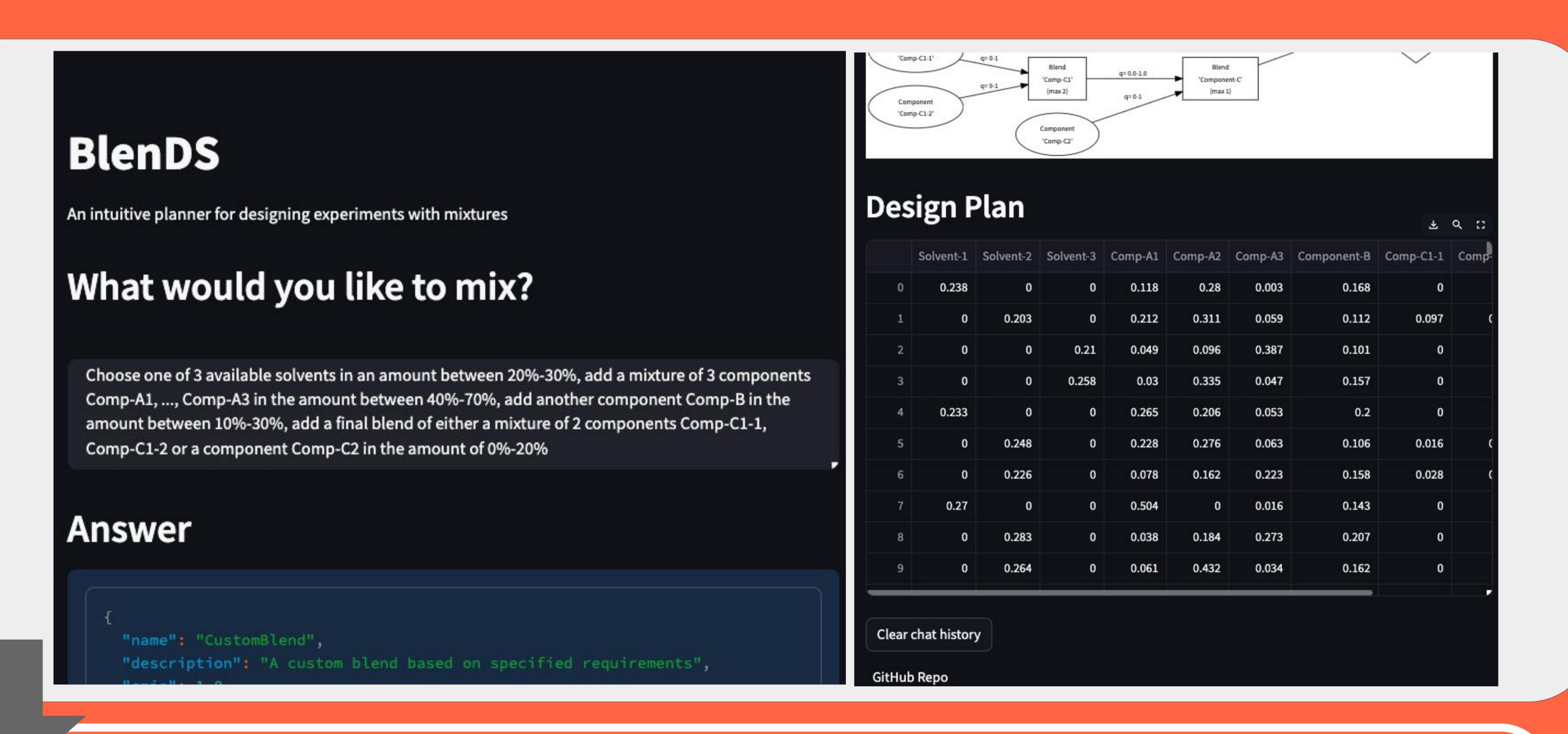


BlenDS - intuitive specification of the design space for blends of components



BlenDS
focuses here
for chemical
mixtures

GUI designed with
OpenAl
LangChain
Streamlit



Blend specified in dict/json format

{'name': 'myRootBlend', 'description': 'An example of a Blend. Properties: price (\$/kg). Choose one of 3 available solvents with an amount between 20%-30%, mix them with an additional 40%-70%blend of 3 components, finally add another component in the amount of 10%-60%. The price of the solvents is 4.3, 3.5 and 2.7 respectively. The price of each component of the blend is 50, 30 and 20, respectively. The last component costs 10 ', 'children': [{'name': 'Solvent', 'description': 'Choose only one of the 3 following solvents', 'children': [{'name': 'Solvent-1', 'description': 'xxx', 'qmin': 0, 'qmax': 1, 