Note: The highlighted portions represent routes completed during different sprints

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Routes** | **Purpose** | **What is sent to the backend?** | **What is sent to the frontend?** | **Method** | **Exceptions** |
| Authentication | | | | | |
| /auth/register  Pat  TESTED | The page where managers can register into the system in the beginning | -email: string  -name: string  -password: string  -resturant\_name: string  -location: string  Eg.  {email: [email@email.com](mailto:email@email.com), password: password,  name: name,  resturant\_name: name,  location: location} | -{manager\_id: integer,  Menu\_id: integer,  Category\_id: str(int),  (of the ‘Best Selling Category’)  token: String (Later Sprint)  } Success JSON Object  - {‘error’: ‘invalid’} JSON Object | POST | InputError  -Email already taken |
| /auth/login  Jibi  TESTED | The page where managers, kitchen staff and wait staff can login to the system | -email: string  -password: string  Eg.  {email: [email@email.com](mailto:email@email.com), password: password,  } | {staff\_id: integer, menu\_id: integer,  Staff\_type: String { ‘manager’ | ‘wait’ | ‘kitchen | invalid’ }  Token: String (Later Sprint)}  Success JSON Object  - {error: <reason>} Fail JSON Object | POST | AccessError  -Invalid email  -Incorrect password |
| /auth/logout  Pat  TESTED | The route for logging staff out of the system | -staff\_id: integer | Success JSON  { ‘success’: ‘successfully logged out’  }  Fail JSON  { ‘error’: ‘invalid staff\_id’  } | POST | -invalid staff\_id |
| Customer | | | | | |
| customer/menu/search  Pat | The page where customers can search which restaurant’s menu they want to access | -query: string  -session\_id | Returns JSON list of JSON Objects containing:  [{ menu\_id: integer,  restaurant\_name: string,  restaurant\_address: string},  { menu\_id: integer,  restaurant\_name: string,  restaurant\_address: string}, …{same as above}] | GET | (Handled by returning empty list) |
| /customer/view\_menu  Jibi | The page which displays the home page menu of a particular restaurant | -session\_id: Str  -menu\_id: integer  (Note: session\_id should be created in the frontend)  -allergies: Array(int)  Eg:  /customer/view\_menu?session\_id=........  &allergies=[1,3,4]  -?excluded\_cat\_ids: list of int category ids, for example  (optional, to be used only when category\_id is of a ‘Best Selling’ Category  “....&excluded\_cat\_ids=[2,34] | JSON: Returns categories and list of menu items in each category  Eg. [  { category\_id (Eg 1):[  ‘Best Selling’,   [  {food\_id: 123, food\_name: Burger,  Food\_description: ‘2 pieces of bread’  Food\_image: string,  Food\_price: float  Food\_ordering\_id: int  food\_ingredients  },  {food\_id: 234, food\_name: Fries,  Food\_description: ‘Potato’  Food\_image: string,  Food\_price: float,  Food\_ordering\_id: int  },  ],  Category\_ordering\_id: int  ]},  {2: [‘Chef’s Choice’, [], Category\_ordering\_id: int]},  {3: [‘Cat1’, [],  Category\_ordering\_id: int  ]},  {4: [‘Category 3’, [],  Category\_ordering\_id: int  ]}  ]  Or failure object:  {‘error’: ‘menu\_id does not exist’} | GET | AccessError  -If menu\_id is non-existent |
| /customer/view\_category  Jibi | When a customer clicks on a different category, this route gives a list of all menu items under that category | -session\_id: str  -category\_id: str(int)  -allergies: Array(int)  Eg:  /customer/view\_category?session\_id=........  &allergies=[1,2] | [   {food\_id, food\_name, food\_description, food\_image, food\_price,  food\_ordering\_id: int},  {food\_id, food\_name, food\_description,  Food\_image,  food\_price,  food\_ordering\_id: int}  ] | GET | -invalid category\_id  -invalid session\_id |
| /customer/view\_menu\_item  Jibi | When a customer clicks ‘Find out more’ on the list of items page, this route provides the info for what should be displayed on that new screen | -session\_id : str  -menu\_item\_id: str(int) | Success JSON  {food\_id, food\_name, food\_description, food\_image, food\_price, food\_ingredients: str,  Category\_id: int,  Food\_ordering\_id: int,}  Failure JSON  { ‘error’: ‘invalid menu\_item\_id’ | GET | -invalid menu\_item\_id  -invalid session\_id |
