



API Specification Doc

Version	Date	Author	Description
1.0	19-January-2024	Ziyad Alghamdi	Initial draft

Note: This document is in its preliminary stage. If any ambiguities arise, please reach out to me via email ziyad@genuinefze.com

Table of Contents

1. Register	4
This endpoint allows users to register with the application	4
Request	4
Request Body Example:	4
.....	5
sequence diagram for the "Register ":	5
.....	6
Additional Information:	7



User Registration Requirements:	7
Username: 3-50 characters	7
Email: 6-100 characters	7
Type of Plan: (Not specified)	7
Password: 6-50 characters	8
2. Login	8
This endpoint is used to authenticate a user by providing their username and password.	8
Request	8
Request Body Example:	8
.....	8
sequence diagram for the "LOGIN "	9
.....	9
Responses	10
Additional Information:	10
Upon a successful login (HTTP status code 200), the API will respond with a Bearer Token. This token should be included in the Authorization header of every subsequent incoming request for authentication.	10
3.Add Camera	11
This endpoint allows you to add a new camera to the system.	11
Request	11
Request Body Example:	11
sequence diagram for the "ADD CAMERA "	12
.....	13
Responses	13
Additional Information:	14
4.Update camera	14
Request	14
Request Body Example:	15
sequence diagram for the "UPDATE CAMERA "	16
Responses	16
5.Delete camera	17
This endpoint is used to delete a specific camera belonging to a user.	18
Request	18
Request Body Example:	18



sequence diagram for the "DELETE CAMERA "	19
Responses	20
6.Get user all cameras	21
Responses	23
7.Get user info	23
Request	24
Error code:	28
HTTP Status Code	28
Meaning	28
200	28
OK - Successful request	28
400	28
Bad Request - Invalid syntax or parameters	28
401	29
Unauthorized - Authentication or permission failure	29
404	29
Not Found - Requested resource not found	29
422	29
Unprocessable Entity - Semantic errors in the request	29
500	29
Internal Server Error - Unexpected server error	29

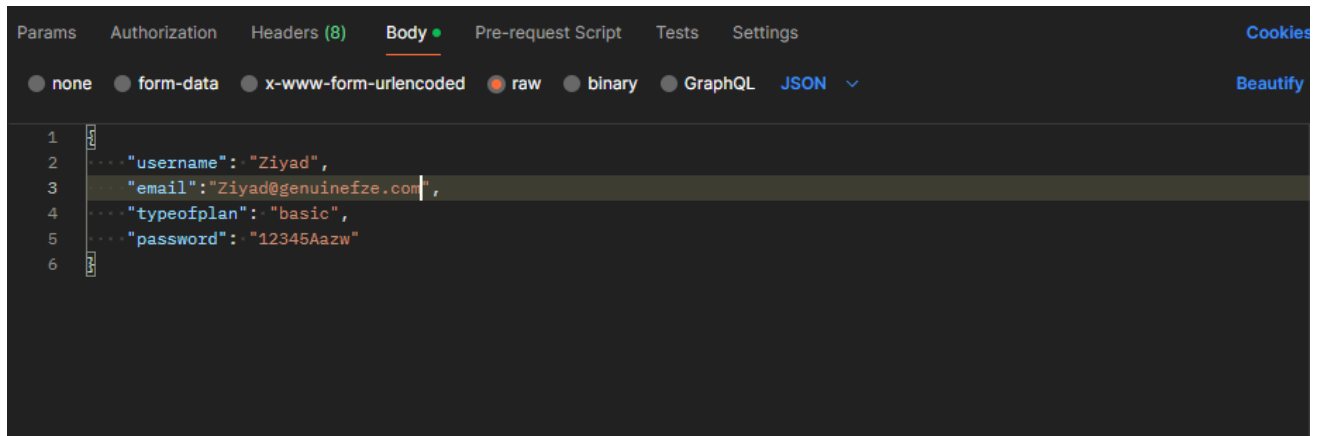
1. Register

This endpoint allows users to register with the application

Request

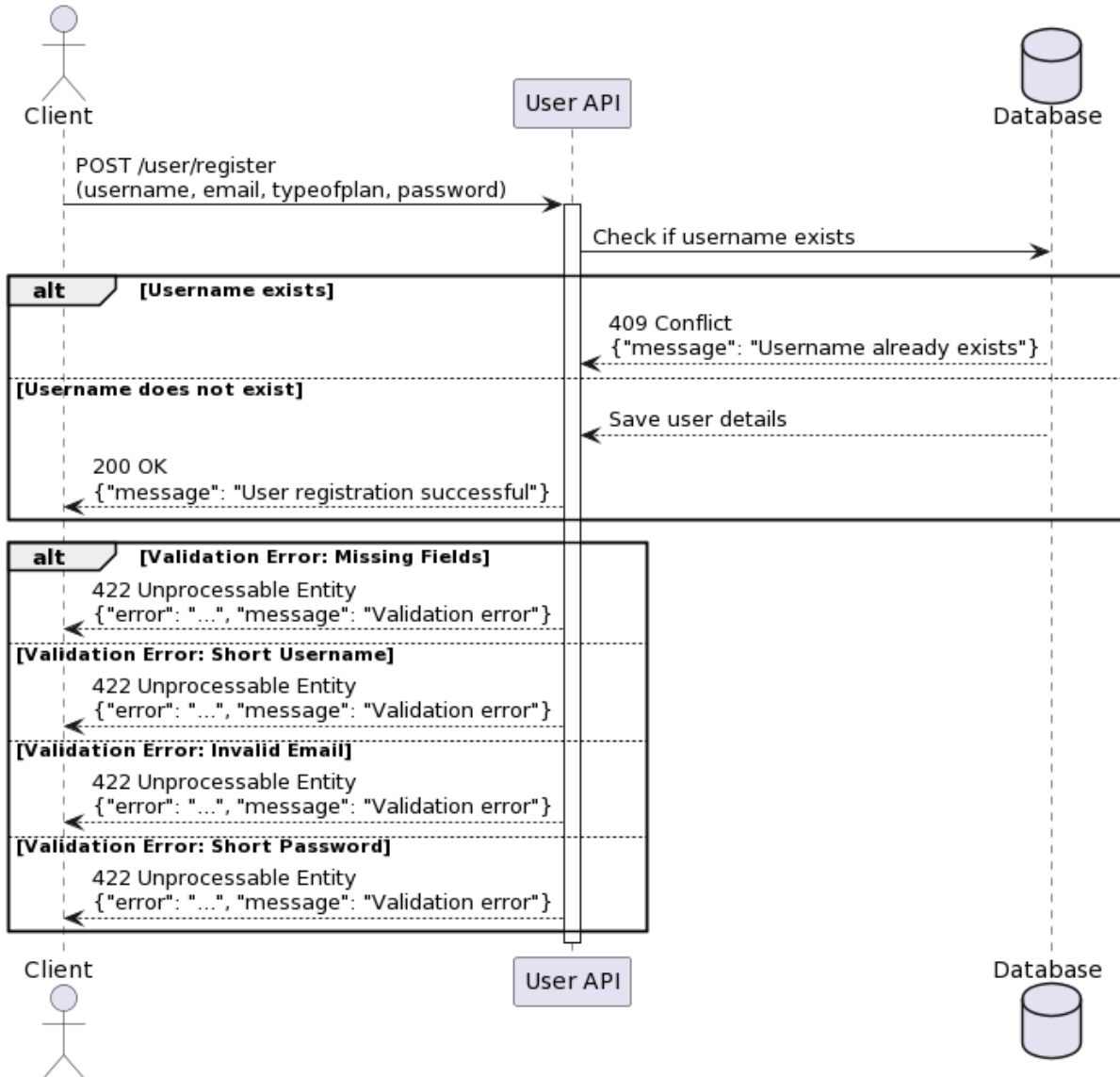
Method		URL	
POST		/user/register	
Type	Params	Values	
POST	username	string	
POST	email	string	
POST	typeofplan	string	
POST	password	string	

