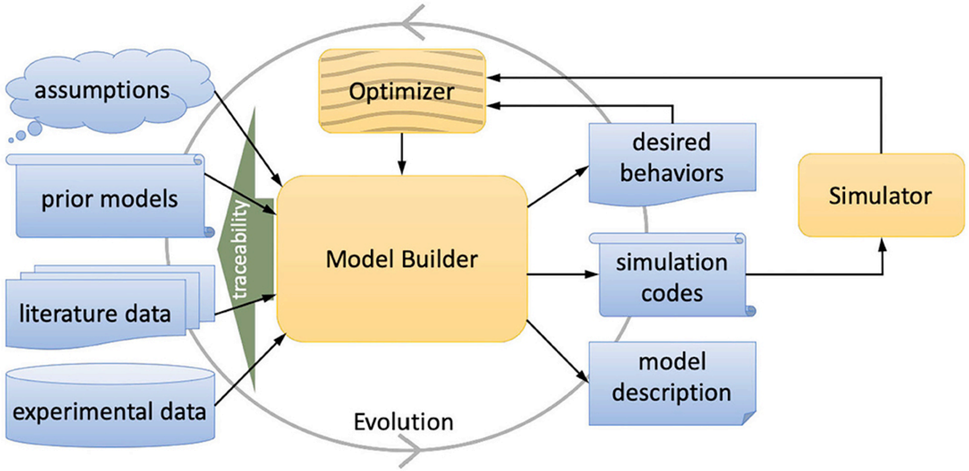
Building BrainModeling Workflows

A proof-of-concept framework



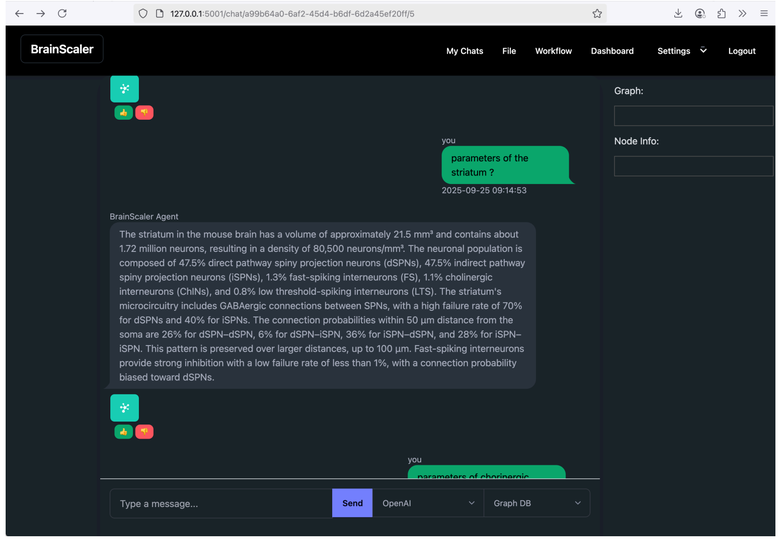
OIST

Carlos Gutierrez, Kenji Doya, Neuro-workflow dev.team

Sept. 28 2025

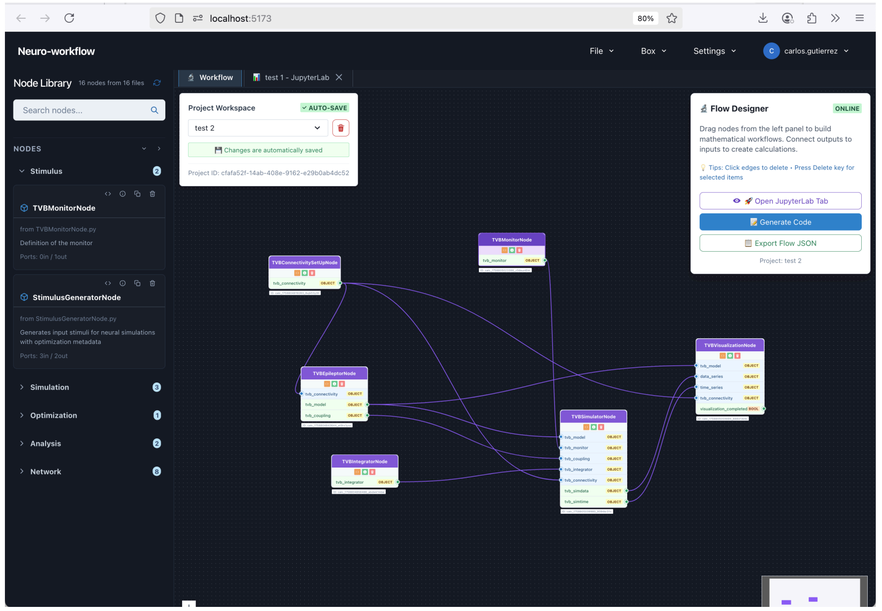
Prototypes - Proof-of-Concept

Chatbot



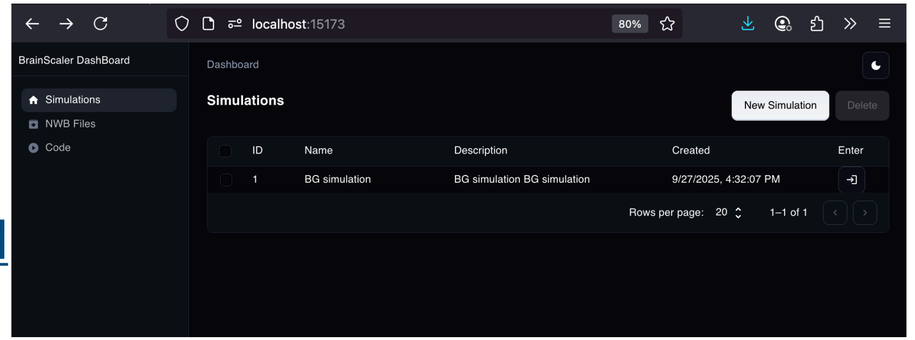
https://github.com/oist/brainscaler_frontend

Neuro-workflow



https://github.com/oist/neuro-workflow

Dashboard



https://github.com/oist/brainscaler_dashboard

Access: send your GitHub user (or GitHub linked email) to —>> carlos.gutierrez@oist.jp

Prototypes - Proof-of-Concept

1 - Clone the repositories in your local laptop/PC

2 - Install instructions

Chatbot

Check “Installation in Docker”

https://github.com/oist/brainscaler_frontend/blob/main/README.md

NeuroWorkflow

https://github.com/oist/neuro-workflow/blob/main/gui/README.md

Dashboard

https://github.com/oist/brainscaler_dashboard/blob/main/README.md

3 - Install instructions include .env files set-up.
Please use the files at the link below and rename each of them as .env

https://www.dropbox.com/scl/fo/vytgfc7gsz2kcte0b55n5/AJg4RYR0fjFNyrD2uHCe870?rlkey=1o28ybwx93ccrqi3afje9i2fa&st=7wiuw86a&dl=0

Prototypes - Proof-of-Concept

NeuroWorkflow

IPs addresses

Chatbot

https://github.com/oist/brainscaler_frontend

http://0.0.0.0:5001

NeuroWorkFlow

https://github.com/oist/neuro-workflow/tree/main/gui

http://localhost:5173/

Django project folder

/Users/carlosengutierrez/Downloads/neuro-workflow/gui/workflow_backend/django-project

Dashboard

https://github.com/oist/brainscaler_dashboard/tree/main

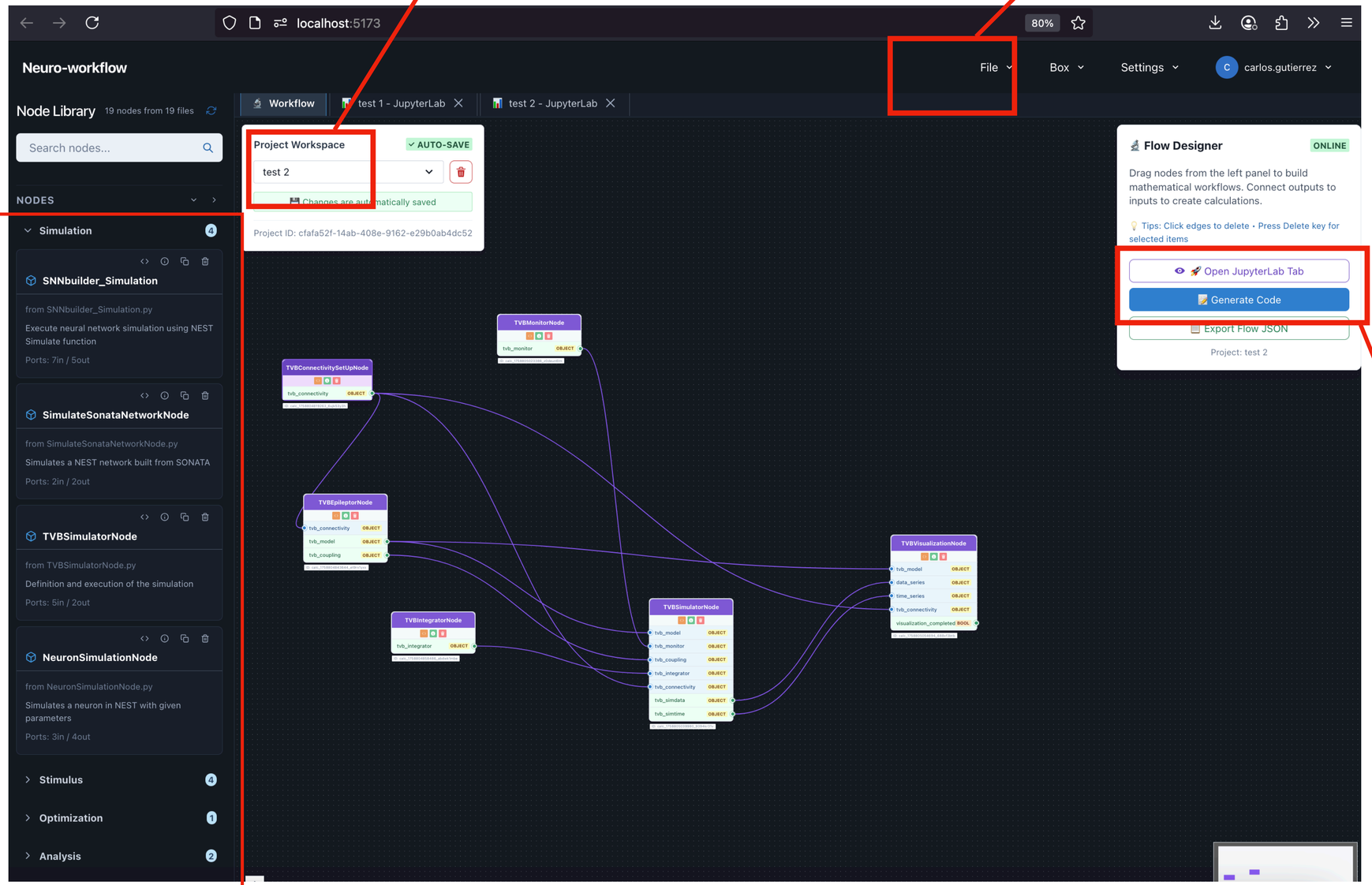
http://localhost:15173/

Prototypes - Proof-of-Concept

NeuroWorkflow

2 - Select your Project

1 - Create a Project (a workflow)