'props': {'price': 4.3}}, {'name': 'Solvent-2', 'description': 'yyy', 'qmin': 0, 'qmax': 1, 'props': {'price': 3.5}}, {'name': 'Solvent-3', 'description': 'zzz', 'qmin': 0, 'qmax': 1, 'props': {'price': 2.7}}], 'qmin': 0.2, 'qmax': 0.3, 'cmax': 1}, {'name': 'Blend-A', 'description': 'Mix the following A components', 'children': [{'name': 'Comp-A1', 'description': 'xxx', 'qmin': 0, 'qmax': 1, 'props': {'price': 50.0}}, {'name': 'Comp-A2', 'description': 'yyy', 'qmin': 0, 'qmax': 1, 'props': {'price': 30.0}}, {'name': 'Comp-A3', 'description': 'zzz', 'qmin': 0, 'qmax': 1, 'props': {'price': 20.0}}], 'qmin': 0.4, 'qmax': 0.7, 'cmax': 3}, {'name': 'Component-B', 'description': 'zzz', 'qmin': 0.1, 'qmax': 0.6,

'props': {'price': 10.0}}], 'qmin': 1.0, 'qmax': 1.0, 'cmax

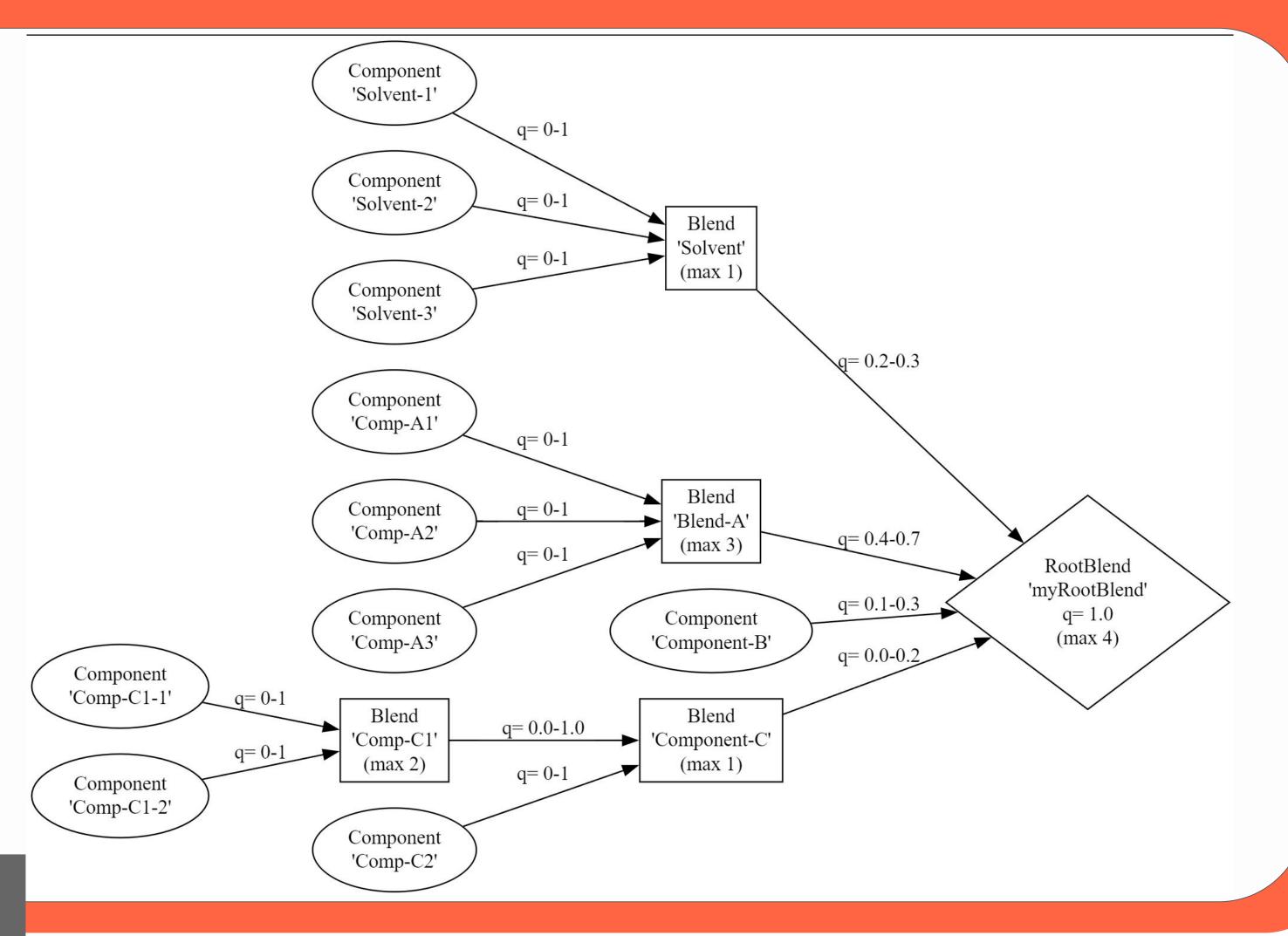
Recursively converted to a python object and validated