Request Body Example:



sequence diagram for the "Register ":

Register Endpoint Sequence Diagram





Status	Responses
200	<pre>{ "message": "User registration successful" }</pre>
422	<pre>{ "error": "1 validation error for User\nusername\n Field required [type=missing, input_value={'email': 'Ziyad@genuinef...'password': '12345Aazw'}, input_type=dict]\n For further information visit https://errors.pydantic.dev/2.5/v/missing", "message": "Validation error" }</pre>
422	<pre>{ "error": "1 validation error for User\nusername\n String should have at least 3 characters [type=string_too_short, input_value='', input_type=str]\n For further information visit https://errors.pydantic.dev/2.5/v/string_too_short", "message": "Validation error" }</pre>
422	<pre>{ "error": "1 validation error for User\nemail\n value is not a valid email address: The email address is not valid. It must have exactly one @- sign. [type=value_error, input_value='', input_type=str]", "message": "Validation error" }</pre>
422	<pre>"error": "1 validation error for User\npassword\n String should have at least 6 characters [type=string_too_short, input_value='', input_type=str]\n For further information visit https://errors.pydantic.dev/2.5/v/string_too_short", "message": "Validation error" }</pre>
409	<pre>{ "message": "Username already exists" }</pre>

Additional Information:
User Registration Requirements:
Username: 3-50 characters
Email: 6-100 characters
Type of Plan: (Not specified)



Password: 6-50 characters

2. Login

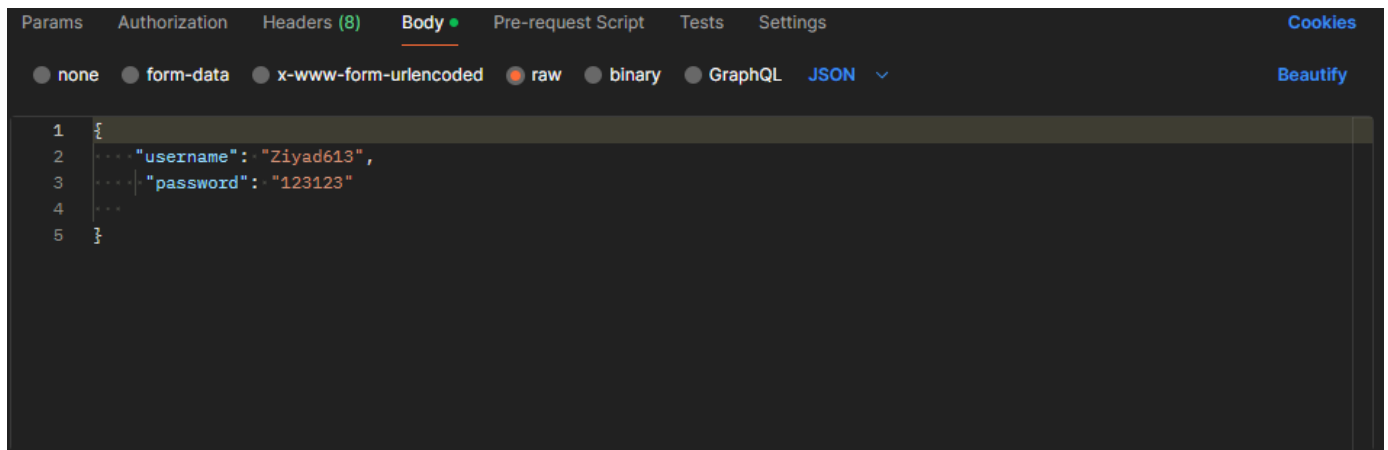
This endpoint is used to authenticate a user by providing their username and password.

