API Documentation for Ride Booking Service

Overview:

The Ride Booking Service API allows users to create, read, update, and delete ride bookings, as well as retrieve information about rides and drivers. The API is built using NodeJS and Express, and uses MongoDB as the database. The API supports CRUD operations for rides, bookings, drivers, and routes.

API Endpoints:

/rides:

- GET: Retrieves all rides
- POST: Creates a new ride

/rides/{id}:

- GET: Retrieves a specific ride by ID
- PUT: Updates a specific ride by ID
- DELETE: Deletes a specific ride by ID

/rides/{id}/bookings:

- GET: Retrieves all bookings for a specific ride by ID

/bookings:

- GET: Retrieves all bookings
- POST: Creates a new booking

/bookings/{id}:

- GET: Retrieves a specific booking by ID
- PUT: Updates a specific booking by ID
- DELETE: Deletes a specific booking by ID

/drivers:

- GET: Retrieves all drivers
- POST: Creates a new driver

/drivers/{id}:

- GET: Retrieves a specific driver by ID
- PUT: Updates a specific driver by ID
- DELETE: Deletes a specific driver by ID

/user:

- GET: Retrieves all user
- POST: Creates a new user

/user/{id}:

- GET: Retrieves a specific user by ID
- PUT: Updates a specific user by ID
- DELETE: Deletes a specific user by ID

Request/Response Formats:

/rides:

GET /rides : This endpoint retrieves all rides.

Request format:
HTTP method: GET
Endpoint: /rides
Parameters: None
Headers: None
Body: None

Response format:
Status code: 200 OK, 404 Not Found otherwise
Headers: Content-Type: application/json
Body: An array of JSON objects, each representing a ride.

Example:

{"id": "ObjectID",
"startLocation": "123 Main Street",
"endLocation": "456 Elm Street",
"driverId": "ObjectID"}

GET /rides/{id}: This endpoint retrieves information about a specific ride by ID.

Request format:
HTTP method: GET
Endpoint: /rides/{id}, where {id} is the ID of the ride to retrieve
Parameters: None
Headers: None
Body: None

Response format:
Status code: 200 OK if the ride exists, 404 Not Found otherwise
Headers: Content-Type: application/json
Body: A JSON object representing the ride.

Example:

```
{
   "id": "ObjectID",
   "startLocation": "123 Main Street",
   "endLocation": "456 Elm Street",
   "driverId": "ObjectID"}
```

GET /rides/{id}/bookings: This endpoint retrieves information about all bookings for a particular ride by ID.

Request format:
HTTP method: GET
Endpoint: /rides/{id}/bookings, where {id} is the ID of the ride to retrieve
Parameters: None
Headers: None
Body: None

Response format:
Status code: 200 OK if the ride exists, 404 Not Found otherwise
Headers: Content-Type: application/json
Body: A JSON object representing the ride.

Example:

```
{
    "_id": "ObjectID",
   "user": "ObjectID",
   "ride": "ObjectID",
   "status": "booked || cancelled",
}
```

POST /rides: This endpoint creates a new ride.

Request format:

HTTP method: POST
Endpoint: /rides
Parameters: None
Headers: Content-Type: application/json
Body: A JSON object representing the ride to create.

Example:

```
{
   "startLocation": "123 Main Street",
   "endLocation": "456 Elm Street",
   "driverId": "ObjectID",
}
```

Response format:

Status code: 201 Created
Headers: Location: /rides/{id}, where {id} is the ID of the newly created ride
Body: A JSON object representing the newly created ride.
Example:

```
{
   "id": "ObjectID",
   "startLocation": "123 Main Street",
   "endLocation": "456 Elm Street",
   "driverId": "ObjectID"
}
```

PUT /rides/{id}: This endpoint updates information about a specific ride by ID.

Request format:

HTTP method: PUT
Endpoint: /rides/{id}, where {id} is the ID of the ride to update
Parameters: None
Headers: Content-Type: application/json
Body: A JSON object representing the ride with the updated information.