Available nodes

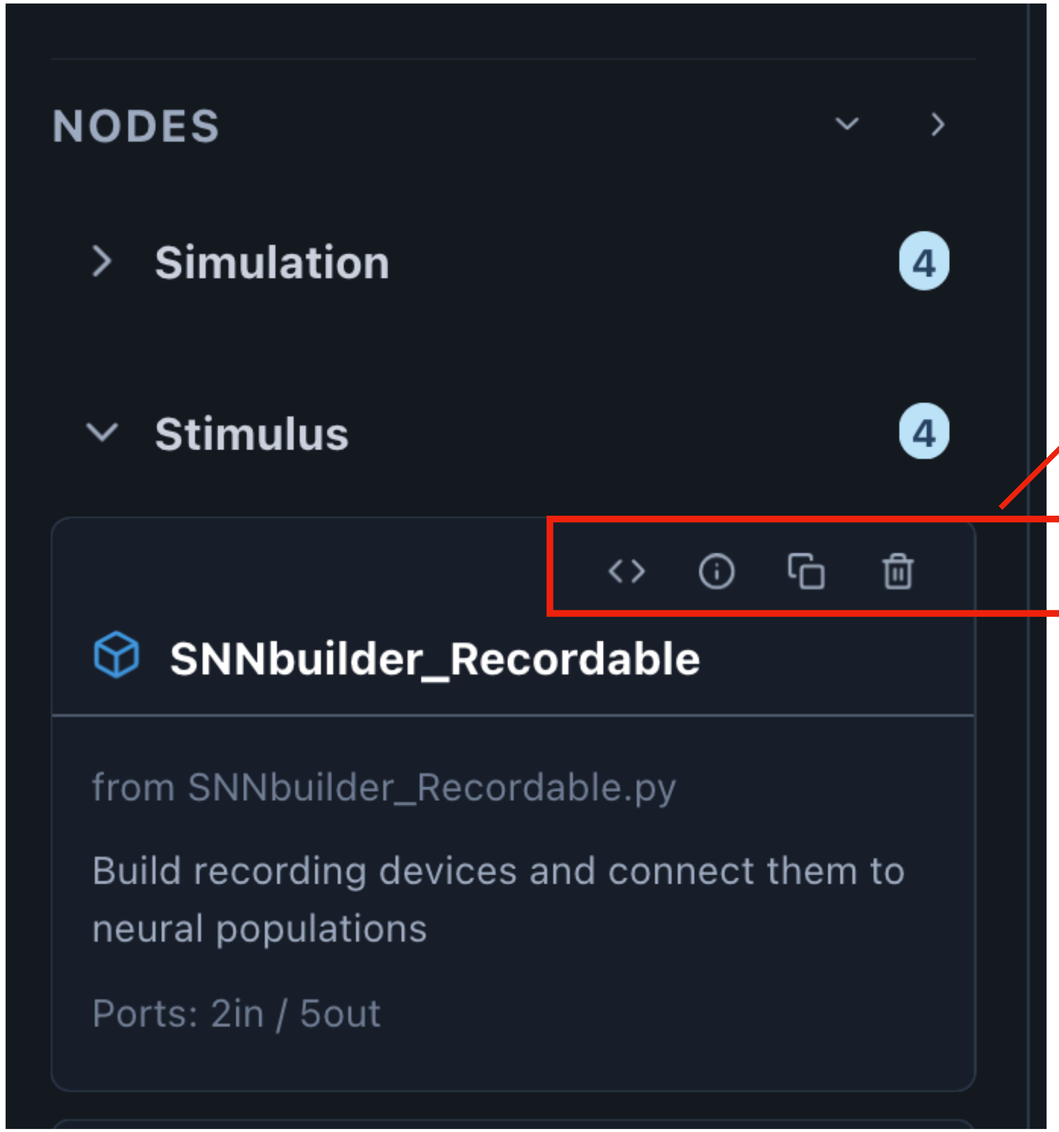
Node Schema:
https://github.com/oist/neuro-workflow/blob/main/NODE_SCHEMA.md

Custom node docs:
https://github.com/oist/neuro-workflow/blob/main/CUSTOM_NODE_TUTORIAL.md

3 - Run a workflow
*First Generate Code
*And next Open Jupyter Lab

Prototypes - Proof-of-Concept

NeuroWorkflow



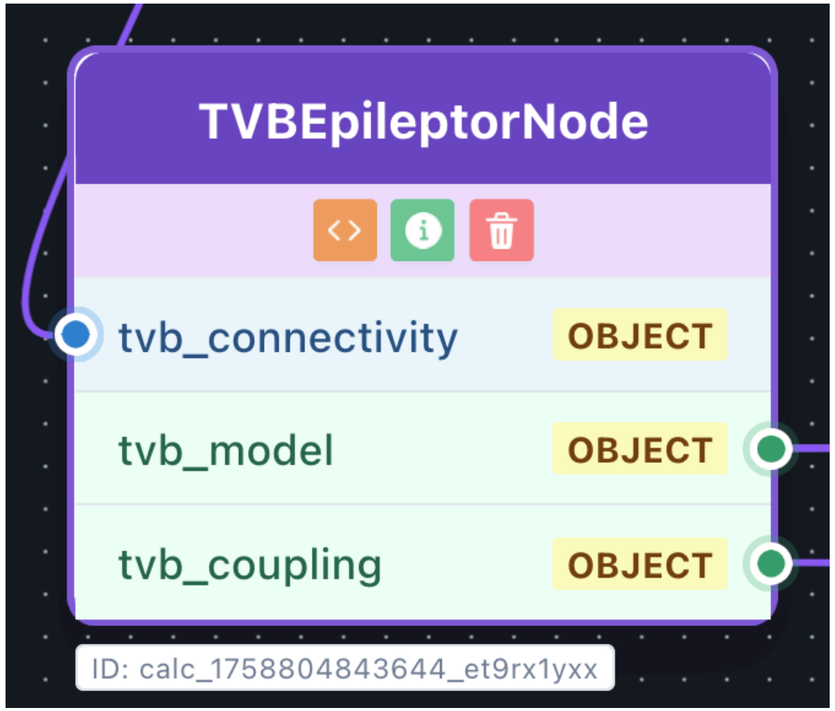
- Examine the custom node

- The node Information (schema)

- Duplicate the node with a different name (If you want to change or create a new node based on the current node)

Prototypes - Proof-of-Concept

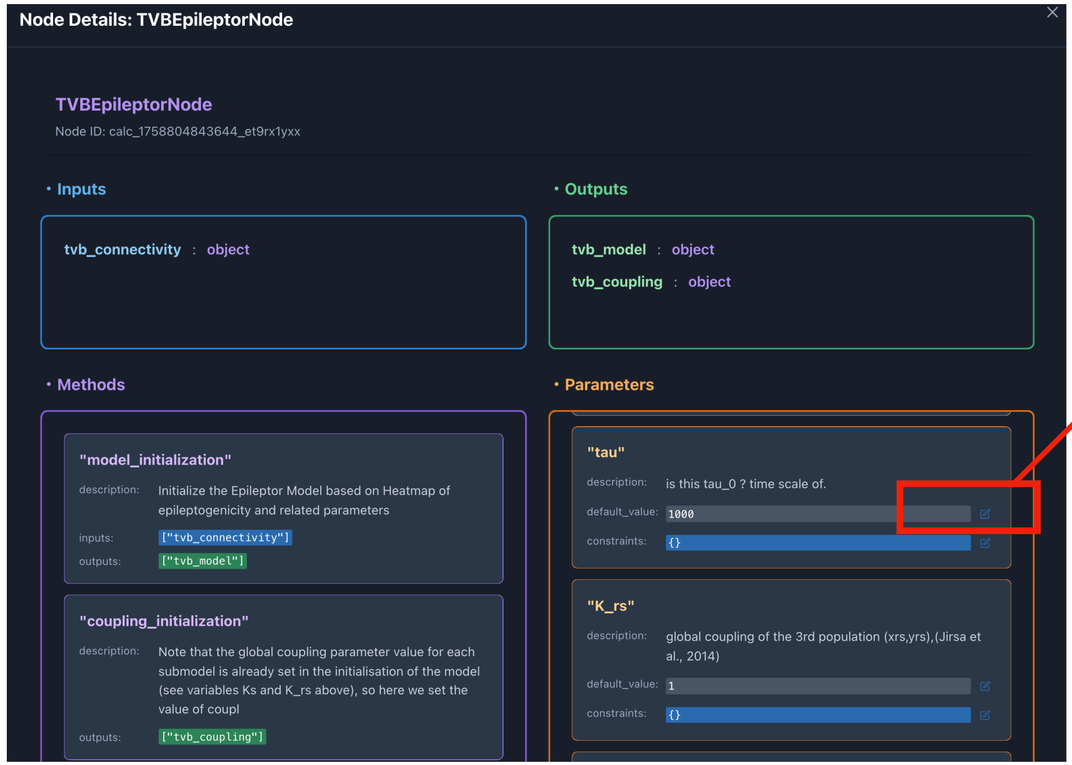
NeuroWorkflow



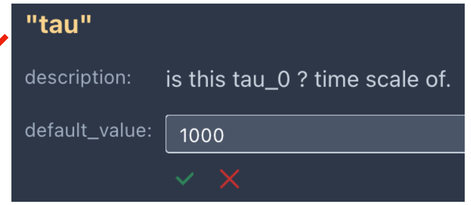
- Examine the custom node

- The node Information (schema)

Use this to UPDATE parameters of the node (configure)



Update the parameter “tau”



