Summary

Routes

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Route | Purpose | Requirements |
| POST | api/auth/customer/register | Register Customer Account | None |
| POST | api/auth/staff/register | Register Staff Account | None |
| POST | api/auth/login | Login | None |
| POST | api/auth/logout | Logout | Auth |
| PUT | api/auth/update | Update Account Details | Auth |
| POST | api/cruise | Create Cruise | Auth + Is Staff |
| GET | api/cruise | View all cruises | None |
| GET | api/cruise/{cruise\_id} | View cruise by numerical id | None |
| GET | api/cruise/{search} | Search cruises by name | None |
| POST | api/book/{cruise\_id} | Make booking by cruise id | Auth |
| DELETE | api/book/{booking\_id} | Cancel booking by booking id | Auth |
| GET | api/book | View all bookings | Auth |
| GET | api/book/{booking\_id} | View booking by booking id | Auth |

Database

Application

Description automatically generated with medium confidence

Notes

Database has been seeded with faker, so some of the data may not make sense (e.g. date)  
Names of Cruises are seeded with names of colors

Postman

- Fail Cases (Auth Middlewares)

- Authentication (api/auth)

- Cruises (api/cruise)

- Bookings (api/book)

Fail Cases (Auth middlewares)

-> Missing header token/header token is ‘’ (Auth requirement)

Response <401>:

{

    "message": "Missing token"

}

-> Invalid token/no users are logged in with the token (Auth requirement)

Request Header -> Token: (non-existent token/any invalid token)

Response <401>:

{

    "message": "Invalid token"

}

-> Token given corresponds to a customer account (Is Staff requirement)

Request Header -> Token: (any valid token that corresponds to a customer account (not staff))  
Response <401>:

{

    "message": "Not staff account"

}

Authentication (api/auth)  
Register Customer Account -> POST api/auth/customer/register  
Request Body -> application/json

{

    "email": "person@example.com",

    "username": "person1203",

    "password": "p@ssw0rd"

}

Response <200>:

{

    "message": "Customer account created"

}

Result: accounts table is inserted with a new account of type ‘customer’ with data in request

Register Staff Account -> POST api/auth/staff/register  
Request Body -> application/json

{

    "email": "staff@example.com",

    "username": "staff1203",

    "password": "p@ssw0rd"

}

Response <200>:

{

    "message": "Staff account created"

}

Result: accounts table is inserted with a new account of type ‘staff’ with data in request

Log into Account-> POST api/auth/login  
Request Body -> application/json

{

    "email": "person@example.com",

    "password": "p@ssw0rd"

}

Response <200>:

{

    "message": "Successfully logged in",

    "token": "T2Sw3c2DFs5XeGyQcdW8G0Jbb1ugIFuA9G9zKHszg7iO55budtq9HZtoFyHl"

}

Result: accounts ‘auth\_token’ column is updated with generated token, returned in response for future authentication via request header ‘Token’

Update Account-> POST api/auth/update (email, password and username can be changed) [Auth required]  
Request Header -> Token: (any valid token that corresponds to an account)

Request Body -> application/json

{

    "password": "n3wp455w0rd"

}

Response <200>:

{

    "message": "Successfully updated account"

}

Result: password is updated to “n3wp455w0rd”

Log out of Account-> POST api/auth/logout [Auth required]  
Request Header -> Token: (any valid token that corresponds to an account)

Response <200>:

{

    "message": "Successfully logged out"

}

Result: ‘auth\_token’ column for corresponding user is updated to ‘’ (token is invalidated)

Cruises (api/cruise)  
Add new Cruise -> POST api/cruise [Auth and Is Staff required]

Request Header -> Token: (any valid token that corresponds to staff account)

Request Body -> application/json

{

    "name": "Richy Cruise",

    "size": "large",

    "owner": "Mr. Richy Guy",

    "capacity": 1200,

    "built\_date": "2022-11-22"

}

Response <200>:

{

    "message": "Successfully added cruise"

}

Result: cruises table is inserted with a new cruise with data given in request

Get all Cruise -> GET api/cruise

Response <200> [dummy data made with faker]:

[

    {

        "id": 1,

        "name": "LightSteelBlue",

        "size": "small",

        "owner": "Adelle Kuhlman",

        "capacity": 769,

        "built\_date": "2016-05-24"

},

… (truncated)

]

Result: All cruises in the ‘cruises’ table are returned

Get a Cruise by id -> GET api/cruise/{cruise\_id} (id must be numerical)

Response <200> (api/cruise/1) [dummy data made with faker]:

{

    "id": 1,

    "name": "LightSteelBlue",

    "size": "small",

    "owner": "Adelle Kuhlman",

    "capacity": 769,

    "built\_date": "2016-05-24"

}

Result: cruise from ‘cruises’ table with corresponding given id is returned

Search Cruises by name -> GET api/cruise/{search} (search is alphanumerical)

Response <200> (api/cruise/al) [dummy data made with faker]:

[

    {

        "id": 2,

        "name": "AliceBlue",

        … (truncated)

    },

    {

        "id": 3,

        "name": "PaleVioletRed",

        … (truncated)

    },

    {

        "id": 4,

        "name": "Coral",

        … (truncated)

    }

]

Result: cruises found from ‘cruises’ table with ‘LIKE’ operator and given search are returned

Bookings (api/book)  
Make booking for cruise by id -> POST api/book/{cruise\_id} [Auth required]

Request Header -> Token: (any valid token that corresponds to an account)

Request Body -> application/json

{

    "price": 50.00,

    "quantity": 4,

    "sailing\_datetime": "2022-08-12 16:34:03"

}

Response <200> (api/book/1):

{

    "message": "Successfully placed booking"

}

Result: ‘bookings’ table is inserted with new booking with given data and linked data with account and cruise id as well as booking\_datetime on the api

Cancel booking by booking id -> DELETE api/book/{booking \_id} [Auth required]

Request Header -> Token: (any valid token that corresponds to an account)

Response <204> (api/book/1)

Result: booking with given booking id is deleted from ‘bookings’ table

View all bookings -> GET api/book [Auth required]

Request Header -> Token: (any valid token that corresponds to an account)

Response <200> (token corresponds to account with id ‘11’) [dummy data made with faker]:

[

    {

        "id": 16,

        "account\_id": 11,

        "cruise\_id": 13,

        "price": 463.34,

        "quantity": 5,

        "sailing\_datetime": "1993-01-22 15:18:00",

        "booking\_datetime": "1975-12-29 13:15:57"

},

… (truncated)

]

Result: All bookings made by the account corresponding to the auth token is returned

View booking by booking id -> GET api/book/{booking\_id} [Auth required]

Request Header -> Token: (any valid token that corresponds to an account)

Response <200> (api/book/171) (token corresponds to account with id ‘11’) [dummy data made with faker]:

{

    "id": 171,

    "account\_id": 11,

    "cruise\_id": 4,

    "price": 697.95,

    "quantity": 1,

    "sailing\_datetime": "1970-06-23 02:38:47",

    "booking\_datetime": "1975-09-19 05:50:01"

}

Result: booking from ‘bookings’ table with corresponding given id is returned