Input Validation and Sanitization

First, make sure you have Flask installed. If not, you can install it using pip:

```
pip install Flask
```

Now, create a Python file for the plugin, which named `input_validator.py`:

```
from flask import request, jsonify

class InputValidator:
    @staticmethod
    def validate_input():
        """
        Middleware function to validate and sanitize input data in incoming requests.
        """
        if request.method in ['POST', 'PUT', 'PATCH']:
            data = request.get_json()

            # Define validation rules for your input fields
            validation_rules = {
                'username': {
                    'required': True,
                    'type': str,
                    'min_length': 3,
                    'max_length': 50
                },
                'email': {
                    'required': True,
                    'type': str,
                    'regex': r'^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$'
                },
                # Add more fields and validation rules as needed
            }

            for field, rules in validation_rules.items():
                if field not in data and rules.get('required', False):
                    return jsonify({'error': f'Missing required field: {field}'}), 400

                value = data.get(field)
                if value is not None:
                    if 'type' in rules and not isinstance(value, rules['type']):
                        return jsonify({'error': f'Invalid data type for field {field}'}), 400

                    if 'min_length' in rules and len(value) < rules['min_length']:
                        return jsonify({'error': f'Field {field} is too short'}), 400

                    if 'max_length' in rules and len(value) > rules['max_length']:
                        return jsonify({'error': f'Field {field} is too long'}), 400

                    if 'regex' in rules and not re.match(rules['regex'], value):
                        return jsonify({'error': f'Invalid format for field {field}'}), 400

        return None
```

Next, you need to integrate this middleware into your Flask application.

```
from flask import Flask
from input_validator import InputValidator

app = Flask(__name__)

# Initialize the input validator middleware
app.before_request(InputValidator.validate_input)

# Define your routes and other application logic below

if __name__ == '__main__':
    app.run()
```

This code sets up a Flask application and uses the `before_request` method to apply the `validate_input` middleware to all incoming requests. You can define your routes and other application logic below this setup.

Remember to customize the `validation_rules` dictionary in the `InputValidator` class to match the specific input fields and validation criteria for your web application. This is just a basic example to get you started, and you can expand upon it as needed.