Autonomous Robot Delivery System API Documentation

Introduction

Welcome to the Autonomous Delivery Robot API! This API allows users to make small-package deliveries within college campuses with robotic drones. It's designed to navigate predefined campus pathways, the robot delivers items such as books, food, and supplies between dormitories, libraries, etc.

Base url : http://127.0.0.1:8000

Deliveries

Get All Deliveries

URL: /deliveries
Method: GET
Description: Fetch a list of all scheduled and completed deliveries in the database.

Request Parameters
No parameters needed

Request Body Object Structure:
No request object needed.

Response Object Structure:
```
[
  {
        "id": "integer", // Unique ID of the Delivery
        "pickup_location": { // Pickup Location Object
                “name”: "string", // Name of Pickup Location
                “latitude”: "float",  // Latitude Coordinate of Pickup Location
                “longitude”: "float",  // Longitude Coordinate of Pickup Location
        },
        "dropoff_location": { // Drop Off Location Object
                “name”: "string", // Name of Drop Off Location
                “latitude”: "float",  // Latitude Coordinate of Drop Off Location
                “longitude”: "float",  // Longitude Coordinate of Drop Off Location
        },
        "status": "string", // Delivery Status (e.g., Scheduled, In Progress, Completed)
        "distance": "float", // Distance Between Pick Up and Drop Off Location in Miles
        "delivery_time": "string" // Time for Delivery in ISO Timestamp
  },
  ...
]
```

Response Example:

```
[
  {
        "id": 1, // Unique ID of the Delivery
        "pickup_location": { // Pickup Location Object
              “name”: "John T. Richardson Library", // Name of John T. Richardson Library
              “latitude”: 41.9249,  // Latitude Coordinate of John T. Richardson Library
              “longitude”: -87.6553,  // Longitude Coordinate of John T. Richardson Library
        },
        "dropoff_location": { // Drop Off DePaul Student Center
              “name”: "DePaul Student Center", // Name of DePaul Student Center
              “latitude”: 41.9228,  // Latitude Coordinate of DePaul Student Center
              “longitude”: -87.6535,  // Longitude Coordinate of DePaul Student Center
        },
        "status": "Completed", // Delivery Status (e.g., Scheduled, In Progress, Completed)
        "distance": 0.30, // Distance Between Pick Up and Drop Off Location in Miles
        "delivery_time": "2025-01-26T15:05:00" // Time for Delivery in ISO Timestamp
  },
  {
  "id": "2", // Unique ID of the Delivery
  "pickup_location": { // Pickup Location Object
    "name": "DePaul Cafeteria", // Name of DePaul Loop Cafeteria
    "latitude": 41.8801,  // Latitude Coordinate of DePaul Loop Cafeteria
    "longitude": -87.6234  // Longitude Coordinate of DePaul Loop Cafeteria
  },
  "dropoff_location": { // Drop Off Location Object
    "name": "DePaul Admin Office", // Name of DePaul Loop Admin Office
    "latitude": 41.8810,  // Latitude Coordinate of DePaul Loop Admin Office
    "longitude": -87.6225  // Longitude Coordinate of DePaul Loop Admin Office
  },
  "status": "Scheduled", // Delivery Status (e.g., Scheduled, In Progress, Completed)
  "distance": 0.50, // Distance Between Pick Up and Drop Off Location in Miles
  "delivery_time": "2025-01-26T16:00:00" // Time for Delivery in ISO Timestamp
  }
]
```

HTTP Status Codes:

- 200 OK: Successfully retrieved the list of deliveries.
- 500 Internal Server Error: Error while fetching deliveries.

Create a New Delivery

URL: /deliveries
Method: POST
Description: Add a new delivery to the database.

Request Parameters
No parameters needed.