def dict_to_blend(children_dict):
 if 'children' in children_dict:
 children = []
 for child_dict in children_dict['children']:
 child = dict_to_blend(child_dict)
 children.append(child)

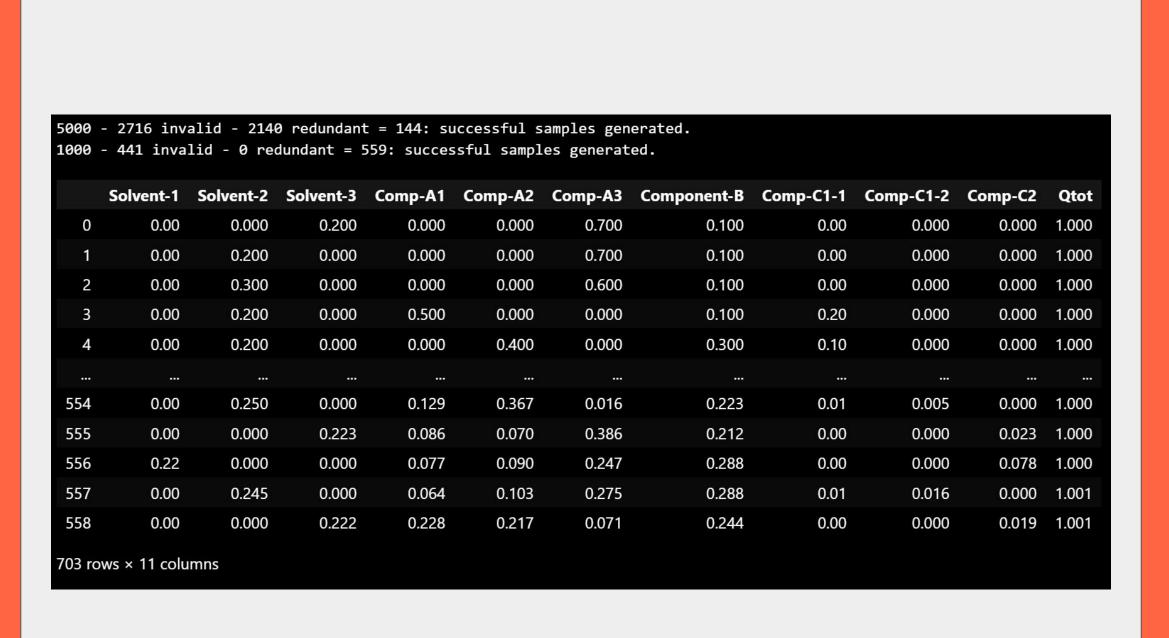
 children_dict_copy = children_dict.copy()
 children_dict_copy.pop('children')
 blend = Blend(**children_dict_copy, children=children)