| /get\_allergies | The route for the frontend to receive a list of allergies for multiple reasons:  1)To show the list of allergy options on the personalise the customer page  2)To add allergies along with certain ingredients during /manager/add\_menu\_item or /manager/update\_menu\_item  3) for /customer\_view\_{menu | category} filter out foods with specified allergies |  | Success returns:  [  # [allergy\_id: int, #allergy\_name: string, #allergy\_description: #string]   [1, ‘Peanuts’, ‘Specifically for peanuts’],  [2, ‘Tree Nuts’, ‘includes a variety of nuts like walnuts, almonds and cashews],  [3, ‘Dairy’, ‘includes milk, cheese, yoghurt and other dairy products], …  ]  Failure:  { ‘error’: ‘invalid session\_id’ } | GET | -invalid session\_id |
| /customer/menu/table  Rohan  DONE | The route for the customer to add the table number | -table\_id: None/ str(integer)  -menu\_id: integer  -session\_id: str(int) | {table\_id: str(int)} Success  {'error': 'invalid table\_id' } Failure | POST | Invalid Table Id  -Non-existent table number |
| /customer/add\_menu\_item  Rohan  Pat  Jibi | This route adds the menu item that the customer wants in their order | -session\_id: str(int)  -menu\_id: str(int)  -menu\_item\_id: str(int)  -amount: integer  -persona\_name: string  Eg.  {  menu\_id: ‘1’  session\_id: ‘12’,  menu\_item\_Id: ‘1’,  amount: ‘1’,  title: ‘Burger’} | success  {  ‘Session\_id’: str(int),  ‘Table\_id’: str(int),  ‘Menu\_id’: str(int),  ‘Menu\_items’: [    { ‘menu\_item\_id’: str(int),      ‘title’: string,      ‘amount’: str(int),      ‘price’: int,      ‘image’: str,      ‘description’: str,      ‘persona’: str    },    { …(Same as above)    }  ]  }  Failure:  {'error': 'invalid session\_id' } | POST | AccessError  -If food\_id is non-existent  -If session\_id is non-existent |
| /customer/remove\_menu\_item  Rohan  Jibi | This route removes the menu item from the customers order | -session\_id: str(int),  -menu\_id: str(int)  -menu\_item\_id: str(int),  -amount: integer  -persona\_name: string  Eg.  {menu\_Id: 1,  Table\_id: 3  menu\_item\_Id: 1,  amount: 1} | Success  {  ‘Session\_id’: str(int),  ‘Table\_id’: str(int),  ‘Menu\_id’: str(int),  ‘Menu\_items’: [    { ‘menu\_item\_id’: str(int),      ‘Title’: string,      ‘Amount’: str(int)    },    { …(Same as above)    }  ]  }  Failure:  {'error': 'invalid session\_id' , | DELETE | AccessError  -If food\_id is non-existent  -If session\_id is non-existent  InputError  -If amount is higher than the actual amount |
| /customer/view\_order  Rohan  DONE | This route returns order details from the backend | -session\_id: str(int)  -menu\_id: str(int) | Returns a JSON order:  {  ‘Session\_id’: str(int),  ‘Table\_id’: str(int),  ‘Menu\_id’: str(int),  ‘Status’: ‘customer’ | ‘kitchen’ | ‘cooking’ | ‘wait’ | ‘serving’,  ‘Timestamp’: str(datetime)  ‘Menu\_items’: [    { ‘menu\_item\_id’: str(int),      ‘menu\_item\_id: str(int)      ‘title’: string,      ‘amount’: str(int),      ‘price’: int,      ‘image’: str,      ‘description’: str    }  ]  }  Failure:  {'error': 'invalid session\_id' } | GET | AccessError  -If session\_id is non-existent  -If menu\_id is non-existent |
| /customer/finalise\_order  Jibi | This route confirms to the backend that  the current selection of menu\_items for the given session\_id is to be sent to the kitchen staff for cooking | -session\_id: str(int),  -menu\_id:  str(int) | { ‘success’: ‘order finalised’ }  { ‘error’: ‘error in finalising order’ } | POST | -invalid session\_id |
| /customer/request\_assistance  Pat | Customer requests assistance from wait staff | -table\_id: str(integer)  -session\_id:  str(integer)  -menu\_id:  str(integer) | { ‘success’: assistance requested’ }  { ‘error’: ‘error in requesting assistance’ } | POST |  |
