**MUHAMMAD HARRIZ AZHAM | PROJECT 2**

**Task 1**

|  |  |  |
| --- | --- | --- |
| Items | | |
| Primary Key | id | Int |
|  | name | Text |
|  | price | Decimal |
| Foreign Key | category\_id | Int |

|  |  |  |
| --- | --- | --- |
| Category | | |
| Primary Key | id | Int |
|  | category\_name | text |

Based on Items and Category tables, the Entity Relationship Diagram (ERD) is each type of category can have many products and both tables can be joined by category\_id.

Hence, both tables are created inside MySQL workbench.

**Task 2**

Category Table

Graphical user interface, text, application

Description automatically generated

Items Table

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, application, table

Description automatically generated

Task 3

|  |  |
| --- | --- |
| METHOD/category |  |
| List (List all category) | ○ HTTP method: GET  ○ HTTP Path: /category  ○ Request parameters:   * Header parameter: Token Authentication * Body parameters:  |  |  |  | | --- | --- | --- | | limit | Int (optional) | Display number of categories | | category\_id | Int (optional) | Filtering to categories with only exact category\_id | | sort | String (optional) | Sort the data in ascending order |   ○ Response structure:  {  “object”: “list”,  “url”: “/category”,  “limit”:”15”  “data”: [  {category\_1}, {category\_2},{category\_3}  ]  }  \*each dictionary of category carries some information such as id and category\_name  ○ Status code  200 – it will return a full list of categories  401 – No valid authentication key provided  ○ A request-response example for the happy case:  Request: GET /category with header of token XXXXXX-XXXX-XXXX-XXXX  Response: it will return a list of categories in json format  ○ At least 1 request-response example for the error case:  Api\_error : problem of retrieving the information of category from database server |
| Retrieve | ○ HTTP method: GET  ○ HTTP Path: category/:id  ○ Request parameters:   * Header parameter: Token Authentication * Route parameters: put the index of category at the end of path   ○ Response structure:  {  "id": "1",  "category\_name": "Marvel",  }  ○ Status code (including failure)  200 – return a category at the respective index  401 - No valid authentication key provided  ○ A request-response example for the happy case:  Request: GET /category/2 with header of token XXXXXX-XXXX-XXXX-XXXX  Response: Information about category at index 2 in json format.  ○ At least 1 request-response example for the error case:  Invalid\_request\_error: provide a wrong type of request parameters such as index is not in integer |
| Create | ○ HTTP method: POST  ○ HTTP Path: /category  ○ Request parameters:   * Header parameter: Token Authentication * Body parameters: category\_name   ○ Response structure  {  “object”: “create”,  “url”: “/category”,  “data”: [  {category\_1}, {category\_2},{category\_3},{new category}  ]  }  \*each dictionary of category carries some information such as id and category\_name  ○ Status code (including failure):  200 – it returns a full list of category with the new category row  401 - No valid API key provided.  ○ A request-response example for the happy case:  Request: POST /category with header of token XXXXXX-XXXX-XXXX-XXXX  Response: it returns a full list of category with the new category row in json format  ○ At least 1 request-response example for the error case:  invalid\_request\_error – supply wrong type of value for category\_name in body parameters |
| Update | ○ HTTP method: PUT  ○ HTTP Path: /category/:id  ○ Request parameters:   * Header parameter: Token Authentication * Body parameters: category\_name * Route Parameters: id   ○ Response structure:  {  “object”: “update”,  “url”: “/category/3”,  “data”: [  {category\_1}, {category\_2},{category\_3},{new updated category}  ]  }  \*each dictionary of category carries some information such as id and category\_name    ○ Status code:  200 – successfully updated the category\_name at the respective index  ○ A request-response example for the happy case:  Request: PUT /category/3 with header of token XXXXXX-XXXX-XXXX-XXXX  Response: it returns a full list of category with the new updated category at 3rd row in json format  ○ At least 1 request-response example for the error case:  invalid\_request\_error – the id does not exist in the database |
| Delete | ○ HTTP method: DELETE  ○ HTTP Path: /category/:id  ○ Request parameters:   * Header parameter: Token Authentication * Route parameter: id   ○ Response structure:  {  “object”: “delete”,  “url”: “/category/4”,  “data”: [  {category\_1}, {category\_2},{category\_3}  ]  }  ○ Status code (including failure)  200 – successfully deleted the respective index in category table  402 – the parameter were valid but there is a double authentication for client to delete the data  ○ A request-response example for the happy case  Request: DELETE /category/:id  Response: it returns a full list of categories without the recent deletion  ○ At least 1 request-response example for the error case:  api\_error – the database server is down |