Request Body Object Structure:

```
{
        "pickup_location": { // Pickup Location Object
                “name”: "string", // Name of Pickup Location
                “latitude”: "float",  // Latitude Coordinate of Pickup Location
                “longitude”: "float",  // Longitude Coordinate of Pickup Location
        },
        "dropoff_location": { // Drop Off Location Object
                “name”: "string", // Name of Drop Off Location
                “latitude”: "float",  // Latitude Coordinate of Drop Off Location
                “longitude”: "float",  // Longitude Coordinate of Drop Off Location
        },
        "distance": "float", // (Optional for Creation, Calculated by Server if Not Provided)
Distance Between Pick Up and Drop Off Location in Miles
        "delivery_time": "string" // (Optional for Creation) Time for Delivery in ISO Timestamp
}
```

Request Body Object Structure Example:

```
{
           "pickup_location": {
              "name": "DePaul Art Museum",
              "latitude": 41.9252,
              "longitude": -87.6527
           },
           "dropoff_location": {
              "name": "Wish Field",
              "latitude": 41.9220,
              "longitude": -87.6565
           },
           "distance": 0.25, // Optional
           "delivery_time": "2025-01-26T14:30:00" // Optional
}
```

Response Object Structure:

```
{
      "id": "integer", // Unique ID of the delivery
      "pickup_location": { // Pickup Location Object
            “name”: "string", // Name of Pickup Location
            “latitude”: "float",  // Latitude Coordinate of Pickup Location
            “longitude”: "float",  // Longitude Coordinate of Pickup Location
      },
      "dropoff_location": { // Drop Off Location Object
            “name”: "string", // Name of Drop Off Location
            “latitude”: "float",  // Latitude Coordinate of Drop Off Location
            “longitude”: "float",  // Longitude Coordinate of Drop Off Location
      },
      "status": "string", // Delivery Status (e.g., Scheduled, In Progress, Completed)
      "distance": "float", // Distance Between Pick Up and Drop Off Location in Miles
      "delivery_time": "string" // Time for Delivery in ISO Timestamp
}
```

Response Example:

```
{
        "id": 4,
        "pickup_location": {
          "name": "DePaul Art Museum",
          "latitude": 41.9252,
          "longitude": -87.6527
        },
        "dropoff_location": {
          "name": "Wish Field",
          "latitude": 41.9220,
          "longitude": -87.6565
        },
        "status": "Scheduled",
        "distance": 0.25,
        "delivery_time": "2025-01-26T14:30:00"
}
```

HTTP Status Codes:

- 201 Created: Successfully created a new delivery request.
- 400 Bad Request: Invalid or incomplete data provided.

- 500 Internal Server Error: Error while creating delivery.

Get Delivery by ID

URL: /deliveries/{id}
Method: GET
Description: Retrieve details of a specific delivery by its ID.

Request Parameters
id: The unique ID of the specific delivery.

Request Body Object Structure:
No request object needed.

Response Object Structure:

```
{
        "id": "integer", // Unique ID of the delivery
        "pickup_location": { // Pickup Location Object
                “name”: "string", // Name of Pickup Location
                “latitude”: "float",  // Latitude Coordinate of Pickup Location
                “longitude”: "float",  // Longitude Coordinate of Pickup Location
        },
        "dropoff_location": { // Drop Off Location Object
                “name”: "string", // Name of Drop Off Location
                “latitude”: "float",  // Latitude Coordinate of Drop Off Location
                “longitude”: "float",  // Longitude Coordinate of Drop Off Location
        },
        "status": "string", // Delivery Status (e.g., Scheduled, In Progress, Completed)
        "distance": "float", // Distance Between Pick Up and Drop Off Location in Miles
        "delivery_time": "string" // Time for Delivery in ISO Timestamp
}
```

Response Example:

```
{
        "id": "1", // Unique ID of the Delivery
        "pickup_location": { // Pickup Location Object
                “name”: "John T. Richardson Library", // Name of John T. Richardson Library
                “latitude”: "41.9249",  // Latitude Coordinate of John T. Richardson Library
                “longitude”: "-87.6553",  // Longitude Coordinate of John T. Richardson Library
        },
        "dropoff_location": { // Drop Off DePaul Student Center
                “name”: "DePaul Student Center", // Name of DePaul Student Center
                “latitude”: "41.9228",  // Latitude Coordinate of DePaul Student Center
                “longitude”: "-87.6535",  // Longitude Coordinate of DePaul Student Center
        },
        "status": "Completed", // Delivery Status (e.g., Scheduled, In Progress, Completed)
        "distance": "0.30", // Distance Between Pick Up and Drop Off Location in Miles
        "delivery_time": "2025-01-26T15:05:00" // Time for Delivery in ISO Timestamp
}
```

HTTP Status Codes:
- 200 OK: Successfully retrieved the list of deliveries.
- 400 Bad Request: Invalid or incomplete data provided
- 500 Internal Server Error: Error while fetching deliveries.