Prototypes - Proof-of-Concept

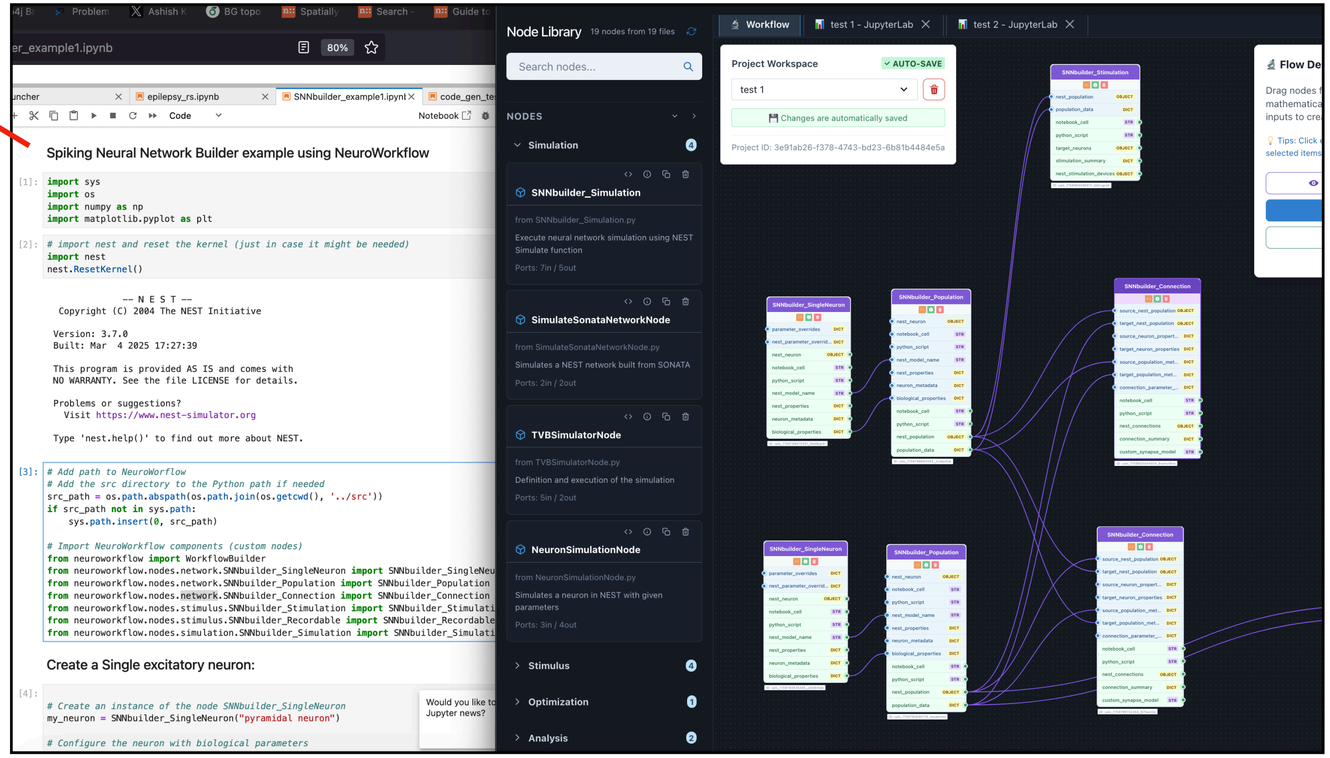
NeuroWorkflow

Examples for the Hackathon at Folder Notebook

https://github.com/oist/neuro-workflow/blob/main/README.md

- `epilepsy_rs.ipynb` - Interactive example of epileptic resting state using the virtual brain TVB
- `SNNbuilder_example1.ipynb` - Interactive example of Spiking Neural Network building using SNNbuilder custom nodes

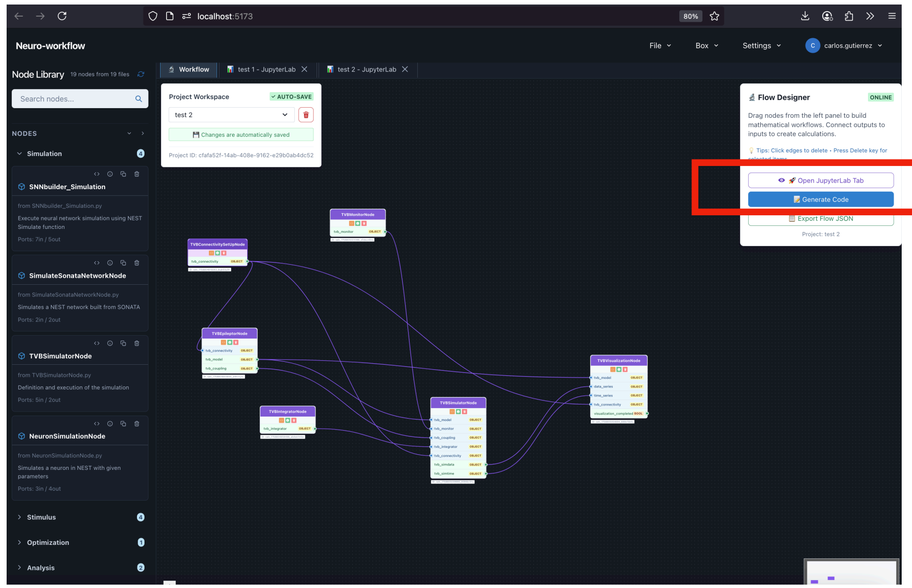
Go through the .ipynb example and build the workflow using the graphical interface.



Prototypes - Proof-of-Concept

NeuroWorkflow

Examples for the Hackathon at Folder Notebook



*First Generate Code
*And next Open Jupyter Lab

The code generation has some bugs.
We prepared “reviewed” generated code for both examples:

https://www.dropbox.com/scl/fo/o859x7s580oakliod9h5u/AHULmFzrgvYxcXdlHohDT0o?rlkey=27bh0br5bfhfeqxg9tcna2sxf&st=lyhr4rvl&dl=0

*SNNbuilder_example1.ipynb —> Test1_reviewed.ipynb

*epilepsy_rs.ipynb —> Test2_reviewed.ipynb

Jupyter credentials:
user1
password

Prototypes - Proof-of-Concept

NeuroWorkflow

Challenge: build nodes for your model (or any model).

(We are working on LLM based conversion: unstructured python —> nodes)

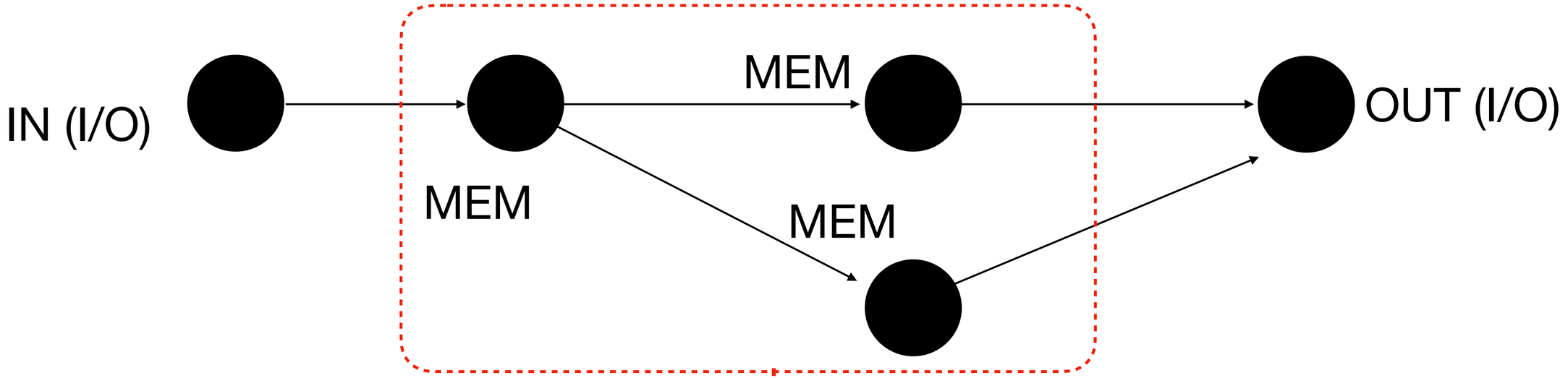
Node Schema:
https://github.com/oist/neuro-workflow/blob/main/NODE_SCHEMA.md

Custom node docs:
https://github.com/oist/neuro-workflow/blob/main/CUSTOM_NODE_TUTORIAL.md

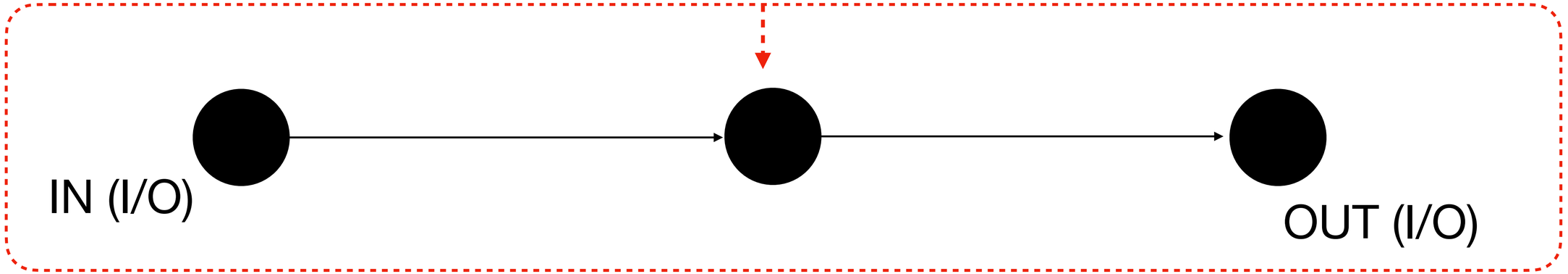
NeuroWorkflow

(Node - Edges)

Representation for Brain Models
Interpretability/Usability and AI-readiness



Execution on cluster HPC (i.e: **representation as SnakeMake Rules**)

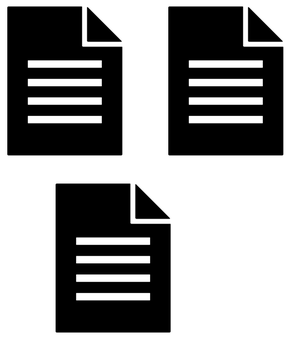


Scaling-up Nodes Generation

NeuroWorkflow

GUI

Brain Model



Python files
(Unstructured)

LLM

Node generation

Template Node
(python class schema)

Node definition - Schema

```python
from neuroworkflow.core.node import Node
from neuroworkflow.core.schema import NodeDefinitionSchema, PortDefinition, ParameterDefinition,
MethodDefinition
from neuroworkflow.core.port import PortType

class MyNode(Node):
    """Brief description of node functionality."""

    NODE_DEFINITION = NodeDefinitionSchema(
        type='node_category',
        description='What this node does',

        parameters={
            'param1': ParameterDefinition(
                default_value=1.0,
                description='Parameter description',
                optimizable=True,
                optimization_range=[0.1, 10.0]
            ),
        },

        inputs={
            'input_data': PortDefinition(
                type=PortType.OBJECT,
                description='Input description'
            ),
        },

        outputs={
            'output_data': PortDefinition(
                type=PortType.OBJECT,
                description='Output description'
            ),
        },

        methods={
            'process_method': MethodDefinition(
                description='Main processing method',
                inputs=['input_data'],
```