|  |  |
| --- | --- |
| METHOD/items |  |
| List | ○ HTTP method : GET  ○ HTTP Path: /items  ○ Request parameters (route param or query or body)   * Header parameter: Token Authentication * Body Parameter: limit   ○ Response structure:  {  “object”: “list”,  “url”: “/items”,  “limit”:”15”  “data”: [  {items\_1}, {items\_2},{items\_3}  ]  }  \*each dictionary of items carries some information such as id, name, price, category\_id  ○ Status code (including failure):  200 – it will return a full list of items  401 – No valid authentication key provided  ○ A request-response example for the happy case  Request: GET /items with header of token XXXXXX-XXXX-XXXX-XXXX  Response: it will return a list of items in json format  ○ At least 1 request-response example for the error case:  Api\_error : problem of retrieving the information of items from database server |
| Retrieve | ○ HTTP method: GET  ○ HTTP Path: items/:id  ○ Request parameters:   * Header parameter: Token Authentication * Route parameters: put the index of items at the end of path   ○ Response structure:  {  "id": "1",  “name”, “DragonBall”,  “price”:123.65,  "category\_id": 4,  }  ○ Status code (including failure)  200 – return an item at the respective index  401 - No valid authentication key provided  ○ A request-response example for the happy case:  Request: GET /items/2 with header of token XXXXXX-XXXX-XXXX-XXXX  Response: Information about item at index 2 in json format.  ○ At least 1 request-response example for the error case:  Invalid\_request\_error: provide a wrong type of request parameters such as index is not in integer |
| Create | ○ HTTP method: POST  ○ HTTP Path: /category  ○ Request parameters:   * Header parameter: Token Authentication * Body parameters: name, price, category\_id   ○ Response structure  {  “object”: “create”,  “url”: “/items”,  “data”: [  {item\_1}, {item\_2},{item\_3},{new item}  ]  }  \*\*each dictionary of items carries some information such as id, name, price, category\_id  ○ Status code (including failure):  200 – it returns a full list of items with the new item row  401 - No valid API key provided.  ○ A request-response example for the happy case:  Request: POST /items with header of token XXXXXX-XXXX-XXXX-XXXX  Response: it returns a full list of items with the new item row in json format  ○ At least 1 request-response example for the error case:  invalid\_request\_error – supply wrong type of value for name, price or category\_id in body parameters |
| Update | ○ HTTP method: PUT  ○ HTTP Path: /items/:id  ○ Request parameters:   * Header parameter: Token Authentication * Body parameters: name, price, category\_id * Route Parameters: id   ○ Response structure:  {  “object”: “update”,  “url”: “/items/3”,  “data”: [  {item\_1}, {item\_2},{item\_3},{new updated item}  ]  }  \*\*each dictionary of items carries some information such as id, name, price, category\_id    ○ Status code:  200 – successfully updated the name, price and category\_id at the respective index  ○ A request-response example for the happy case:  Request: PUT /item/3 with header of token XXXXXX-XXXX-XXXX-XXXX  Response: it returns a full list of items with the new updated item at 3rd row in json format  ○ At least 1 request-response example for the error case:  invalid\_request\_error – the id does not exist in the database |
| Delete | ○ HTTP method: DELETE  ○ HTTP Path: /items/:id  ○ Request parameters:   * Header parameter: Token Authentication * Route parameter: id   ○ Response structure:  {  “object”: “delete”,  “url”: “/items/4”,  “data”: [  {category\_1}, {category\_2},{category\_3}  ]  }  ○ Status code (including failure)  200 – successfully deleted the respective index in items table  402 – the parameter were valid but there is a double authentication for client to delete the data  ○ A request-response example for the happy case  Request: DELETE /items/4  Response: it returns a full list of categories without the recent deletion  ○ At least 1 request-response example for the error case:  api\_error – the database server is down |

**Task 4**

|  |
| --- |
| /category |
| LIST |
| CREATE |
| RETRIEVE |
| UPDATE |
| DELETE |

|  |
| --- |
| /items |
| LIST |
| RETRIEVE |
| CREATE |
| UPDATE |
| DELETE |

|  |
| --- |
| Validation error, the index must be integer, else error |
|  |

Task 5

|  |  |
| --- | --- |
| CATEGORY/METHOD |  |
| List |  |
| Retrieve |  |
| Create |  |
| Update |  |
| Delete |  |

|  |  |
| --- | --- |
| ITEMS/ METHOD |  |
| List |  |
| Retrieve |  |
| Create |  |
| Update |  |
| Delete |  |

Task 6

Migrating SQL Injection

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Use parameter binding

Graphical user interface, text, application, email

Description automatically generated