Example:

```
{
   "startLocation": "123 Main Street",
   "endLocation": "456 Elm Street",
   "driverId": "ObjectID"
}
```

Response format:
Status code: 200 OK if the ride was updated, 404 Not Found if the ride doesn't exist
Headers: None
Body: A JSON object representing the updated ride.
Example:

```
{
   "id": "ObjectID",
   "startLocation": "123 Main Street",
   "endLocation": "456 Elm Street",
   "driverId": "ObjectID",
}
```

DELETE /rides/:id This endpoint deletes a specific ride by ID.

Request format:

HTTP method: PUT
Endpoint: /rides/{id}, where {id} is the ID of the ride to delete
Parameters: None
Headers: Content-Type: application/json
Body: None

Response format:
Status code: 200 OK if the ride was deleted, 404 Not Found if the ride doesn't exist
Headers: None
Body: An object representing the deletedride.
Example:
{ "id": "ObjectID"}

/bookings:

GET /bookings : This endpoint retrieves all bookings.

Request format:

HTTP method: GET
Endpoint: /bookings
Parameters: None
Headers: None
Body: None

Response format:
Status code: 200 OK, 404 Not Found otherwise
Headers: Content-Type: application/json
Body: An array of JSON objects, each representing a ride.

Example:

```
{
    "_id": "ObjectID",
   "user": "ObjectID",
   "ride": "ObjectID",
   "status": "booked || cancelled",
}
```

GET /bookings/:id : This endpoint retrieves about a specific booking by ID.

Request format:
HTTP method: GET
Endpoint: /bookings/:id
Parameters: None
Headers: None
Body: None

Response format:
Status code: 200 OK, 404 Not Found otherwise
Headers: Content-Type: application/json
Body: An array of JSON objects, each representing a ride.

Example:

```
{
    "_id": "ObjectID",
   "user": "ObjectID",
   "ride": "ObjectID",
   "status": "booked || cancelled",
}
```

POST /booking: This endpoint creates a new booking.

Request format:

HTTP method: POST
Endpoint: /booking
Parameters: None
Headers: Content-Type: application/json
Body: A JSON object representing the booking to create.

Example:
{
"_id": "ObjectID",
"user": "ObjectID",
"ride": "ObjectID",
"status": "booked || cancelled",
}

Response format:

Status code: 201 Created
Headers: Location: /booking/{id}, where {id} is the ID of the newly created booking
Body: A JSON object representing the newly created booking.

Example:
{
"_id": "ObjectID",
"user": "ObjectID",
"ride": "ObjectID",
"status": "booked || cancelled",
}

PUT /booking /{id}: This endpoint updates information about a specific booking by ID.

Request format:

HTTP method: PUT
Endpoint: /booking /{id}, where {id} is the ID of the booking to update
Parameters: None
Headers: Content-Type: application/json
Body: A JSON object representing the booking with the updated information.

Example:
{
"_id": "ObjectID",
"user": "ObjectID",
"ride": "ObjectID",
"status": "booked || cancelled",
}

Response format:
Status code: 200 OK if the booking was updated, 404 Not Found if the bookingdoesn't exist

Headers: None
Body: A JSON object representing the updated booking.
Example:
Example:

```
{
    "_id": "ObjectID",
   "user": "ObjectID",
   "ride": "ObjectID",
   "status": "booked || cancelled",
}
```


DELETE /bookings/:id This endpoint deletes a specific booking by ID.

Request format:

HTTP method: PUT
Endpoint: /bookings/{id}, where {id} is the ID of the booking to delete
Parameters: None
Headers: Content-Type: application/json
Body: None

Response format:
Status code: 200 OK if the booking was deleted, 404 Not Found if the booking doesn't exist
Headers: None
Body: An object representing the deleted booking.
Example:

```
{ "id": "ObjectID"}
```