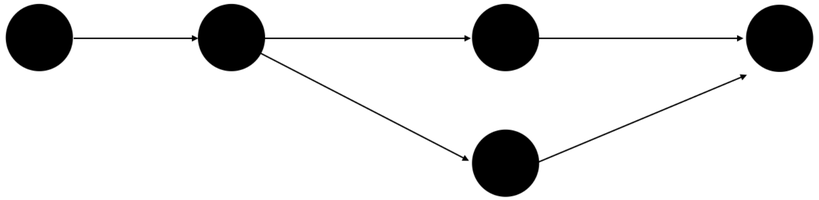
Parameters

Input ports

Output ports

Methods

Brain Model



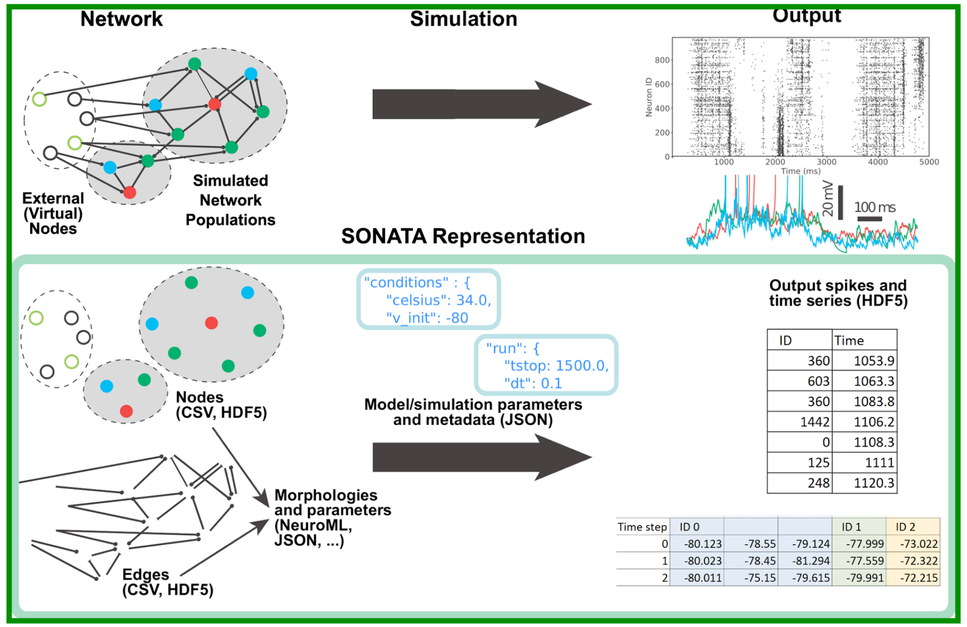
Nodes and connections

Curation

GitHub repository

SONATA:

SONATA is a standardized format for large-scale neural network models jointly developed by the Allen Institute for Brain Science and the Blue Brain Project. It's designed for:



ID	Time
360	1053.9
603	1063.3
360	1083.8
1442	1106.2
0	1108.3
125	1111
248	1120.3

Time step	ID 0			ID 1	ID 2
0	-80.123	-78.55	-79.124	-77.999	-73.022
1	-80.023	-78.45	-81.294	-77.559	-72.322
2	-80.011	-75.15	-79.615	-79.991	-72.215

Key Components:

- **Internal nodes**: Network neurons with 3D coordinates and properties
- **External nodes**: Input/stimulus sources
- **Internal-internal edges**: Synapses between network neurons
- **External-internal edges**: Input connections
- **Spike trains**: External input data

Example in NEST:

https://github.com/oist/neuro-workflow/blob/main/notebooks/01_Basic_Simulation.ipynb

Basic Neural Simulation with NeuroWorkflow

This notebook demonstrates how to create a simple neural simulation workflow using the NeuroWorkflow library. We'll build a workflow that:

1. Creates a neural network from SONATA format
2. Simulates the network using NEST

Setup

First, let's make sure we can import the NeuroWorkflow library. If you've installed it with pip, you can import it directly. Otherwise, we'll add the source directory to the Python path.

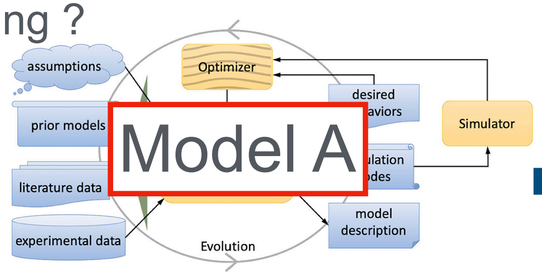
```
import sys
import os
import numpy as np
import matplotlib.pyplot as plt

# Add the src directory to the Python path if needed
src_path = os.path.abspath(os.path.join(os.getcwd(), '../src'))
if src_path not in sys.path:
    sys.path.insert(0, src_path)

# Import NeuroWorkflow components
from neuroworkflow import WorkflowBuilder
from neuroworkflow.nodes.network.BuildSonataNetworkNode import BuildSonataNetworkNode
```

SONATA: What we would like to do ?

BM2 modeling platform

Down size scaling ?

SONATA representation
(External Large model
Data / Model A)

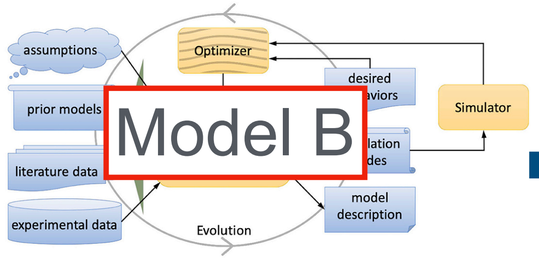


scaling up ?

SONATA representation
(Large model **Model A'**)

A potential solution :
build custom nodes in
neuro-workflow for
BMTK, NEST support
sonata.

Explore, Change Parameter,
Optimize, Experiment,
Change the model, add components,
(Real size, Scaled down size?)



scaling up ?

SONATA representation
(Large model **Model B**)

A potential solution : build
custom nodes in neuro-workflow
for SNNbuilder-SONATA
conversion

Explore, Change Parameter,
Optimize, Experiment,
Change the model, add components,
(Real size, Scaled down size?)

SONATA:

Current State:

- SNNbuilder: Custom workflow-based approach with components like populations, stimuli, connections, recording, simulation
- SONATA: Standardized format with internal/external nodes, edges, configuration files
- Both represent the same underlying concepts but with different structures and semantics

SNNbuilder → SONATA mappings:

- SNNbuilder_Population → SONATA internal nodes
- SNNbuilder_Stimulation → SONATA external nodes + input spike trains
- SNNbuilder_Connection → SONATA internal-internal edges or external-internal edges
- SNNbuilder_Recordable → SONATA reports configuration
- SNNbuilder_Simulation → SONATA simulation configuration

Data Structure Differences: SNNbuilder uses NEST-native objects, SONATA uses HDF5/CSV

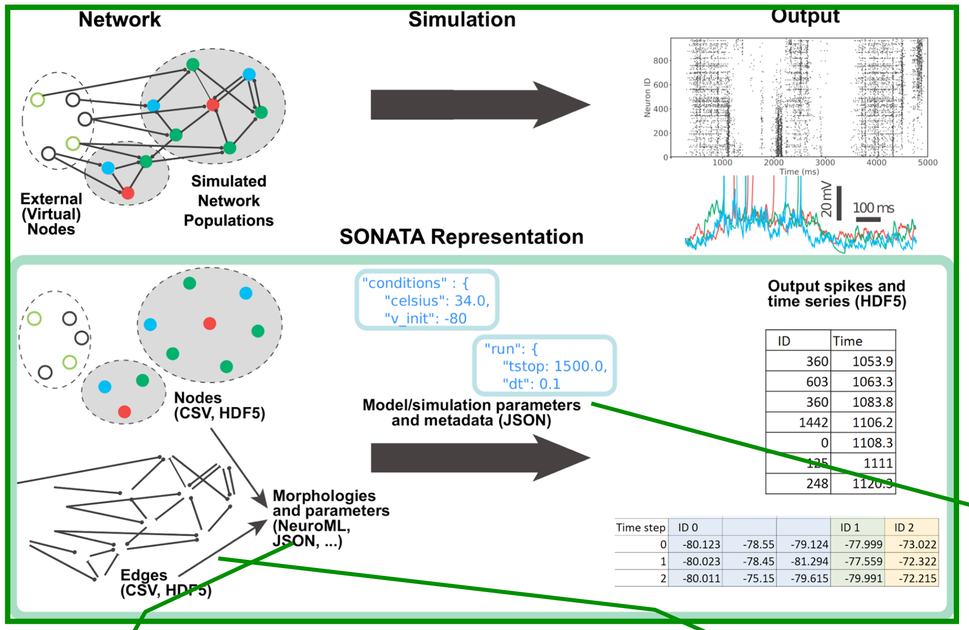
Bidirectional Conversion: Need both SNNbuilder→SONATA and SONATA→SNNbuilder

SONATA:

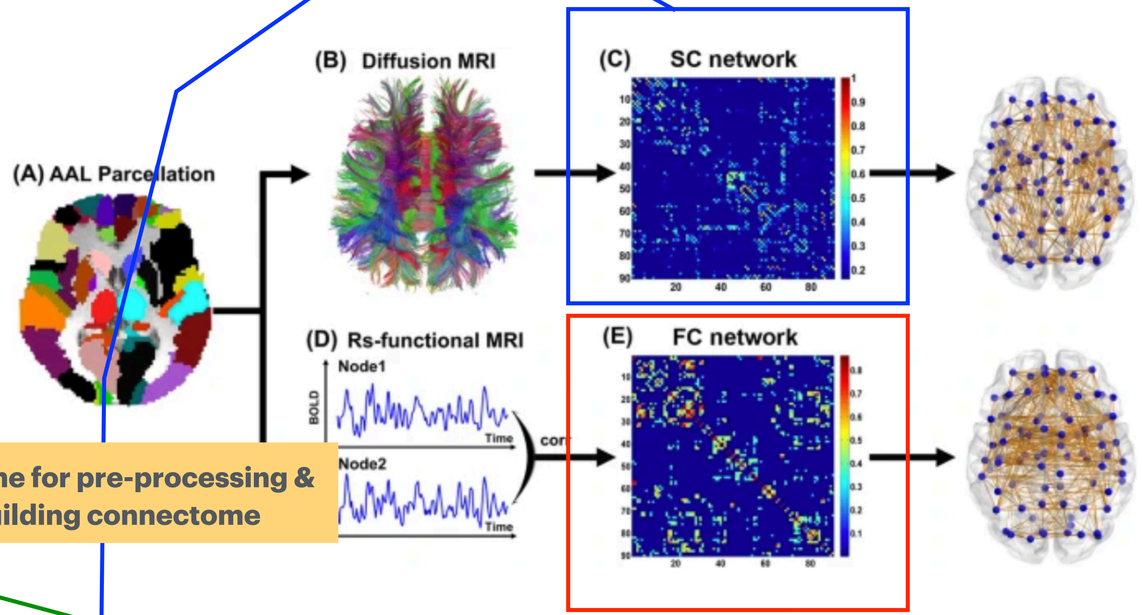
Transformations Nodes from NeuroWorkflow to Sonata:

- **Connection Rules** → Explicit connectivity matrices with proper indexing
- **Parametric Stimuli** → Pre-generated spike trains with timing preservation
- **Spatial Structures** → 3D coordinates with extent calculations
- **Model Parameters** → Comprehensive neuron/synapse parameter mappings

SONATA: Example of Data-to-Model Mapping



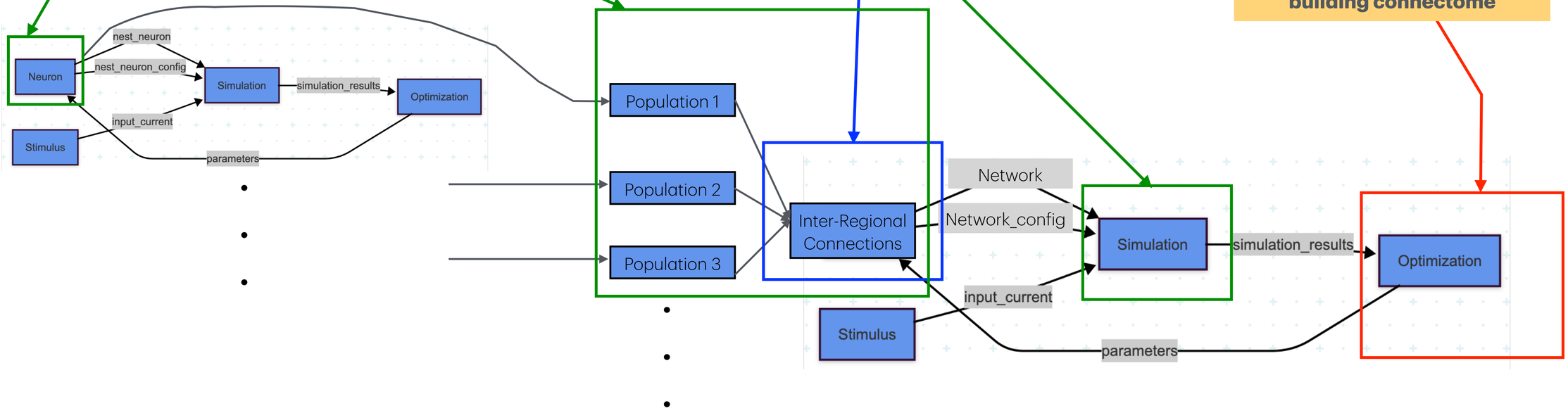
Data in SONATA



Connectomme

Pipeline for pre-processing & building connectome

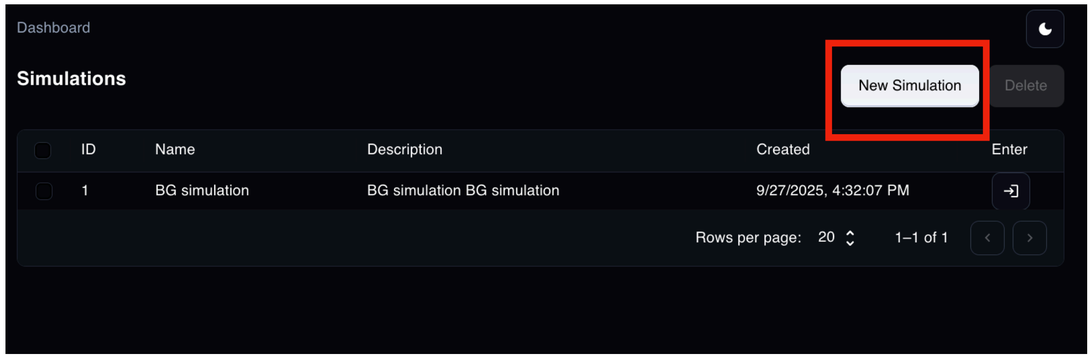
Pipeline for pre-processing & building connectome

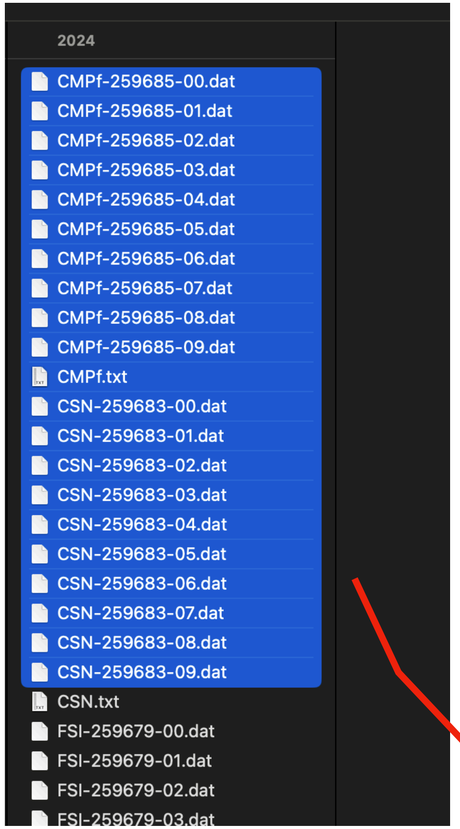


Dashboard

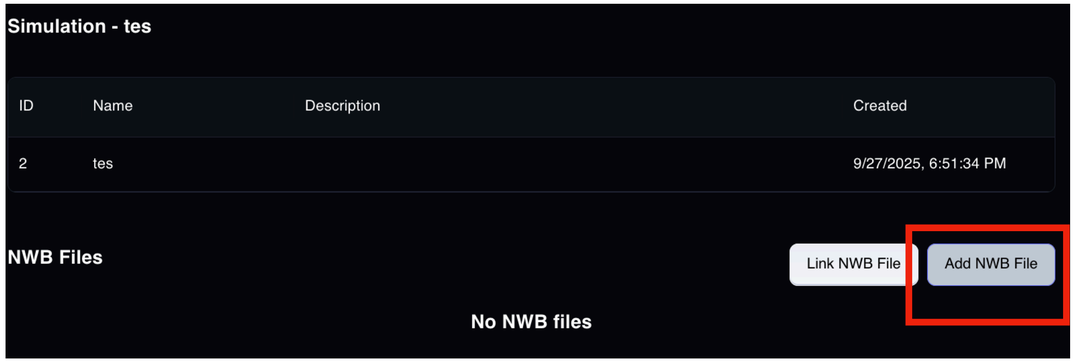
Examples files (NEST simulator output spiking data) here:
(please use only the .DAT files)
https://www.dropbox.com/scl/fo/k5psx0lgsw6dhjr2tk2hz/ADXp6PWMp9Fde7BbJvtKhn0?rlkey=13b0zy7y1oyerb6r75e37exbs&st=bx157kd6&dl=0

1 - Create a simulation record

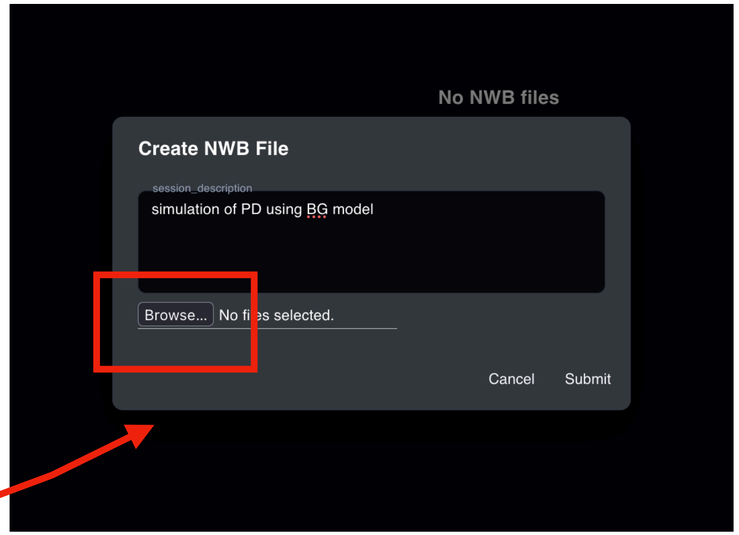




2 - Select it and Add NWB file



3 - upload dat files (NEST)



(Automatic conversion to NWB)

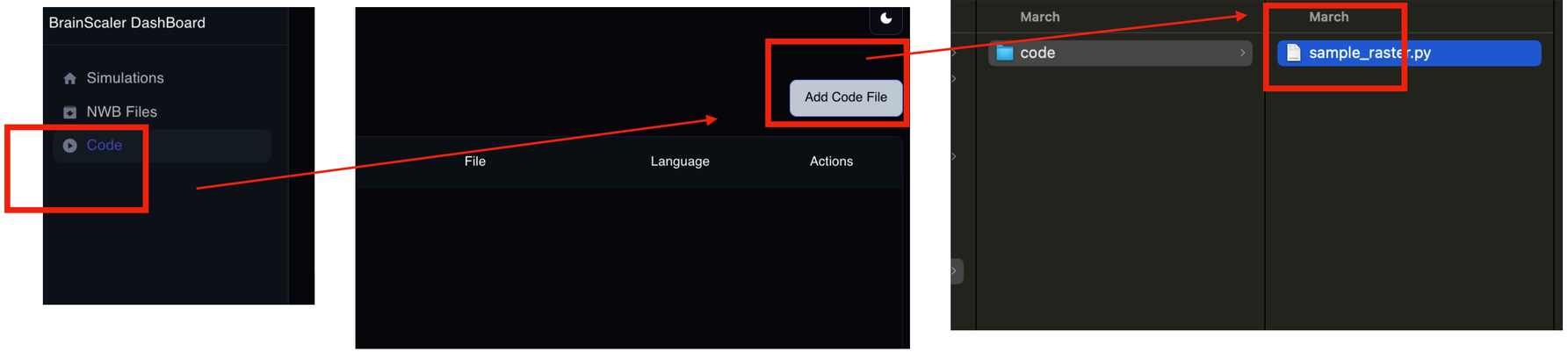
Operation manual—> https://github.com/oist/brainscaler_dashboard/blob/main/doc/operation_manual.pdf