Request

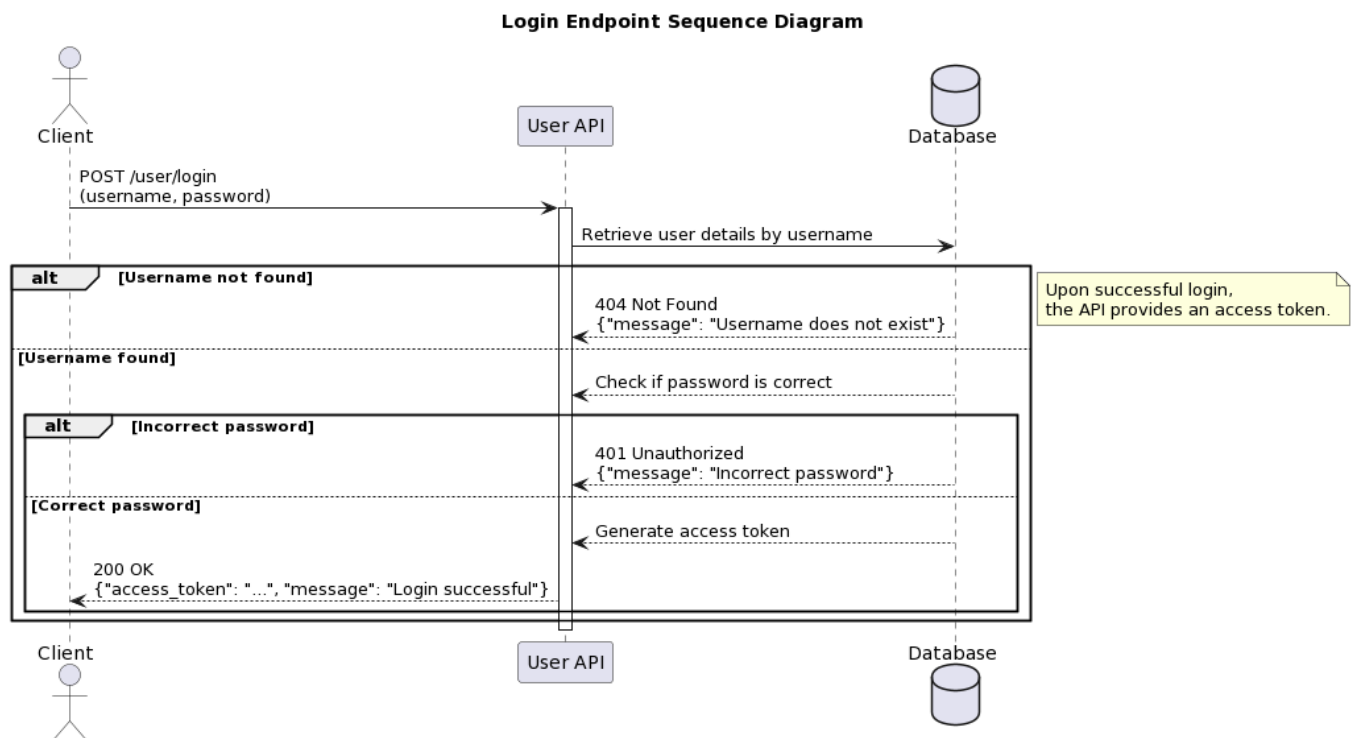
Method	URL
POST	/user/register

Type	Params	Values
POST	username	string
POST	password	string

Request Body Example:



sequence diagram for the "LOGIN "





Responses

Status	Responses
200	<pre>{ "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImlppeWFkNjEzIn0.cTJGgX 7w37rOM-G5i-kvBjyYAPRTZ8JVoR2pava8Mis", "message": "Login successful" }</pre>
404	<pre>{ "message": "Username does not exist" }</pre>
401	<pre>{ "message": "Incorrect password" }</pre>

Additional Information:

Upon a successful login (HTTP status code 200), the API will respond with a Bearer Token. This token should be included in the Authorization header of every subsequent incoming request for authentication.

3.Add Camera

This endpoint allows you to add a new camera to the system.

