```
from typing import Any, Callable, Dict, List, Optional
from pydantic import BaseModel, Field
from pydantic.v1 import validator


class AgentSchema(BaseModel):
    llm: Any = Field(..., description="The language model to use")
    max_tokens: int = Field(
        ..., description="The maximum number of tokens", ge=1
    )
    context_window: int = Field(
        ..., description="The context window size", ge=1
    )
    user_name: str = Field(..., description="The user name")
    agent_name: str = Field(..., description="The name of the agent")
    system_prompt: str = Field(..., description="The system prompt")
    template: Optional[str] = Field(default=None)
    max_loops: Optional[int] = Field(default=1, ge=1)
    stopping_condition: Optional[Callable[[str], bool]] = Field(
        default=None
    )
    loop_interval: Optional[int] = Field(default=0, ge=0)
    retry_attempts: Optional[int] = Field(default=3, ge=0)
    retry_interval: Optional[int] = Field(default=1, ge=0)
    return_history: Optional[bool] = Field(default=False)
    stopping_token: Optional[str] = Field(default=None)
```

dynamic_loops: Optional[bool] = Field(default=False)

interactive: Optional[bool] = Field(default=False)

dashboard: Optional[bool] = Field(default=False)

agent_description: Optional[str] = Field(default=None)

tools: Optional[List[Callable]] = Field(default=None)

dynamic_temperature_enabled: Optional[bool] = Field(default=False)

sop: Optional[str] = Field(default=None)

sop_list: Optional[List[str]] = Field(default=None)

saved_state_path: Optional[str] = Field(default=None)

autosave: Optional[bool] = Field(default=False)

self_healing_enabled: Optional[bool] = Field(default=False)

code_interpreter: Optional[bool] = Field(default=False)

multi_modal: Optional[bool] = Field(default=False)

pdf_path: Optional[str] = Field(default=None)

list_of_pdf: Optional[str] = Field(default=None)

tokenizer: Optional[Any] = Field(default=None)

long_term_memory: Optional[Any] = Field(default=None)

preset_stopping_token: Optional[bool] = Field(default=False)

traceback: Optional[Any] = Field(default=None)

traceback_handlers: Optional[Any] = Field(default=None)

streaming_on: Optional[bool] = Field(default=False)

docs: Optional[List[str]] = Field(default=None)

docs_folder: Optional[str] = Field(default=None)

verbose: Optional[bool] = Field(default=False)

parser: Optional[Callable] = Field(default=None)

best_of_n: Optional[int] = Field(default=None)

```
callback: Optional[Callable] = Field(default=None)
metadata: Optional[Dict[str, Any]] = Field(default=None)
callbacks: Optional[List[Callable]] = Field(default=None)
logger_handler: Optional[Any] = Field(default=None)
search_algorithm: Optional[Callable] = Field(default=None)
logs_to_filename: Optional[str] = Field(default=None)
evaluator: Optional[Callable] = Field(default=None)
output_json: Optional[bool] = Field(default=False)
stopping_func: Optional[Callable] = Field(default=None)
custom_loop_condition: Optional[Callable] = Field(default=None)
sentiment_threshold: Optional[float] = Field(default=None)
custom_exit_command: Optional[str] = Field(default="exit")
sentiment_analyzer: Optional[Callable] = Field(default=None)
limit_tokens_from_string: Optional[Callable] = Field(default=None)
custom_tools_prompt: Optional[Callable] = Field(default=None)
tool_schema: Optional[Any] = Field(default=None)
output_type: Optional[Any] = Field(default=None)
function_calling_type: Optional[str] = Field(default="json")
output_cleaner: Optional[Callable] = Field(default=None)
function_calling_format_type: Optional[str] = Field(
    default="OpenAI"
)
list_base_models: Optional[List[Any]] = Field(default=None)
metadata_output_type: Optional[str] = Field(default="json")
state_save_file_type: Optional[str] = Field(default="json")
chain_of_thoughts: Optional[bool] = Field(default=False)
```

```
algorithm_of_thoughts: Optional[bool] = Field(default=False)
tree_of_thoughts: Optional[bool] = Field(default=False)
tool_choice: Optional[str] = Field(default="auto")
execute_tool: Optional[bool] = Field(default=False)
rules: Optional[str] = Field(default=None)
planning: Optional[bool] = Field(default=False)
planning_prompt: Optional[str] = Field(default=None)
device: Optional[str] = Field(default=None)
custom_planning_prompt: Optional[str] = Field(default=None)
memory_chunk_size: Optional[int] = Field(default=2000, ge=0)
agent_ops_on: Optional[bool] = Field(default=False)
log_directory: Optional[str] = Field(default=None)
project_path: Optional[str] = Field(default=None)
tool_system_prompt: Optional[str] = Field(
    default="tool_sop_prompt()"
)
top_p: Optional[float] = Field(default=0.9, ge=0, le=1)
top_k: Optional[int] = Field(default=None)
frequency_penalty: Optional[float] = Field(
    default=0.0, ge=0, le=1
)
presence_penalty: Optional[float] = Field(default=0.0, ge=0, le=1)
temperature: Optional[float] = Field(default=0.1, ge=0, le=1)

@validator(
    "tools",
```

```
    "docs",
    "sop_list",
    "callbacks",
    "list_base_models",
    each_item=True,
)
def check_list_items_not_none(cls, v):
    if v is None:
        raise ValueError("List items must not be None")
    return v

@validator(
    "tokenizer",
    "memory",
    "traceback",
    "traceback_handlers",
    "parser",
    "callback",
    "search_algorithm",
    "evaluator",
    "stopping_func",
    "custom_loop_condition",
    "sentiment_analyzer",
    "limit_tokens_from_string",
    "custom_tools_prompt",
    "output_cleaner",
```

```
    )
    def check_optional_callable_not_none(cls, v):
        if v is not None and not callable(v):
            raise ValueError(f"{v} must be a callable")
        return v



# # Example of how to use the schema
# agent_data = {
#     "llm": "OpenAIChat",
#     "max_tokens": 4096,
#     "context_window": 8192,
#     "user_name": "Human",
#     "agent_name": "test-agent",
#     "system_prompt": "Custom system prompt",
# }

# agent = AgentSchema(**agent_data)
# print(agent)
```