Delete a Delivery by ID

URL: /deliveries/{id}
Method: DELETE
Description: Delete a specific delivery from the database using its unique ID.

Request Parameters
id: The unique ID of the specific delivery.

Request Body Object Structure:
No request object needed.

Response Object Structure:
```
{
        "id": "integer", // Unique ID of the delivery
        "pickup_location": { // Pickup Location Object
                "name": "string", // Name of Pickup Location
                "latitude": "float",  // Latitude Coordinate of Pickup Location
                "longitude": "float",  // Longitude Coordinate of Pickup Location
        },
        "dropoff_location": { // Drop Off Location Object
                "name": "string", // Name of Drop Off Location
                "latitude": "float",  // Latitude Coordinate of Drop Off Location
                "longitude": "float",  // Longitude Coordinate of Drop Off Location
        },
        "status": "string", // Delivery Status (e.g., Scheduled, In Progress, Completed)
        "distance": "float", // Distance Between Pick Up and Drop Off Location in Miles
        "delivery_time": "string" // Time for Delivery in ISO Timestamp
}
```

Response Example:
```
{
        "id": "1", // Unique ID of the Delivery
        "pickup_location": { // Pickup Location Object
                “name”: "John T. Richardson Library", // Name of John T. Richardson Library
                “latitude”: "41.9249",  // Latitude Coordinate of John T. Richardson Library
                “longitude”: "-87.6553",  // Longitude Coordinate of John T. Richardson Library
        },
        "dropoff_location": { // Drop Off DePaul Student Center
                “name”: "DePaul Student Center", // Name of DePaul Student Center
                “latitude”: "41.9228",  // Latitude Coordinate of DePaul Student Center
                “longitude”: "-87.6535",  // Longitude Coordinate of DePaul Student Center
        },
        "status": "Completed", // Delivery Status (e.g., Scheduled, In Progress, Completed)
        "distance": "0.30", // Distance Between Pick Up and Drop Off Location in Miles
        "delivery_time": "2025-01-26T15:05:00" // Time for Delivery in ISO Timestamp
}
```

HTTP Status Codes:
- 200 OK: Successfully deleted the delivery.
- 404 Not Found: Delivery with the specified ID does not exist.
- 500 Internal Server Error: Error while trying to delete the delivery.

Update a Delivery by ID

URL: /deliveries/{id}
Method: PATCH
Description: Update a specific delivery from the database using its unique ID.

Request Parameters
id: The unique ID of the delivery to be retrieved.

Request Body Object Structure:
All fields are optional. Only fields provided in request will be updated.

```
{
        "pickup_location": { // Optional
        “name”: "string", // Updated Name of Pickup Location
        “latitude”: "float",  // Updated Latitude Coordinate of Pickup Location
        “longitude”: "float",  // Updated Longitude Coordinate of Pickup Location
        },
        "dropoff_location": { // Optional
                “name”: "string", // Updated Name of Drop Off Location
                “latitude”: "float",  // Updated Latitude Coordinate of Drop Off Location
                “longitude”: "float",  // Updated Longitude Coordinate of Drop Off Location
        },
        "status": "string", // Updated delivery status (e.g., Scheduled, In Progress, Completed)
```

```
     delivery_time": "string" // Updated delivery time in ISO timestamp
}
```

Response Object Structure:

```
{
	"id": "integer", // Unique ID of the delivery
	"pickup_location": { // Updated Pickup Location Object
		“name”: "string", // Updated Name of Pickup Location
		“latitude”: "float",  // Updated Latitude Coordinate of Pickup Location
		“longitude”: "float",  // Updated Longitude Coordinate of Pickup Location
	},
	"dropoff_location": { // Updated Drop Off Location Object
		“name”: "string", // Updated Name of Drop Off Location
		“latitude”: "float",  // Updated Latitude Coordinate of Drop Off Location
		“longitude”: "float",  // Updated Longitude Coordinate of Drop Off Location
	},
	"status": "string", // Updated Delivery Status (e.g., Scheduled, In Progress, Completed)
	"distance": "float", // Recalculated Distance Between Pick Up and Drop Off Location in
Miles
	"delivery_time": "string" // Updated Time for Delivery in ISO Timestamp
}
```