| /customer/give\_rating | Lets the customer give a rating value from 1 to 5 inclusive for a given menu\_item | -session\_id: str(int)  -menu\_item\_id: str(int)  -rating: str(int)  (from 1 to 5)  -amount: str(int) |  | POST |  |
| Manager | | | | | |
| /manager/view\_menu  Jibi  DONE | The base menu screen accessible to the manager where the manager can see a list of menu items from the default category (Best Selling), and add, remove, or update categories or menu items of respectively | -menu\_id: integer  - manager\_id: integer  -?excluded\_cat\_ids: list of int category ids, for example  (optional, to be used only when category\_id is of a ‘Best Selling’ Category  “....&excluded\_cat\_ids=[2,34]  -?top\_k: int  (optional argument, get top k items from the category) | JSON list eg:  [   {  Category\_id (string):[  ‘Best Selling’,   [   {food\_id, food\_name, food\_description, food\_image, food\_price,  Food\_ingredients,  food\_ordering\_id},  {food\_id, food\_name, food\_description,  Food\_image,  Food\_price,  Food\_ingredients,  food\_ordering\_id}}  ],  Category\_ordering\_id: str(int)  ]  },  {category\_id1: [ ‘Category 1’, [], Category\_ordering\_id: str(int)  ] }  {category\_id2: [ ‘Category 2’, [],  Category\_ordering\_id: str(int)  ] }  ] | GET |  |
| /manager/view\_category  Jibi  DONE | Lists all the menu items within a given category  Used when the user clicks on a different category to list its contents | -manager\_id: integer  -category\_id: int  -?top\_k: int  (optional argument, get top k items from the category) | [   {food\_id, food\_name, food\_description, food\_image, food\_price,  Food\_ingredients,  food\_ordering\_id},  {food\_id, food\_name, food\_description,  Food\_image,  Food\_price,  Food\_ingredients,  food\_ordering\_id}  ] | GET |  |
| /manager/add\_staff  Jibi  DONE | The route for adding new staff given a manager\_id | {Manager\_id: string,  Email: string,, password: string, staff\_type: string { kitchen | wait },  Name: string,  Menu\_id: str(int)  }  Eg.  {email: [email@email.com](mailto:email@email.com), password: password,  Staff\_type: waitStaff  } | { ‘success’ : string,  ‘staff\_id’: string,  } Success JSON  { ‘error’: <reason>} Failure JSON | POST |  |
| /manager/view\_menu\_item  Jibi  DONE | Manager views information about a particular food item | -manager\_id:  -menu\_item\_id: str(integer) | Success JSON  {food\_id, food\_name, food\_description, food\_image, food\_price, food\_ingredients: str,  Category\_id: int,  Food\_ordering\_id: }  Failure JSON  { ‘error’: ‘invalid menu\_item\_id’ | GET |  |
| /manager/add\_menu\_item  Rohan  Jibi | Manager adds a food item to certain category | {  Manager\_id: string,  title: ‘Bread’,  Price: 12.50  ingredients:  [  # #ingredient\_nam#e: string,  #Allergy\_id: int | #null  [‘Potatoes’, null],  [‘Eggs’, 5],  [‘Milk’, 3]  ]  ,  Description: ‘Sample Descr’,  Category\_id: int,  menu\_id: int  } | {menu\_item\_id: str(int)} | POST |  |
| /manager/delete\_menu\_item  Rohan  DONE | Manager deletes food from the menu | {Manager\_id:  menu\_item\_id: integer} | { ‘error’: ‘invalid menu\_item\_id’}  Or  { ‘success’: menu item deleted } | DELETE |  |
| /manager/update\_menu\_item  Rohan  Jibi | Manger updates food from menu | {Manager\_id: str(int)  menu\_item\_id: str(integer),  title: ‘Bread’,  Price: 12.50  ingredients:  [  # #ingredient\_nam#e: string,  #Allergy\_id: int | #null  [‘Potatoes’, null],  [‘Eggs’, 5],  [‘Milk’, 3]  ]  ,  Description: ‘Sample Descr’,  image: string,  Category\_id: int,  menu\_id: int  Image: str} | * menu\_item\_id | POST |  |
| /manager/add\_category  Pat | Manager adds a food category to the menu | {manager\_id: int  menu\_id: integer  category\_name: ‘Example’ } | { category\_id: int } | POST |  |
| /manager/delete\_category  Pat  DONE | Manager deletes a food category from the menu | {Manager\_id: int  Menu\_id: integer, category\_id: int} | Failure JSON:  { 'error': 'invalid category'}  Success JSON:  { 'success': 'success in removing category' } | DELETE |  |