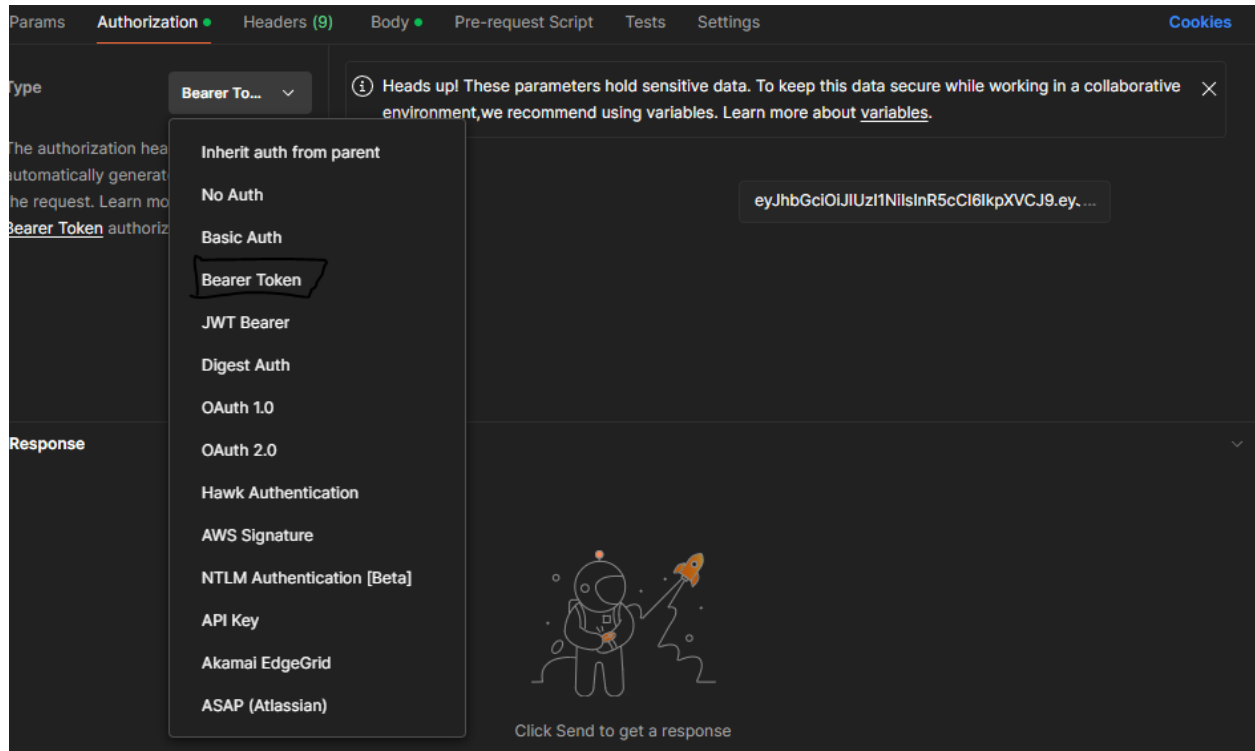
Request

Method	URL
POST	user/camera/add

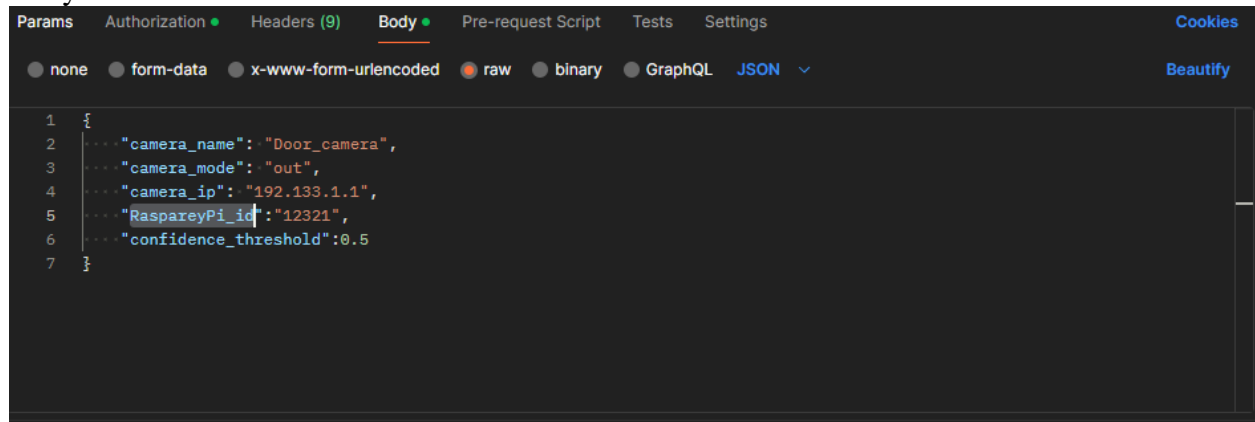
Type	Params	Values
Authorization	Bearer_Token	Token
POST	camera_name	string
POST	camera_mode	string
POST	camera_ip	string
POST	RaspberyPi_id	string
POST	confidence_threshold	float
POST	camera_port	INT

Request Body Example:

Authorization:

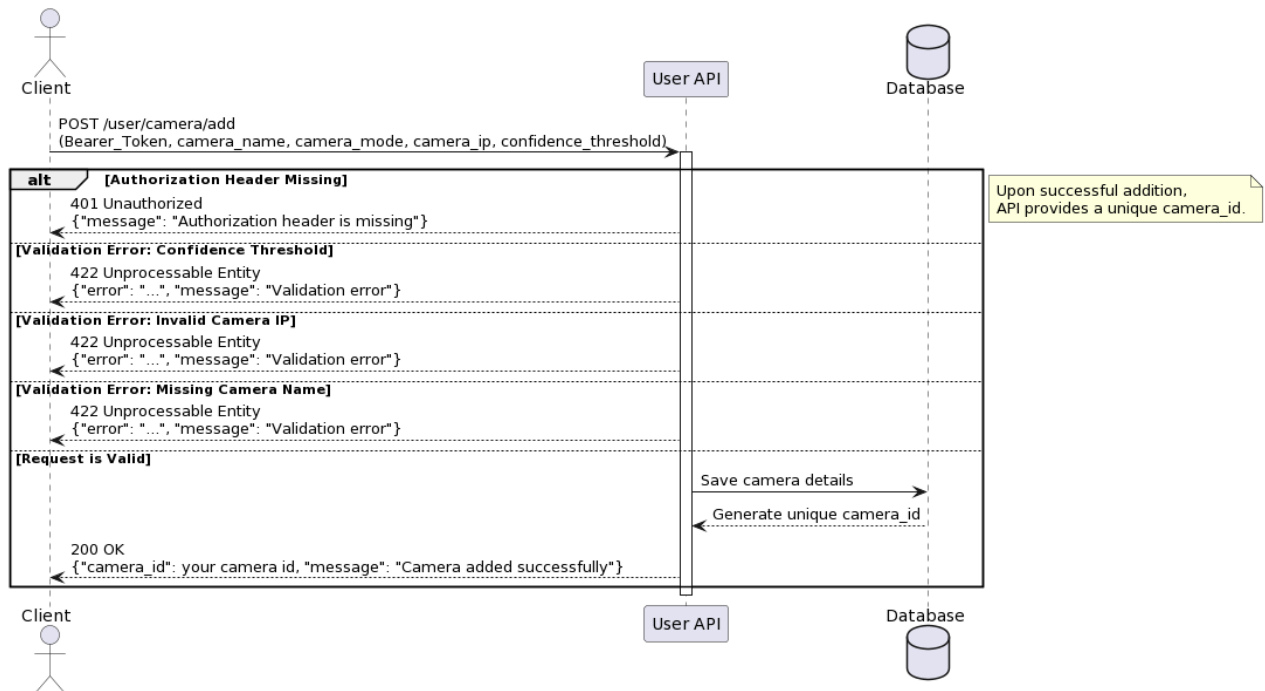


Body:



sequence diagram for the "ADD CAMERA"

Add Camera Endpoint Sequence Diagram



Responses

Status	Response
200	<pre>{ "camera_id": 90, "message": "Camera added successfully" }</pre>
401	<pre>{ "message": "Authorization header is missing" }</pre>
422	<pre>{ "error": "1 validation error for CameraData\nconfidence_threshold\n Input should be less than 1 [type=less_than, input_value=1.5, input_type=float]\n For further information visit https://errors.pydantic.dev/2.5/v/less_than", "message": "Validation error" }</pre>
422	<pre>{</pre>



	<pre>"error": "1 validation error for CameraData\ncamera_ip\n Input should be a valid string [type=string_type, input_value=1921331, input_type=int]\n For further information visit https://errors.pydantic.dev/2.5/v/string_type",\n "message": "Validation error"\n}</pre>
422	<pre>{\n "error": "1 validation error for CameraData\ncamera_name\n Field required [type=missing, input_value={'camera_mode': 'out', 'c...fidence_threshold': 0.5}, input_type=dict]\n For further information visit https://errors.pydantic.dev/2.5/v/missing",\n "message": "Validation error"\n}</pre>

Additional Information:

Upon successful addition of a camera (HTTP status code 200), the API will provide a response containing a unique `camera_id`. Subsequently, this `camera_id` must be included in the body of every incoming request related to the specific camera

To make adjustments to a previously added camera, you must include its specific `camera_id` in the request. The `camera_id` serves as the unique identifier required to modify the settings or properties of the corresponding camera.

Rasparey Pi will not directly supply video footage. Instead, the front-end developer will establish a link between the camera's IP and the front end using the `camera_ip`. This configuration enables the camera to simultaneously provide footage to both Rasparey Pi and the front end.