Dashboard

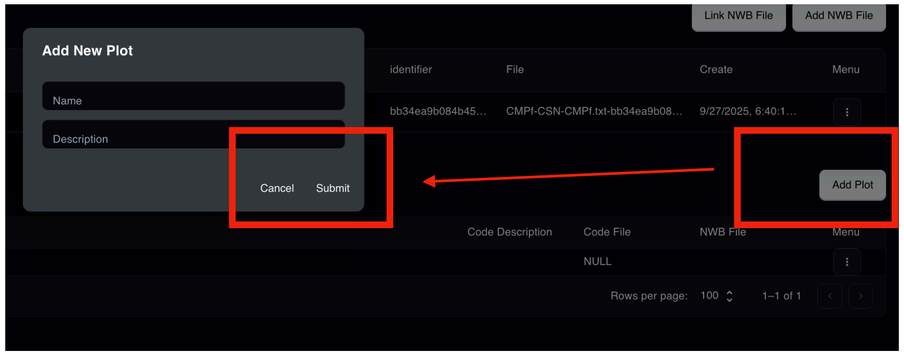
Sample plot code (lets develop more plots !!)

https://github.com/oist/brainscaler_dashboard/blob/main/sample/code/sample_raster.py

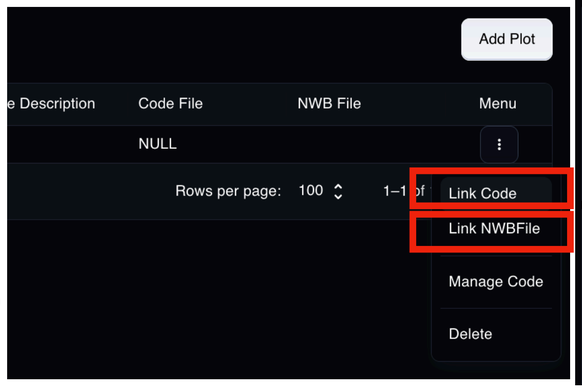
4 - go to code menu to upload a python code of a plot



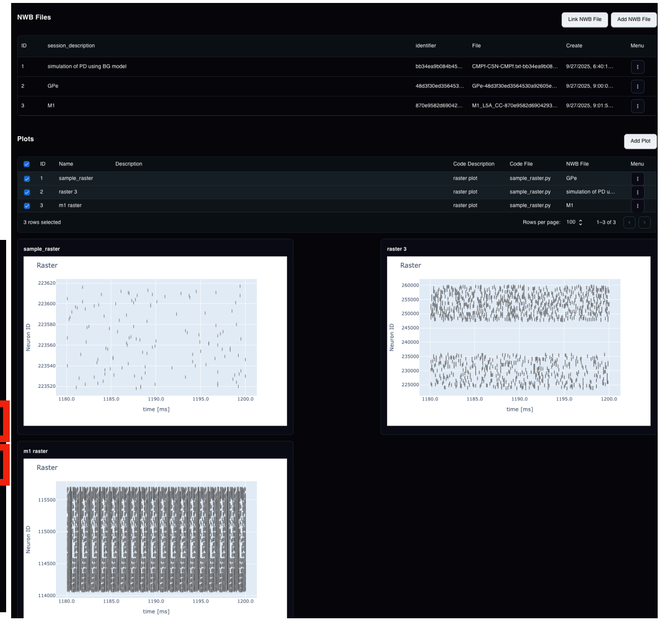
5 - In the simulation record, add a plot (input any description)



6- link that plot with a code, and then with a NWB



7 - you can have many NWB files and plots per simulation (efficient RE-USE of python code for plots)



Dashboard (makes uses of plotly, in theory we can make any interactive plot)

Challenge: Sample plot code (lets develop more plots and upload them !!)

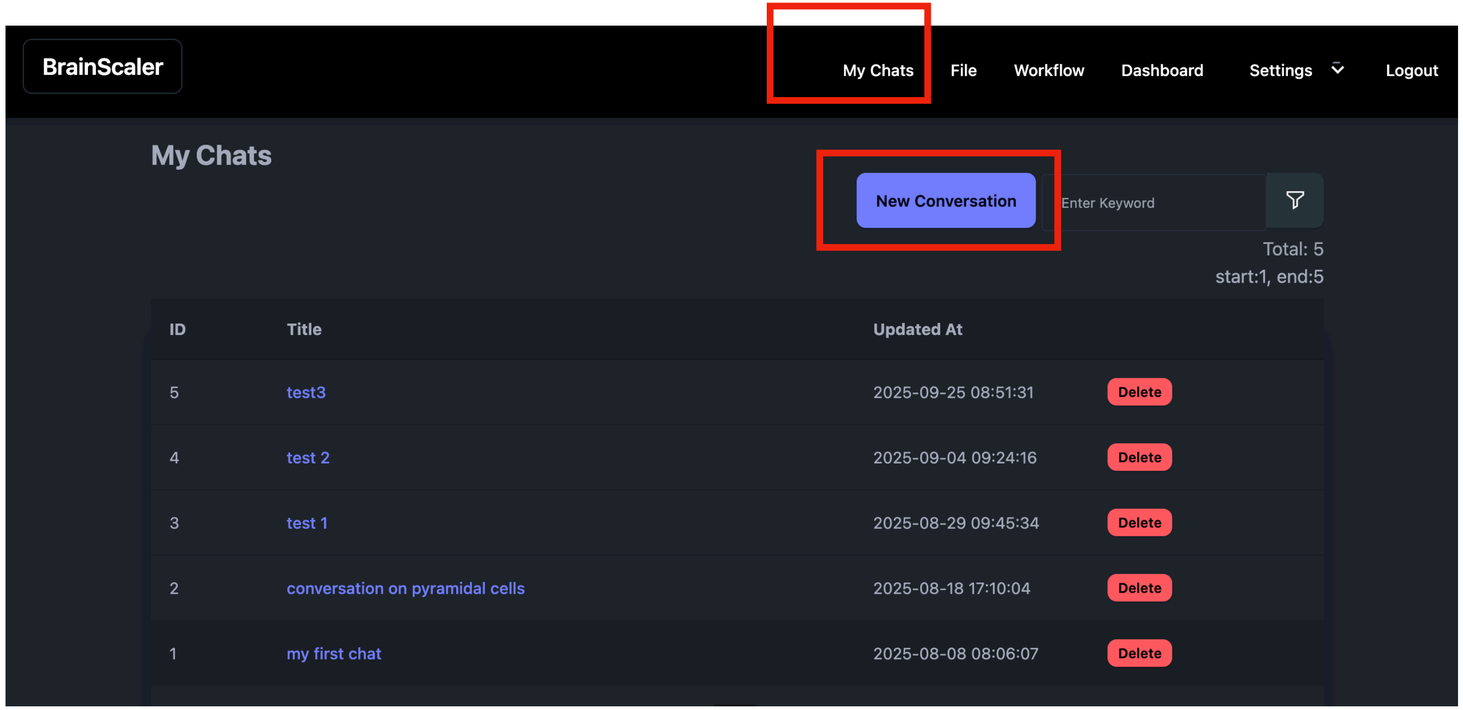
https://github.com/oist/brainscaler_dashboard/blob/main/sample/code/sample_raster.py

Prototypes - Proof-of-Concept

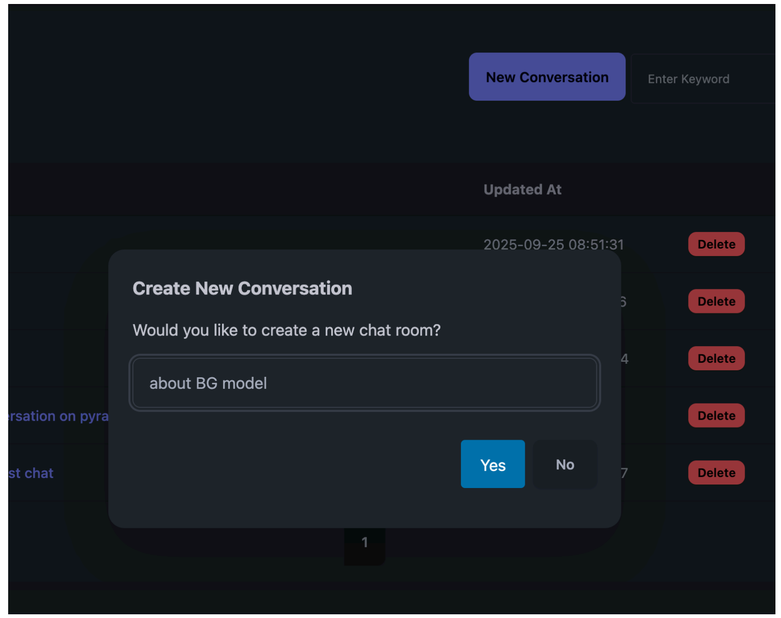
Chatbot

After installation, the Knowledge graph is small by default (only 1 paper, we will update it during hackathon)

1 - Go to My chats and create a new conversation



2 - Add conversation theme / name



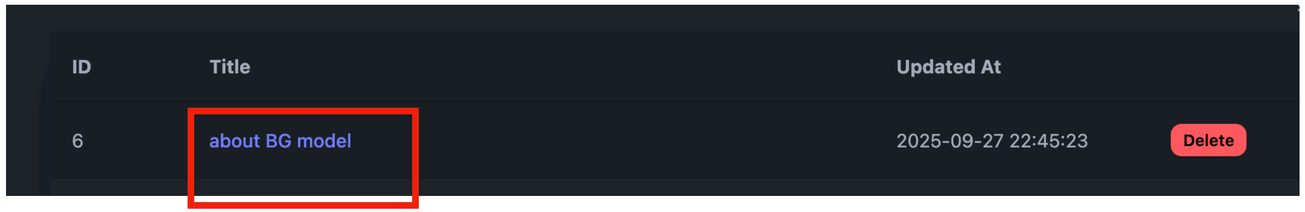
Prototypes - Proof-of-Concept

Chatbot

After installation, the Knowledge graph is small by default (only 1 paper, we will update it during hackathon)