 return blend

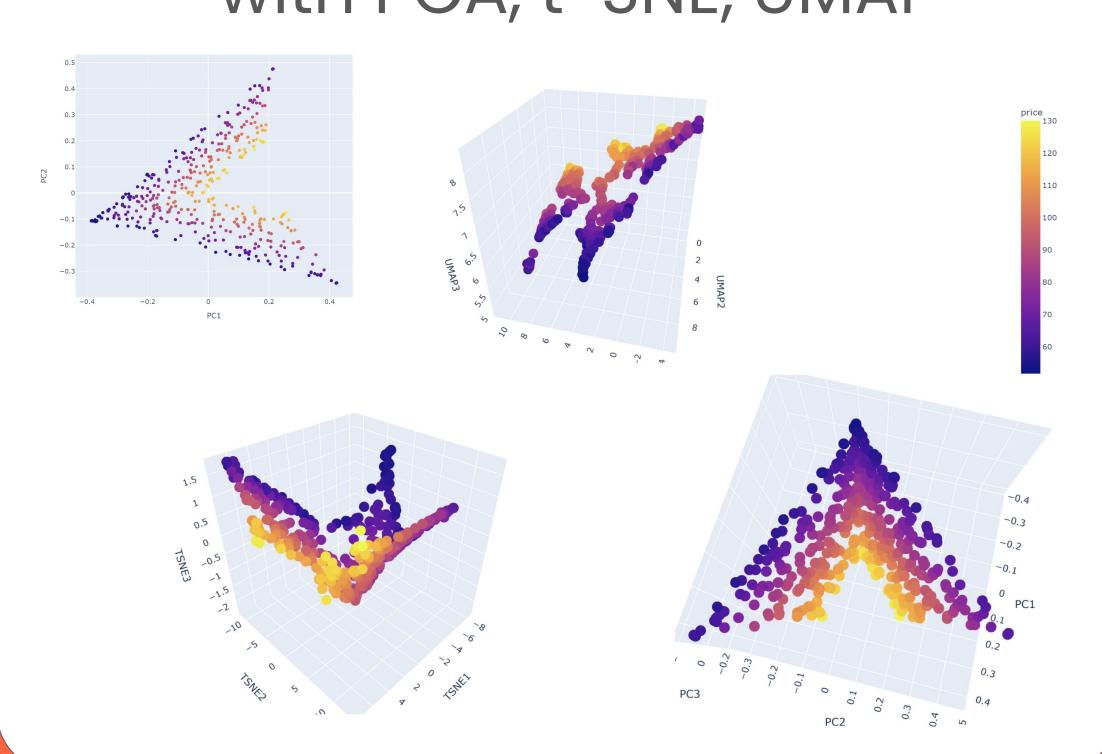
else:
 component = Component(**children_dict)
 return component



Generate samples for vertices and space-filling



Visualize the design space with PCA, t-SNE, UMAP



Sort samples by MinMax distance to pick the most diverse ones (and to have a fingerprint of their diversity)



Further work:

- accurate validation of the input (including the outcome of the LLM)
- convert to mass/molar units given the specifics from a database
 create a new valid sample when the
- user wants to lower/increase the quantity of a single component
 suggest a reasonable number of component increase the initial campbing
- suggest a reasonable number of samples for the initial sampling, given the complexity of the blend (using e.g., some elbow rule)

Repo: github.com/danieleongari/blends

- → daniele.ongari@syensqo.com (Italy)
- → alessio.tamburro@syensqo.com (ÚS)