4.Update camera

Request

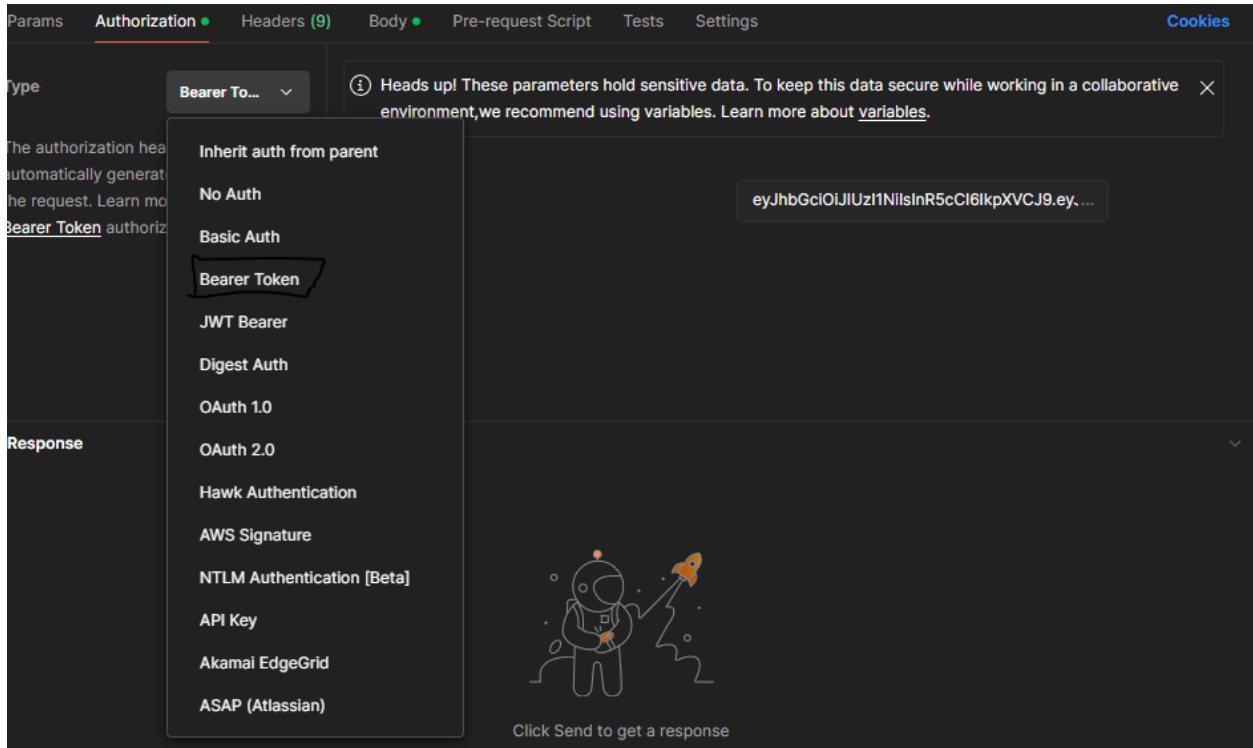
Method	URL
PUT	user/camera/update

Type	Params	Values
Authorization	Bearer_Token	Token
PUT	camera_id	string
PUT	camera_name	string
PUT	camera_mode	string
PUT	camera_ip	string

PUT	RaspareyPi_id	string
PUT	confidence_threshold	float
PUT	camera_port	INT

Request Body Example:

Authorization:



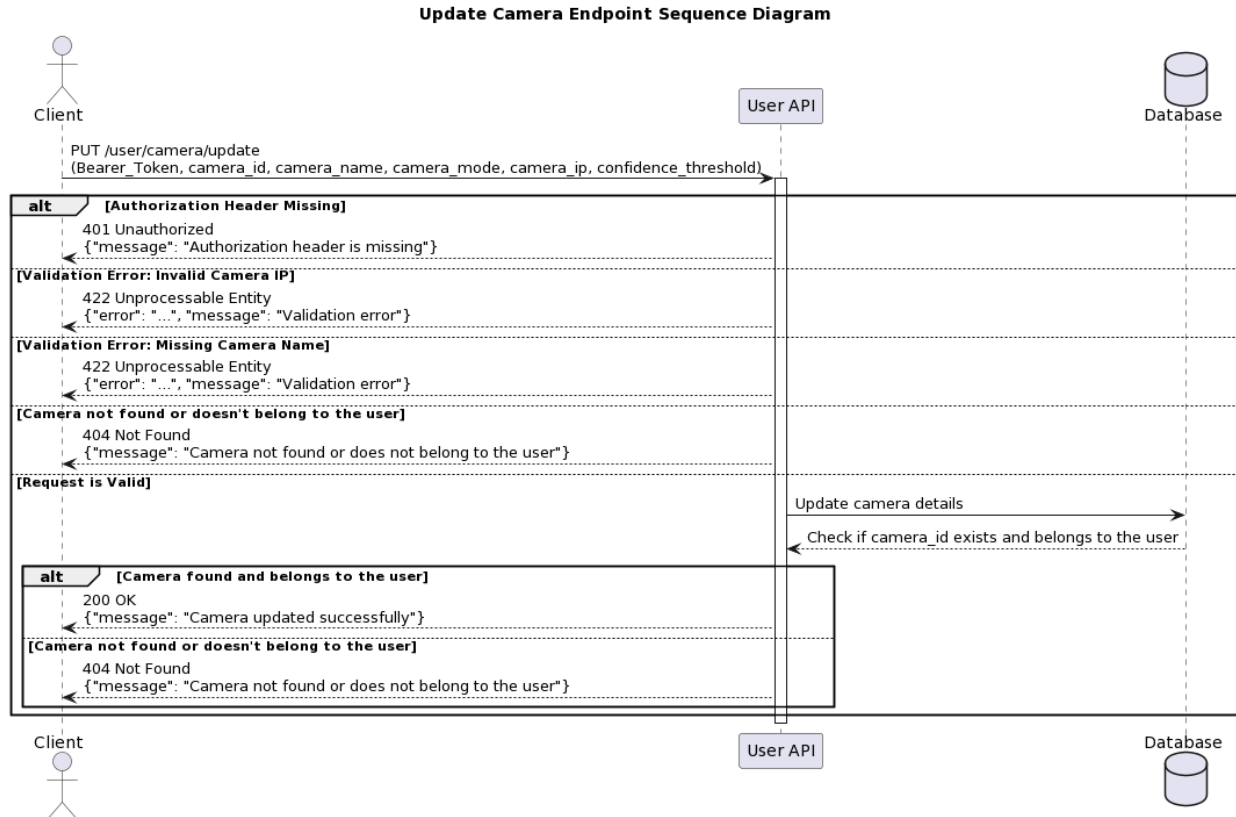
Body:

```

1 {
2   "camera_id": "94",
3   "camera_name": "Door_camera",
4   "camera_mode": "out",
5   "camera_ip": "193.133.1.2",
6   "RaspareyPi_id": "12321",
7   "confidence_threshold": 0.5
8 }