Response Example:

```
{
      "id": 4,
      "pickup_location": {
         "name": "DePaul Art Museum",
         "latitude": 41.9252,
         "longitude": -87.6527
      },
      "dropoff_location": {
         "name": "Wish Field",
         "latitude": 41.9220,
         "longitude": -87.6565
      },
      "status": "Delivered", // Updated Status
      "distance": 0, // (Automatically set to 0 because delivery status was updated)
      "delivery_time": "2025-01-26T14:30:00"
}
```

HTTP Status Codes:

- 200 OK: Successfully updated the delivery.
- 404 Not Found: Delivery with the specified ID does not exist.
- 500 Internal Server Error: Error while updating deliveries.

Robots

Get All Robots

URL: /robots
Method: GET
Description: Fetch a list of all robots currently in the database.

Request Parameters
No parameters needed

Request Body Object Structure:
No request object needed.

Response Object Structure:
```
[
   {
         "id": "integer", // Unique ID of the Robot
         "current_location": { // Current Location of Robot
                  “latitude”: "float",  // Latitude Coordinate of Robot’s Current Location
                  “longitude”: "float",  // Longitude Coordinate of Robot’s Current Location
         },
         "status": "string", // Current status of the robot (e.g., Available, In Progress, Offline)
         "battery": "float", // Robot's Battery Level as a Percentage (0-100)
         "max_load": "float" // Maximum Load the Robot Can Carry in Pounds (lbs)
   },
   ...
]
```

Response Example:
```
[
   {
         "id": 1, // Unique ID of the Robot
         "current_location": { // Current Location of Robot
                  “latitude”: 41.9249,  // Latitude Coordinate of Robot’s Current Location
                  “longitude”: -87.6553,  // Longitude Coordinate of Robot’s Current Location
         },
         "status": "Available", // Current status of the robot (e.g., Available, In Progress, Offline)
         "battery": 95.5, // Robot's Battery Level as a Percentage (0-100)
         "max_load": 25.0 // Maximum Load the Robot Can Carry in Pounds (lbs)
   },
   {
         "id": 2, // Unique ID of the Robot
         "current_location": { // Current Location of Robot
                  “latitude”: 41.9228,  // Latitude Coordinate of Robot’s Current Location
                  “longitude”: -87.6535,  // Longitude Coordinate of Robot’s Current Location
         },
         "status": "In Progress", // Current status of the robot (e.g., Available, In Progress, Offline)
         "battery": 99.9, // Robot's Battery Level as a Percentage (0-100)
         "max_load": 50.0 // Maximum Load the Robot Can Carry in Pounds (lbs)
   }
]
```

HTTP Status Codes:
- 200 OK: Successfully retrieved the list of robots.
- 500 Internal Server Error: Error while fetching robots.

Create a New Robot

URL: /robots
Method: POST
Description: Add a new robot to the database.

Request Parameters
No parameters needed.

Request Body Object Structure:
No optional fields.

```
{
        "current_location": { // Current Location of Robot
                “latitude”: "float",  // Latitude Coordinate of Robot’s Current Location
                “longitude”: "float",  // Longitude Coordinate of Robot’s Current Location
        },
        "status": "string", // Current status of the robot (e.g., Available, In Progress, Offline)
        "battery": "float", // Robot's Battery Level as a Percentage (0-100)
        "max_load": "float" // Maximum Load the Robot Can Carry in Pounds (lbs)
}
```

Request Body Object Structure Example:

```
{
  "current_location": {
    "latitude": 41.4573,
    "longitude": -87.0299,
  },
  "status": "Available",
  "battery": 100.0,
  "max_load": 50.0
}
```

Response Object Structure:

```
{
        "id": "integer", // Unique ID of the Robot
        "current_location": { // Current Location of Robot
                "latitude": "float",  // Latitude Coordinate of Robot's Current Location
                "longitude": "float",  // Longitude Coordinate of Robot's Current Location
        },
        "status": "string", // Current status of the robot (e.g., Available, In Progress, Offline)
        "battery": "float", // Robot's Battery Level as a Percentage (0-100)
        "max_load": "float" // Maximum Load the Robot Can Carry in Pounds (lbs)
}
```