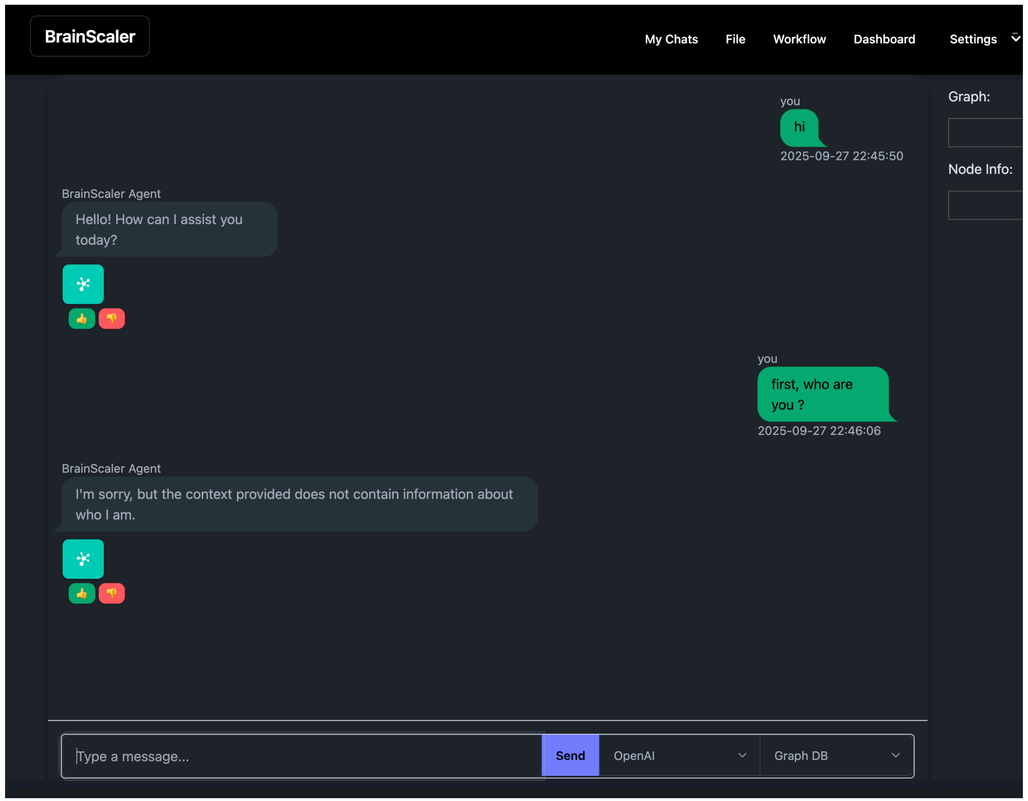
Retrieve context from graph (not fully implemented)

3 - Select the conversation

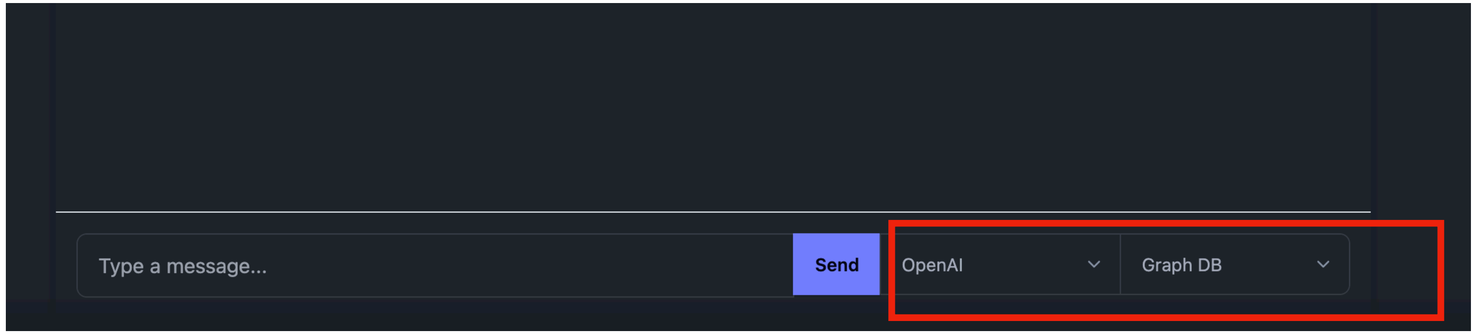


5 - Responses add 3 icons

Rate its response (for creating training data for fine-tuning)



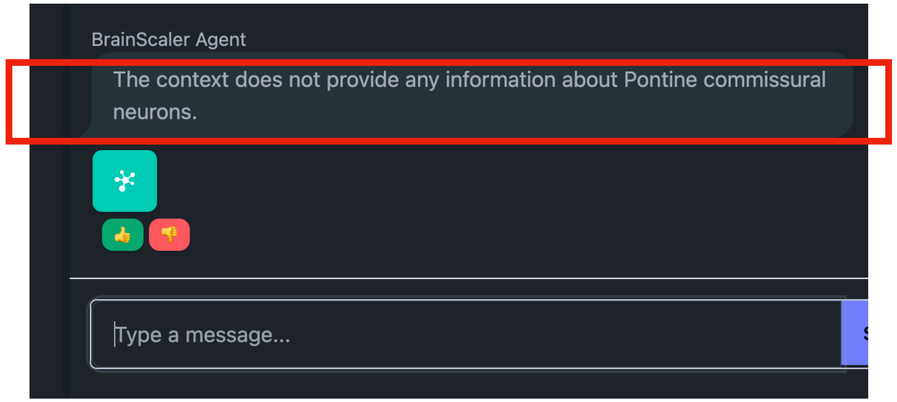
4 - Select OpenAI (there are bugs for other LLMs). Choose Graph DB for hybrid RAG (KG query + vector search) or choose Vector DB for conventional RAG (vector search)



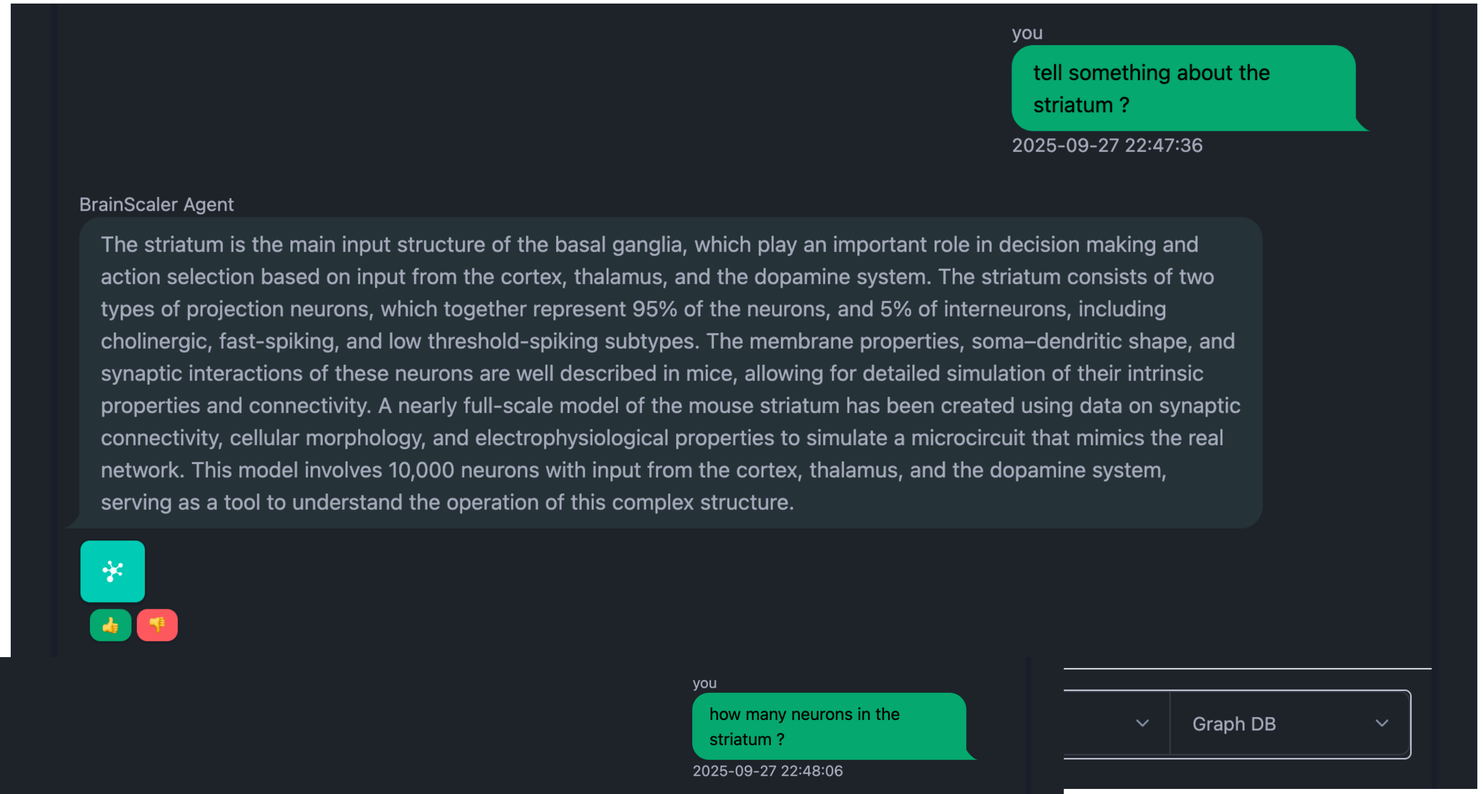
Prototypes - Proof-of-Concept

Chatbot

If context is not found in the graph db
(data related to your query is not found)



If context exists:

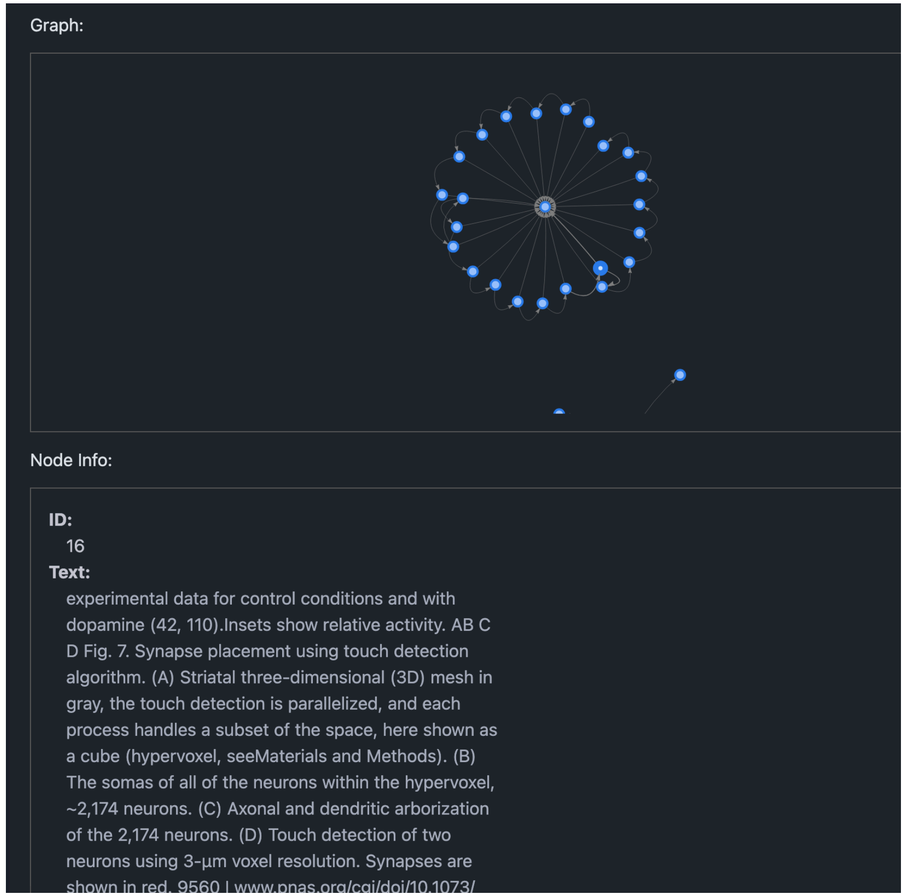


©2025 Okinawa Institute of Science and Technology OIST

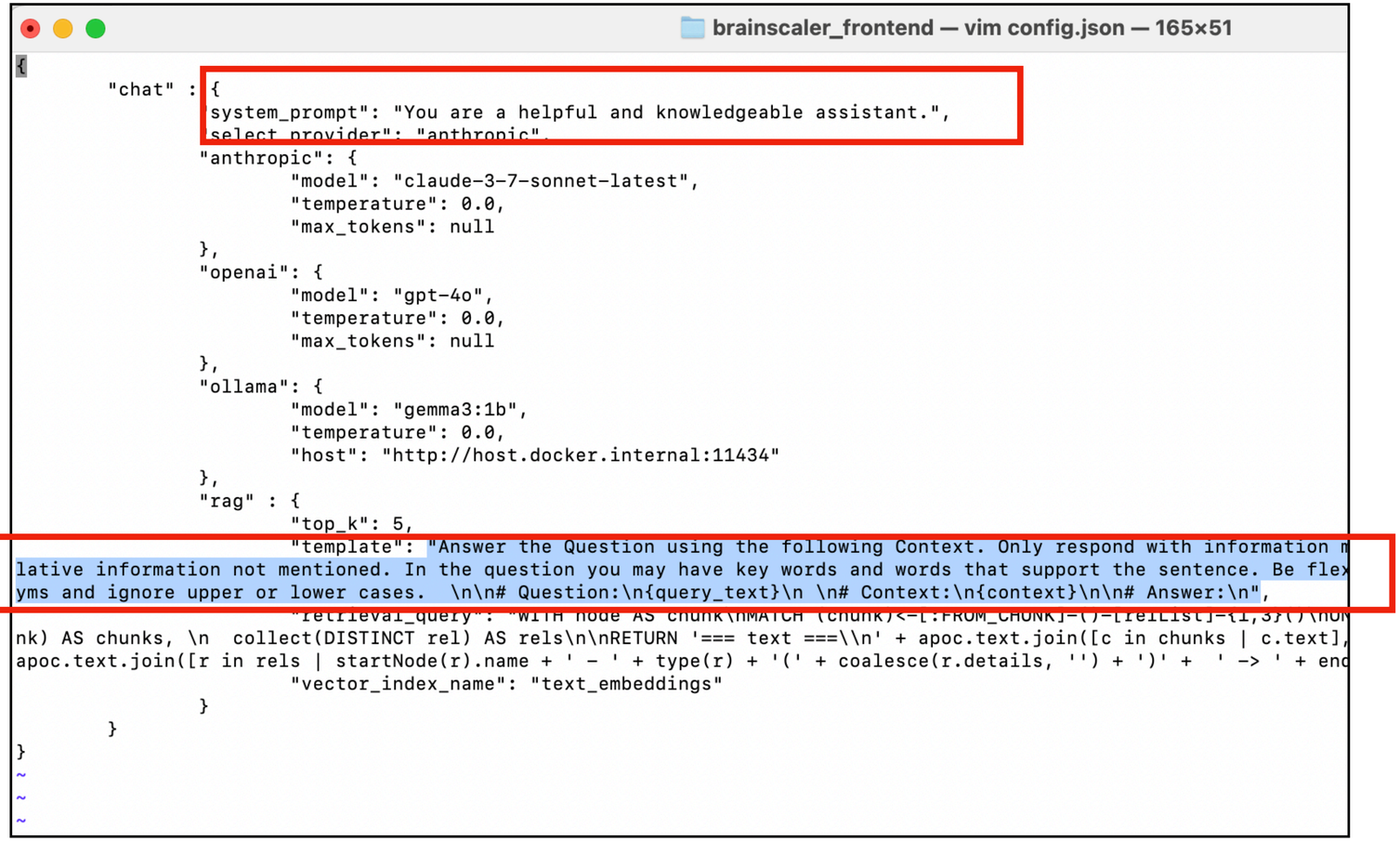
Prototypes - Proof-of-Concept

Chatbot

Retrieve context from graph
(not fully implemented)



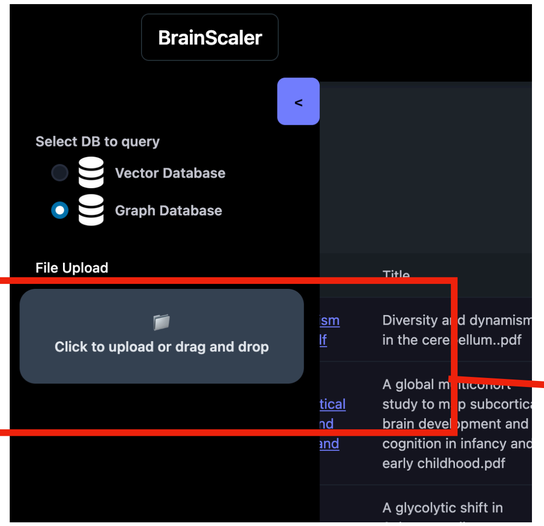
Change the character of your chatbot:
config.json contains the system and template prompts (changeable)

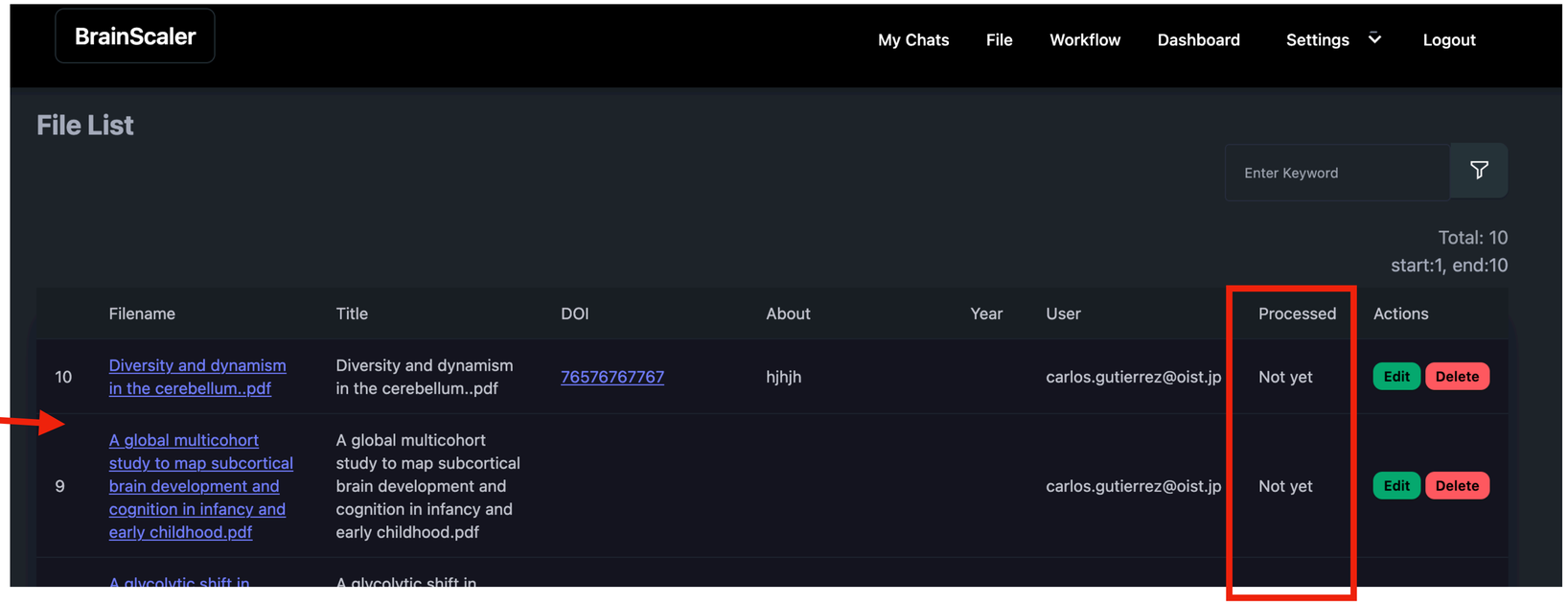


Prototypes - Proof-of-Concept

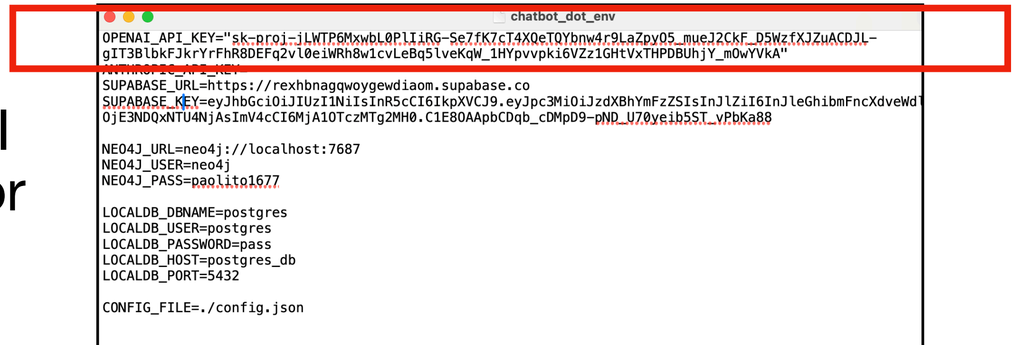
Chatbot

The DBs are built from papers
(It is possible to propose and upload papers, however the processing and detection of entities and connection is done by a centralized batch process)





We are making available a OPENAI temporal api key for testing (.env file)



Thanks, C.