```

sequence diagram for the "UPDATE CAMERA "



Responses

Status	Response
200	<pre>{ "message": "Camera updated successfully" }</pre>
401	<pre>{ "message": "Authorization header is missing" }</pre>
404	<pre>{ "message": "Camera not found or does not belong to the user" }</pre>
422	<pre>{ "error": "1 validation error for CameraData\ncamera_ip\n Input should be a valid string [type=string_type, input_value=1921331,"</pre>

	<pre>input_type=int]\n For further information visit https://errors.pydantic.dev/2.5/v/string_type", "message": "Validation error" }</pre>
422	<pre>{ "error": "1 validation error for CameraData\ncamera_name\n Field required [type=missing, input_value={'camera_mode': 'out', 'c...fidence_threshold': 0.5}, input_type=dict]\n For further information visit https://errors.pydantic.dev/2.5/v/missing", "message": "Validation error" }</pre>

5.Delete camera



This endpoint is used to delete a specific camera belonging to a user.

Request

Method	URL
delete	user/camera/delete

Type	Params	Values
Authorization Delete	Bearer_Token camera_id	Token string

Request Body Example:

Authorization:

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

Type Bearer To... Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about [variables](#).

The authorization header is automatically generated for the request. Learn more about [Bearer Token](#) authorization.

Response

Inherit auth from parent

No Auth

Basic Auth

Bearer Token

JWT Bearer

Digest Auth

OAuth 1.0

OAuth 2.0

Hawk Authentication

AWS Signature

NTLM Authentication [Beta]

API Key

Akamai EdgeGrid

ASAP (Atlassian)

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ...

Click Send to get a response

Body:

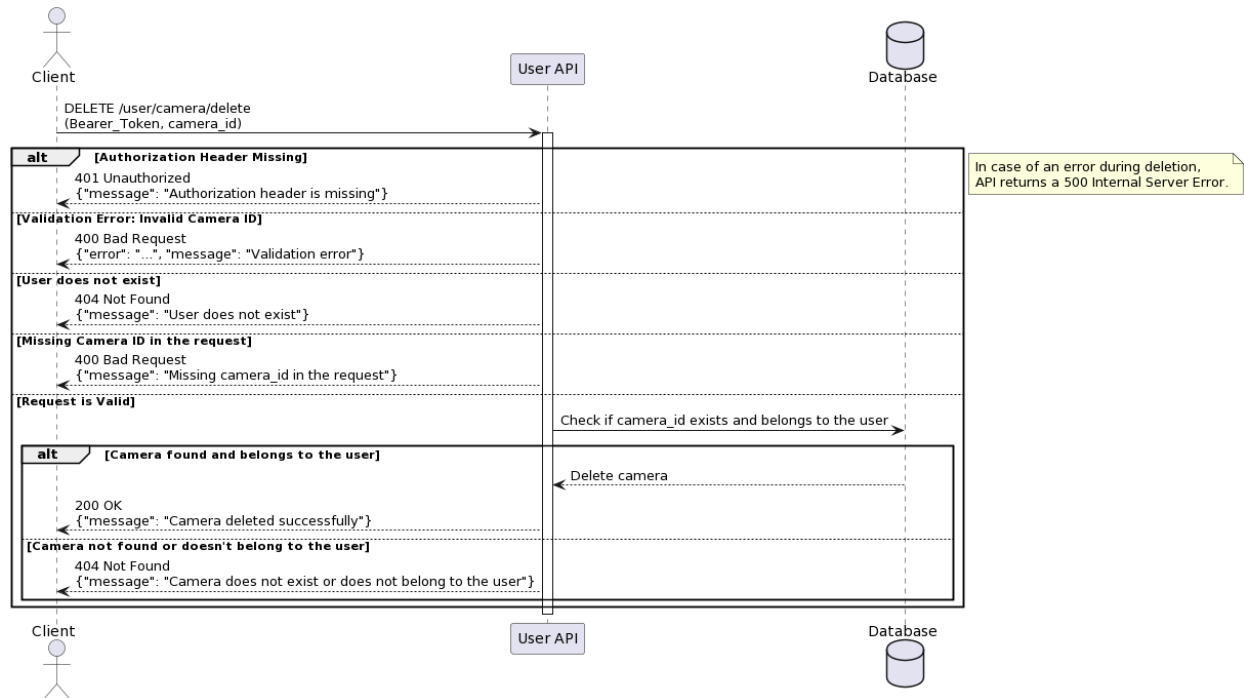
none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautiful

```

1  {
2    "camera_id": "91"
3  }
4  
```

sequence diagram for the "DELETE CAMERA "

Delete Camera Endpoint Sequence Diagram



Responses

Status	Response
200	<pre>{ "message": "Camera deleted successfully" }</pre>
401	<pre>{ "message": "Authorization header is missing" }</pre>
404	<pre>{ "message": "Camera does not exist or does not belong to the user" }</pre>
400	<pre>{ "error": "1 validation error for CameraIdBaseModel\ncamera_id\n Input should be a valid string [type=string_type, input_value=1, input_type=int]\n For further information visit https://errors.pydantic.dev/2.5/v/string_type", "message": "Validation error" }</pre>
404	<pre>"message": "User does not exist"</pre>



400	<pre>{ "message": "Missing camera_id in the request" }</pre>
500	<pre>"message": "Failed to delete camera", "error"</pre>