Response Example:

```
{
        "id": 3, // Unique ID of the Robot
        "current_location": { // Current Location of Robot
                "latitude": 41.4573,  // Latitude Coordinate of Robot's Current Location
                "longitude": -87.0299,  // Longitude Coordinate of Robot's Current Location
        },
        "status": "Available", // Current status of the robot (e.g., Available, In Progress, Offline)
        "battery": 100.0, // Robot's Battery Level as a Percentage (0-100)
        "max_load": 50.0 // Maximum Load the Robot Can Carry in Pounds (lbs)
 }
```

HTTP Status Codes:

- 201 Created: Successfully created a new robot request.
- 400 Bad Request: Invalid or incomplete data provided.
- 500 Internal Server Error: Error while creating robot.

Get Robot by ID

URL: /robots/{id}
Method: GET
Description: Retrieve details of a specific robot by its ID.

Request Parameters
id: The unique ID of the specific robot.

Request Body Object Structure:

No request object needed.

Response Object Structure:
```
{
        "id": "integer", // Unique ID of the Robot
        "current_location": { // Current Location of Robot
                “latitude”: "float",  // Latitude Coordinate of Robot’s Current Location
                “longitude”: "float",  // Longitude Coordinate of Robot’s Current Location
        },
        "status": "string", // Current status of the robot (e.g., Available, In Progress, Offline)
        "battery": "float", // Robot's Battery Level as a Percentage (0-100)
        "max_load": "float" // Maximum Load the Robot Can Carry in Pounds (lbs)
}
```

Response Example:
```
{
        "id": 1, // Unique ID of the Robot
        "current_location": { // Current Location of Robot
                “latitude”: 41.9249,  // Latitude Coordinate of Robot’s Current Location
                “longitude”: -87.6553,  // Longitude Coordinate of Robot’s Current Location
        },
        "status": "Available", // Current status of the robot (e.g., Available, In Progress, Offline)
        "battery": 95.5, // Robot's Battery Level as a Percentage (0-100)
        "max_load": 25.0 // Maximum Load the Robot Can Carry in Pounds (lbs)
}
```

HTTP Status Codes:
- 200 OK: Successfully retrieved the robots.
- 400 Bad Request: Invalid or incomplete data provided
- 500 Internal Server Error: Error while fetching robot.

Delete a Robot by ID

URL: /robots/{id}
Method: DELETE
Description: Delete a specific robot from the database using its unique ID.

Request Parameters
id: The unique ID of the specific robot.

Request Body Object Structure:
No request object needed.

Response Object Structure:

```
{
        "id": "integer", // Unique ID of the Robot
        "current_location": { // Current Location of Robot
                “latitude”: "float",  // Latitude Coordinate of Robot’s Current Location
                “longitude”: "float",  // Longitude Coordinate of Robot’s Current Location
        },
        "status": "string", // Current status of the robot (e.g., Available, In Progress, Offline)
        "battery": "float", // Robot's Battery Level as a Percentage (0-100)
        "max_load": "float" // Maximum Load the Robot Can Carry in Pounds (lbs)
}
```

Response Example:

```
{
        "id": 1, // Unique ID of the Robot
        "current_location": { // Current Location of Robot
                “latitude”: 41.9249,  // Latitude Coordinate of Robot’s Current Location
                “longitude”: -87.6553,  // Longitude Coordinate of Robot’s Current Location
        },
        "status": "Available", // Current status of the robot (e.g., Available, In Progress, Offline)
        "battery": 95.5, // Robot's Battery Level as a Percentage (0-100)
        "max_load": 25.0 // Maximum Load the Robot Can Carry in Pounds (lbs)
}
```

HTTP Status Codes:

- 200 OK: Successfully deleted the robot.
- 404 Not Found: Robot with the specified ID does not exist.
- 500 Internal Server Error: Error while trying to delete the robot.

Update a Robot by ID

URL: /robots/{id}
Method: PATCH
Description: Update a specific robot’s status from the database using its unique ID.

