topping name as the key and the amount of pizzas with that topping as the value. Alternate URIs were also created to get the topping names and topping amounts separately as arrays for simpler handling in the frontend.

|  |
| --- |
| **Request Method:** GET  **URI:** /metrics/toptoppings |
| **Successful Map Return:**  {  "Chicken": 19,  "Pepperoni": 15,  "Onions": 11,  "Roasted Cauliflower": 10,  "Broccoli": 10  } |
| **URI for names:** /metrics/topping/names  **URI for amount:** /metrics/topping/amounts |

### Get Top 3 Customers

Gets the top 3 customers based on the amount of orders made. Two separate arrays are retrieved using two URIs for ease of use for the front end.

|  |
| --- |
| **Request Method:** GET  **URI for usernames:** /metrics/customer/names  **URI for order amounts:** /metrics/customer/orders |
| **Username return:**  [  "schaud24",  "StarPlatinum",  "MrPower"  ]  **Order amount return:**  [  9,  3,  2  ] |

### Get Total Revenue

This is a simple service used to get the total revenue between all orders from the very beginning. Returns a number (int) for the total amount.

|  |
| --- |
| **Request Method:** Put  **URI:** /metrics/earnings/total |
| **Int Return:**  546 |

### Get Average Ticket Price

This is a simple service used to get the average price across all tickets. Returns a number (int) for the average of all ticket prices.

|  |
| --- |
| **Request Method:** Put  **URI:** /metrics/earnings/average |
| **Int Return:**  27 |

## Index

### Sizes

|  |  |  |
| --- | --- | --- |
| Small | Medium | Large |

### Pizza Types

|  |  |  |  |
| --- | --- | --- | --- |
| PlainPizza | WhitePizza | VeggiePizza | MeatLoversPizza |
| HawaiianPizza | MediterraneanPizza | FourCheesePizza | AlfredoPizza |
| ItalianPizza | SupremePizza | PepperoniPizza | MushroomPizza |
| ChickenPizza | AnchoviePizza | SpinachPizza | BaconPizza |
| CauliflowerPizza | SausagePizza | EggplantPizza | CustomPizza |

### Toppings

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mushrooms | Onions | Green Peppers | Red Peppers | Black Olives |
| Jalapeños | Tomatoes | Broccoli | Roasted Eggplant | Roasted Cauliflower |
| Extra Cheese | Pineapples | Sausage | Bacon | Chicken |
| Meatballs | Proscuitto | Anchovies | Ham | Pepperoni |
| Spinach | Feta Cheese | Parmesan Cheese | Ricotta Cheese | White Sauce |

## API for General SSOOS Services

### Create Ticket

POST: api/ticket

### Read Ticket By ID

GET: api/ticket/id/{number}

### Read All Tickets

GET: api/ticket/all

### Read Tickets By Username

GET: api/ticket/username/{username}

### Update Ticket

PUT: api/ticket

### Delete Ticket

DELETE: api/ticket

### Ticket Sample JSON

|  |
| --- |
| {  "ticketId": 55,  "placementTime": "19:16:40",  "status": "Complete",  "note": "Freeze time before opening fence.",  "totalCost": 26.0,  "placementDate": "2020-05-08",  "pizzas": [  {  "pizzaId": 103,  "type": "CustomPizza",  "toppings": [  {  "toppingId": 1,  "toppingName": "Small",  "cost": 8.0  },  {  "toppingId": 24,  "toppingName": "Chicken",  "cost": 2.0  }  ]  }  ],  "user": {  "userId": 53,  "username": "StarPlatinum",  "userRole": {  "roleId": 2,  "roleTitle": "Customer"  },  "tickets": [  {  "ticketId": 55,  "placementTime": "19:16:40",  "status": "Complete",  "note": "Freeze time before opening fence.",  "totalCost": 26.0,  "placementDate": "2020-05-08"  },  {  "ticketId": 57,  "placementTime": "19:37:44",  "status": "Ready",  "note": "Freeze time before opening fence.",  "totalCost": 26.0,  "placementDate": "2020-05-08"  }  ]  }  } |

### 

### Create Topping

POST: api/topping?name={name}&cost={cost}

### Read Topping By ID

GET: api/topping/id/{id}

### Read Topping By Name

GET: api/topping/name/{name}

### Read All Toppings

GET: api/topping/all?order=id

### Read All Toppings Ordered By Popularity

GET: api/toppings/all?order=popularity

### Update Topping

PUT: api/topping

### Delete Topping

DELETE: api/topping

### Sample Topping JSON

|  |
| --- |
| {  "toppingId": 20,  "toppingName": "Extra Cheese",  "cost": 1.0,  "pizzas": [  {  "pizzaId": 118,  "type": "SupremePizza"  },  {  "pizzaId": 119,  "type": "SupremePizza"  },  {  "pizzaId": 120,  "type": "CustomPizza"  },  {  "pizzaId": 129,  "type": "CustomPizza"  }  ]  } |