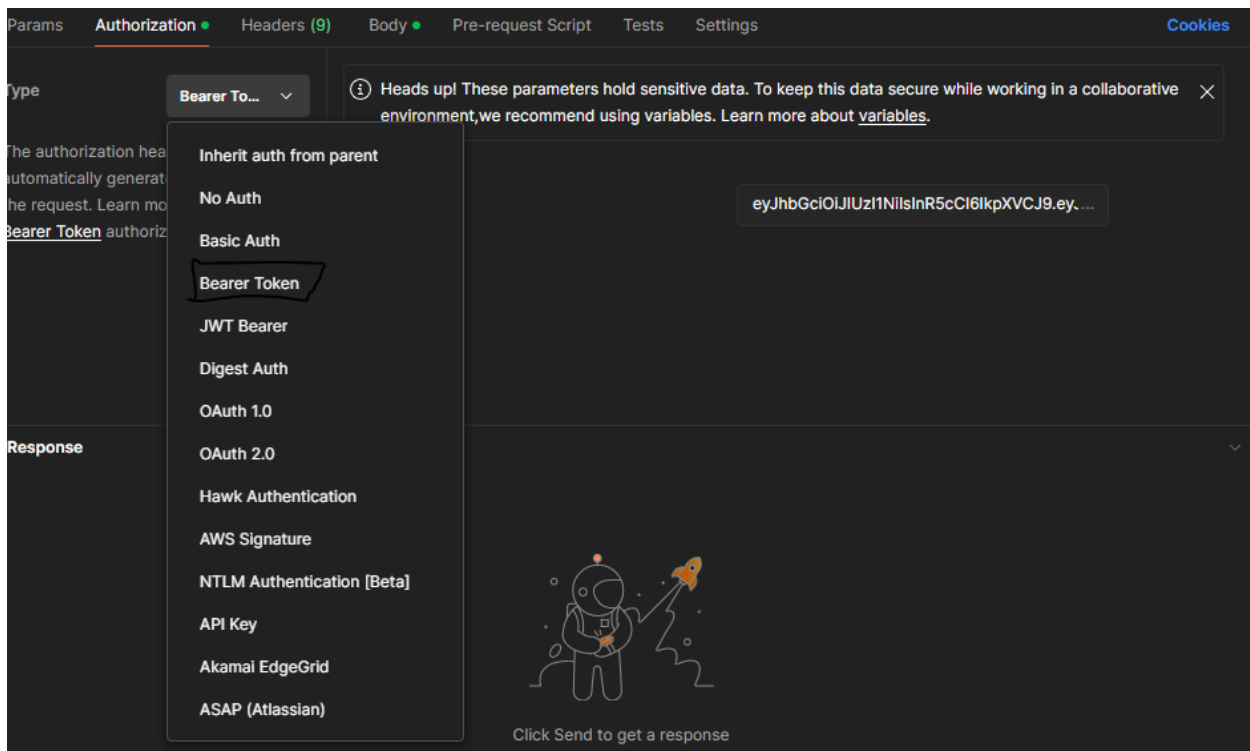
6. Get user all cameras

This endpoint retrieves a list of cameras for the user.

Method	URL
GET	/user/cameras/get

Type	Params	Values
Authorization	Bearer_Token	Token

Authorization:



Params **Authorization** Headers (9) Body Pre-request Script Tests Settings Cookies

Type **Bearer To...** Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about [variables](#).

The authorization header is automatically generated from the request. Learn more about [Bearer Token](#) authorization.

Response

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ...

Click Send to get a response

Responses

Status	Response
200	<pre>{ "camera_id": 90, "camera_ip": "192.133.1.1", "camera_mode": "out", "camera_name": "Door_camera", "confidence_threshold": 0.5, "raspberrypi_id": null, "userid": 15 }, { "camera_id": 94, "camera_ip": "193.133.1.2", "camera_mode": "out", "camera_name": "Door_camera", "confidence_threshold": 0.5, "raspberrypi_id": "12321", "userid": 15 }], "message": "Cameras fetched successfully" }</pre>
401	<pre>{ "message": "Authorization header is missing" }</pre>
404	<pre>{ "message": "No cameras found for the user" }</pre>

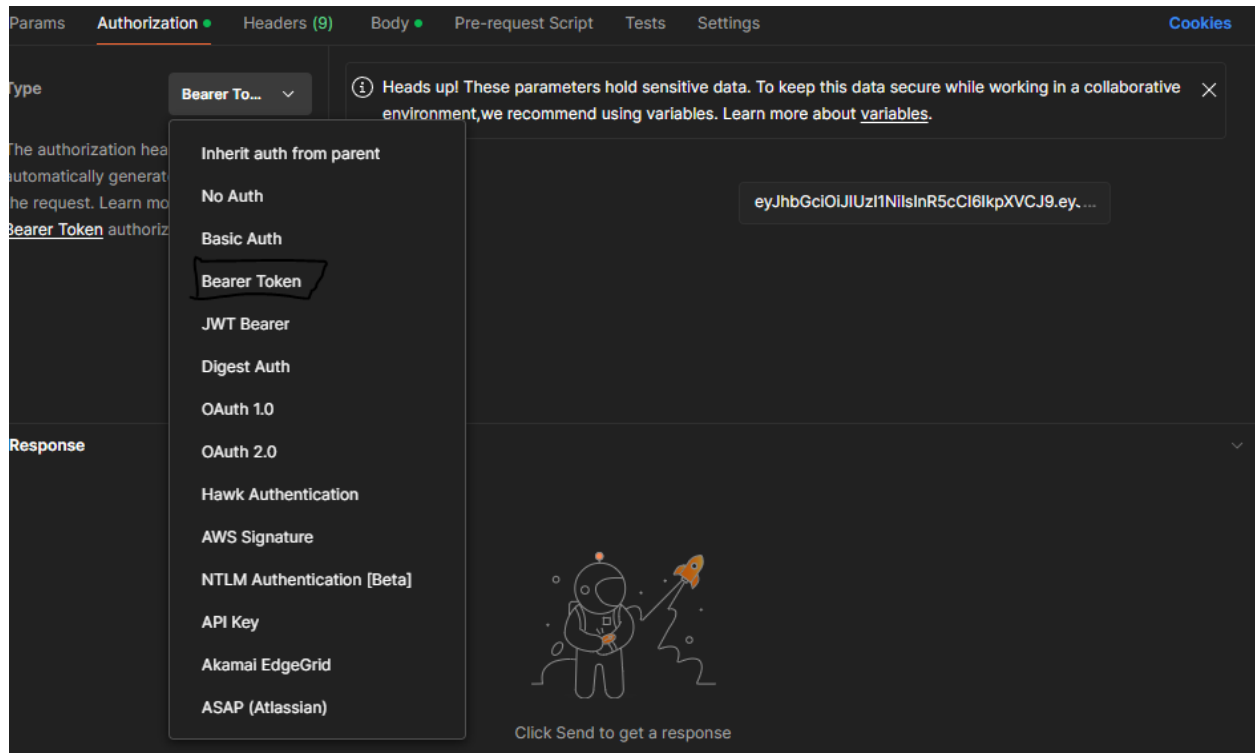
7. Get user info

This endpoint is used to retrieve user information.