Request Parameters
id: The unique ID of the robot to be retrieved.

Request Body Object Structure:
All fields listed here are optional.

```
{
        "current_location": { // Updated Current Location of Robot
                “latitude”: "float",  // Updated Latitude Coordinate of Robot’s Current Location
                “longitude”: "float",  // Updated Longitude Coordinate of Robot’s Current Location
        },
        "status": "string", // Updated Current status of the robot (e.g., Available, In Progress,
Offline)
        "battery": "float", // Updated Robot's Battery Level as a Percentage (0-100)
        "max_load": "float" // Updated Maximum Load the Robot Can Carry in Pounds (lbs)
}
```

Response Object Structure:

```
{
        "id": "integer", // Unique ID of the Robot
                "current_location": { // Current Location of Robot
                “latitude”: "float",  // Latitude Coordinate of Robot’s Current Location
                “longitude”: "float",  // Longitude Coordinate of Robot’s Current Location
        },
        "status": "string", // Current status of the robot (e.g., Available, In Progress, Offline)
        "battery": "float", // Robot's Battery Level as a Percentage (0-100)
        "max_load": "float" // Maximum Load the Robot Can Carry in Pounds (lbs)
}
```

Response Object Example:

```
{
        "id": 2, // Unique ID of the Robot
        "current_location": { // Current Location of Robot
                “latitude”: 41.9228,  // Latitude Coordinate of Robot’s Current Location
                “longitude”: -87.6535,  // Longitude Coordinate of Robot’s Current Location
        },
        "status": "Offline", // Updated Current status of the robot (e.g., Available, In Progress,
Offline)
        "battery": 99.9, // Robot's Battery Level as a Percentage (0-100)
        "max_load": 50.0 // Maximum Load the Robot Can Carry in Pounds (lbs)
   }
```

HTTP Status Codes:

- 200 OK: Successfully updated the status of the robot.
- 404 Not Found: Robot with the specified ID does not exist.
- 500 Internal Server Error: Error while updating robot status.

Get All Users

URL: /users

Method: GET
Description: Fetch a list of all users in the database.

Request Parameters
No parameters needed

Request Body Object Structure:
No request object needed.

Response Object Structure:
```
[
   {
         "id": “integer”,  // Unique ID of the User
         “full_name”: “string”, // First and Last Name of the User
         “username”: “string”, // Name of the User
         “email”: “string”, // Email of the User
         “role”: “string”, // Role of the User (e.g., Admin, Student, etc.)
    },
   …
]
```

Response Example:
```
[
   {
         "id": 1,  // Unique ID of the User
         “full_name”: “Vincent DePaul”, // First and Last Name of the User
         “username”: “VDepaul”, // Name of the User
         “email”: “VDepaul@depaul.edu”, // Email of the User
         “role”: “Student”, // Role of the User (e.g., Admin, Student, etc.)
    },
    {
         "id": 2,  // Unique ID of the User
         “full_name”: “Robert Manuel”, // First and Last Name of the User
         “username”: “PrezDePaul”, // Name of the User
         “email”: “prezdepaul@depaul.edu”, // Email of the User
         “role”: “Admin”, // Role of the User (e.g., Admin, Student, etc.)
    },
   …
]
```

HTTP Status Codes:
- 200 OK: Successfully retrieved the list of users.
- 500 Internal Server Error: Error while fetching users.

Create a New User

URL: /users
Method: POST
Description: Add a new user to the database.

Request Parameters
No parameters needed.

Request Body Object Structure:
```
{
	“full_name”: “string”, // First and Last Name of the User
	“username”: “string”, // Name of the User
	“email”: “string”, // Email of the User
	“role”: “string”, // (Optional) Role of the User (e.g., Admin, Student, etc.)
}
```

Request Body Object Structure Example:
```
{
	“full_name”: “John Smith”, // First and Last Name of the User
	“username”: “jsmith”, // Name of the User
	“email”: “jsmith@depaul.edu”, // Email of the User
	“role”: “Student”, // (Optional) Role of the User (e.g., Admin, Student, etc.)
}
```