| /manager/update\_category  Pat  DONE | Manager updates category information eg. category\_name,  category\_order\_id, | {Manager\_id:  Category name: ‘Example’,  Category\_id: int } | * category \_id | POST |  |
| /manager/update\_category\_ordering  Pat | Route which updates the ordering of categories in the database | {‘Manager\_id’:  ‘Category\_id’: int,  ‘prev\_ordering\_id’: str(int)  ‘new\_ordering\_id’:  str(int)} | Failure JSON:  { 'error': 'invalid category'}  Success JSON:  { 'success': 'success in updating category ordering' } | POST |  |
| /manager/update\_menu\_item\_ordering  Pat | Route which updates the ordering of menu items in the database | {‘Manager\_id’: str(int),  ‘menu\_item\_id’: str(int),  ‘prev\_ordering\_id’: str(int),  ‘New\_ordering\_id’: str(int) | Failure JSON:  { 'error': 'invalid category'}  Success JSON:  { 'success': 'success in updating order of menu items in category' } | POST |  |
| Kitchen Staff | | | | | |
| /kitchen\_staff/get\_order\_list  Pat | Gets a time-sorted list of customer orders with ‘status’ = ‘kitchen’  Or (‘status’ = ‘cooking’ and this is the current chef who is cooking it) | -kitchen\_staff\_id: str(int)  -menu\_id: str(int) | Returns ordered list of JSON objects  [   {‘session\_id’: str(int),   ‘table\_id’: int,   ‘Menu\_items’: []   ‘status’: ‘kitchen’,  ‘Timestamp’: str(datetime)   },   {‘session\_id’: str(int),   ‘table\_id’: int,   ‘Menu\_items’: []   ‘status’: ‘cooking’,  ‘Timestamp’: str(datetime)   },  ]  {  ‘menu\_items’: [   food\_id: integer,   food\_name: string,  Image: string,   amount: string,  ]  } | GET |  |
| /kitchen\_staff/mark\_currently\_cooking  Jibi | Marks and order item as being cooked currently, so it disappears from the list of other kitchen staffs’ order list,  Sets ‘status’ = ‘cooking’ | -menu\_id: str(int)  -session\_id: str(int)  -kitchen\_staff\_id: str(int) | Success  Or  failure | POST |  |
| /kitchen\_staff/mark\_order\_complete  Pat | Marks an order item as prepared once it is ready for serving, and mark ‘status’ = ‘wait’  Also updates the timestamp when the cooking is finished | -menu\_id: str(int)  -session\_id: str(int) | Returns  success | POST |  |
| Wait Staff | | | | | |
| /wait\_staff/get\_order\_list  Rohan | Gets a time-sorted list of customer orders with ‘status’ = ‘wait’  Or (‘status’ = ‘serving’ and order[‘wait\_staff\_id’] = serving staff’s id) | -menu\_id: str(int)  -wait\_staff\_id: str(int) | List of JSON objects  [{ ‘session\_id’: str(int),   Table\_number:int,  ‘Status’: ‘wait’ | ‘serving’,  ‘Timestamp’: str(datetime.now())  ‘menu\_items’: [{food\_id:    integer, food\_name:  string,  Image: string},  {food\_id: integer, food\_name: string,  Image: string}  ]  },  …  ] | GET |  |
| /wait\_staff/mark\_currently\_serving | Marks and order item as being served currently, so it disappears from the list of other wait staffs’ order list,  Sets ‘status’ = ‘serving’ | -menu\_id: str(int)  -session\_id: str(int)  -wait\_staff\_id: str(int) | Success  Or  failure | POST |  |
| /wait\_staff/mark\_order\_complete  Rohan | Removes order  From orders[menu\_id] list in backend | -menu\_id: str(int)  -session\_id: str(int) |  | DELETE |  |
| /wait\_staff/get\_assistance\_notifications  Pat | Gets a list of notifications from a list from the backend | -menu\_id  -wait\_staff\_id: str(int) | List of json objects  [{  session\_id: str(int),  table\_id: str(int)},  ‘Status’: ‘customer’,  ‘Timestamp’: str(datetime.now())  { ‘session\_id’: ‘122333’,  ‘Table\_id’: ‘12’,  ‘Status’: ‘customer’,  ‘Timestamp’: str(datetime.now())  }]  Failure  [] | GET |  |
| /wait\_staff/mark\_currently\_assisting  Jibi | Mark a particular notification as being addressed currently by a waiter so it does not appear on other wait staff notifications’ screens | -menu\_id: str(int)  -table\_id: str(int)  -session\_id: str  -wait\_staff\_id: str(int) | Success  { ‘success’: ‘marked currently assisting’ }  Failure  { ‘error’: ‘internal error’ } | POST |  |
| /wait\_staff/mark\_notification\_complete  Rohan | Removes a notification from the list in the backend notifications | -table\_id  -session\_id  -menu\_id |  | DELETE |  |