Request

Method	URL
GET	/user/get_info

Type	Params	Values
Authorization	Bearer_Token	Token



8. Web socket

WebSocket

The WebSocket API provides real-time communication for retrieving license plate information

Connection Details

WebSocket URL: `ws://our api url`

Authentication

To connect to the WebSocket, include the following headers in the connection request:

- `X-My-Auth`: Bearer Token
- `X-Camera-Id`: Camera ID (integer)



Example :

MessageEvents (1)ParamsHeadersSettings

HeadersHide auto-generated headers

	Key	Value	Description	...	Bulk Edit	Presets
<input checked="" type="checkbox"/>	Host	<calculated at runtime>				
<input checked="" type="checkbox"/>	Connection	Upgrade				
<input checked="" type="checkbox"/>	Upgrade	websocket				
<input checked="" type="checkbox"/>	Sec-WebSocket-Key	<calculated at runtime>				
<input checked="" type="checkbox"/>	Sec-WebSocket-Version	13				
<input checked="" type="checkbox"/>	Sec-WebSocket-Extensions	permessage-deflate; client_max_window_bits				
<input checked="" type="checkbox"/>	X-My-Auth	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImppZWFnJ2EzLn0.c...				
<input checked="" type="checkbox"/>	X-Camera-Id	66				
	Key	Value	Description			

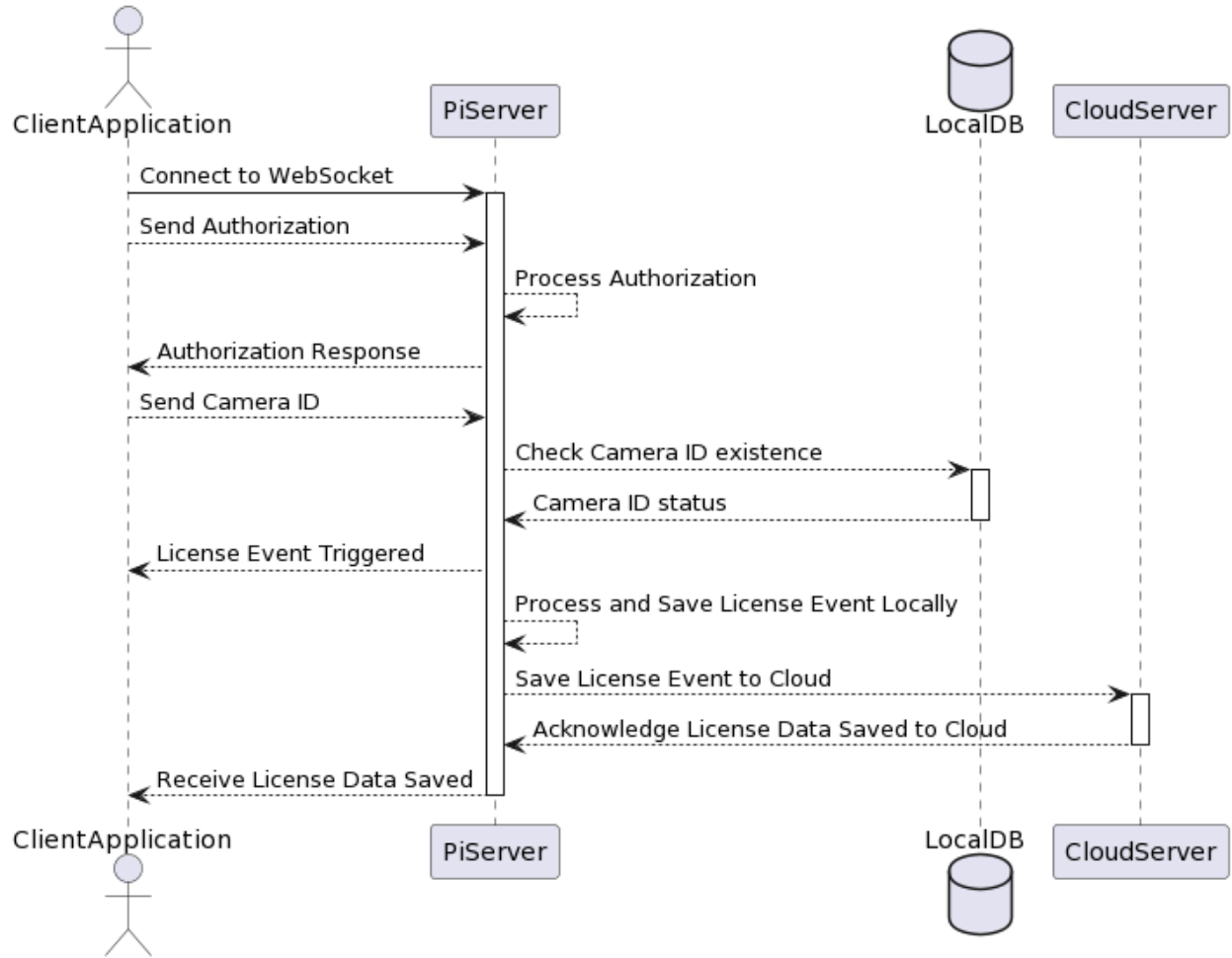
Supported Events

Listen to Event

- Event: `license`

- Description: Triggered when a camera detects a car.

```
- Payload Received in the front end: (
    'license_type',
    'plate_in_arabic':
    'plate_in_english':
    'orientation' :,
    'request_datetime' :,
    'photo_data':,
    'userid' :,
    'camera_name':,
    'camera_id' :
)
```



Error code:

HTTP Status Code	Meaning
200	OK - Successful request
400	Bad Request - Invalid syntax or parameters

401	Unauthorized - Authentication or permission failure
404	Not Found - Requested resource not found
422	Unprocessable Entity - Semantic errors in the request
500	Internal Server Error - Unexpected server error