Response Object Structure:
```
{
	"id": “integer”,  // Unique ID of the User
	“full_name”: “string”, // First and Last Name of the User
	“username”: “string”, // Name of the User
	“email”: “string”, // Email of the User
	“role”: “string”, // Role of the User (e.g., Admin, Student, etc.)
}
```

Response Example:
```
{
	"id": 3,  // Unique ID of the User
	“full_name”: “John Smith”, // First and Last Name of the User
	“username”: “jsmith”, // Name of the User
	“email”: “jsmith@depaul.edu”, // Email of the User
	“role”: “Student”, // Role of the User (e.g., Admin, Student, etc.)
}
```

HTTP Status Codes:

- 201 Created: Successfully created a new user.
- 400 Bad Request: Invalid or incomplete data provided.
- 500 Internal Server Error: Error while creating user.

Get User by ID

URL: /users/{id}
Method: GET
Description: Retrieve details of a specific user by their ID.

Request Parameters
Id: The unique ID of the user to be retrieved

Request Body Object Structure:
No request object needed.

Response Object Structure:

```
{
        "id": “integer”,  // Unique ID of the User
        “full_name”: “string”, // First and Last Name of the User
        “username”: “string”, // Name of the User
        “email”: “string”, // Email of the User
        “role”: “string”, // Role of the User (e.g., Admin, Student, etc.)
}
```

Response Example:

```
{
        "id": 1,  // Unique ID of the User
        “full_name”: “Vincent DePaul”, // First and Last Name of the User
        “username”: “VDepaul”, // Name of the User
        “email”: “VDepaul@depaul.edu”, // Email of the User
        “role”: “Student”, // Role of the User (e.g., Admin, Student, etc.)
}
```

HTTP Status Codes:

- 200 OK: Successfully retrieved the user.
- 400 Bad Request: Invalid or incomplete data provided.
- 500 Internal Server Error: Error while fetching user.

Delete a User by ID

URL: /users/{id}
Method: DELETE
Description: Delete a specific user from the database using its unique ID.

Request Parameters
id: The unique ID of the specific user.

Request Body Object Structure:
No request object needed.

Response Object Structure:
```
{
        "id": “integer”,  // Unique ID of the User
        “full_name”: “string”, // First and Last Name of the User
        “username”: “string”, // Name of the User
        “email”: “string”, // Email of the User
        “role”: “string”, // Role of the User (e.g., Admin, Student, etc.)
}
```

Response Example:
```
{
        "id": 1,  // Unique ID of the User
        “full_name”: “Vincent DePaul”, // First and Last Name of the User
        “username”: “VDepaul”, // Name of the User
        “email”: “VDepaul@depaul.edu”, // Email of the User
        “role”: “Student”, // Role of the User (e.g., Admin, Student, etc.)
}
```

HTTP Status Codes:
- 200 OK: Successfully deleted the user.
- 404 Not Found: User with the specified ID does not exist.
- 500 Internal Server Error: Error while trying to delete the user.

Update a User by ID

URL: /users/{id}
Method: PATCH
Description: Update a specific user from the database using its unique ID.

Request Parameters
id: The unique ID of the user to be retrieved.

Request Body Object Structure:
All fields are optional. Only fields provided in request will be updated.

```
{
        “full_name”: “string”, // Updated First and Last Name of the User
        “username”: “string”, // Updated Name of the User
        “email”: “string”, // Updated Email of the User
        “role”: “string”, // Updated Role of the User (e.g., Admin, Student, etc.)
}
```

Response Object Structure:

```
{
        "id": “integer”,  // Unique ID of the User
        “full_name”: “string”, // First and Last Name of the User
        “username”: “string”, // Name of the User
        “email”: “string”, // Email of the User
        “role”: “string”, // Role of the User (e.g., Admin, Student, etc.)
}
```

Response Example:

```
{
        “full_name”: “Jonathan Smith”, // Updated First and Last Name of the User
        “username”: “jsmith”, // Name of the User
        “email”: “jsmith@depaul.edu”, // Email of the User
        “role”: “Student”, // (Optional) Role of the User (e.g., Admin, Student, etc.)
}
```

HTTP Status Codes:

- 200 OK: Successfully updated the user.
- 404 Not Found: User with the specified ID does not exist.
- 500 Internal Server Error: Error while updating user.