Unlocking Dynamic Frontend Development Through JSON Schema

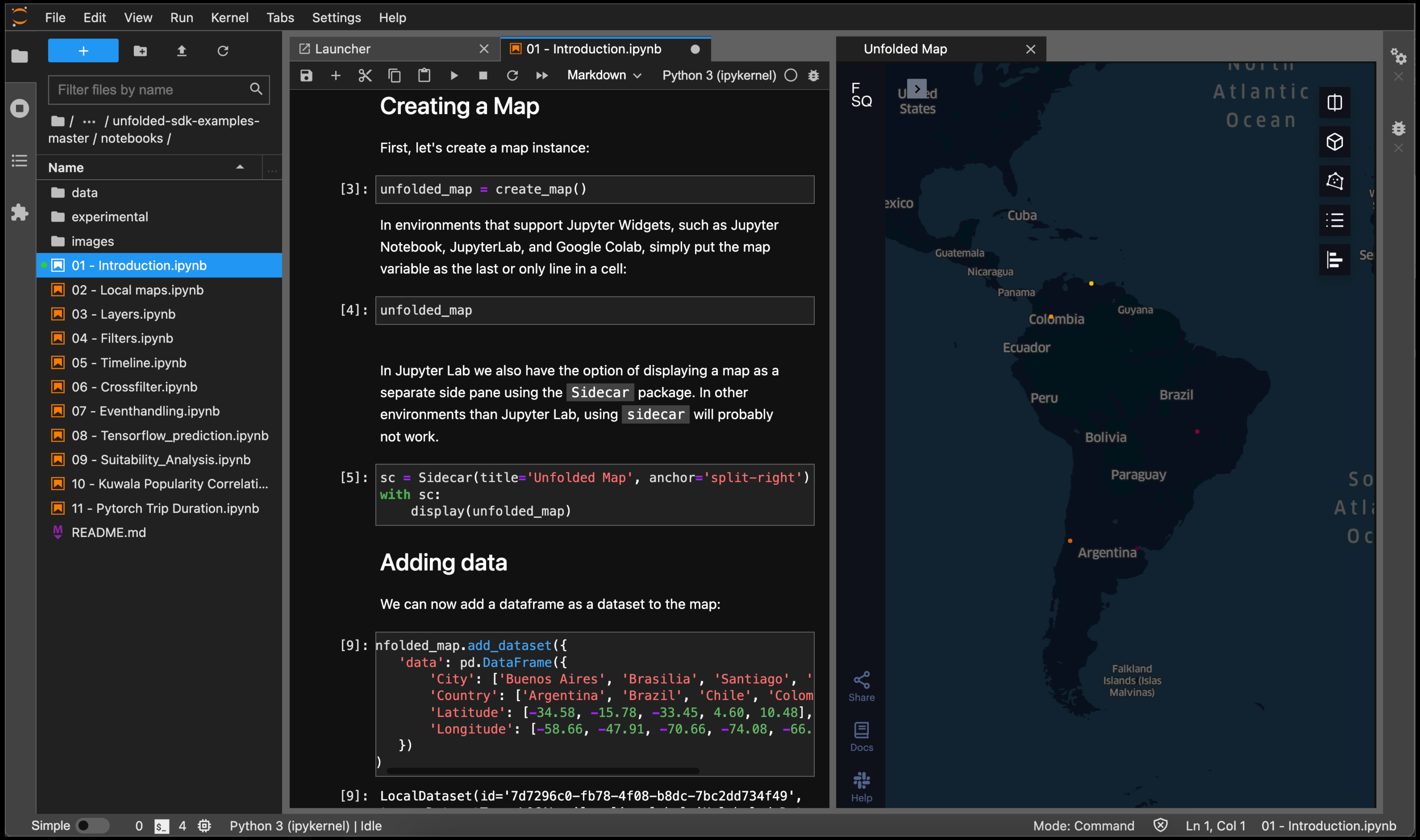
Neha Singla, Apple
Sathish Kumar Thangaraj, Apple
Zachary Sailer, Apple
Andrey Velichkevich, Apple

Agenda

- Jupyter Notebooks
- Jupyter Kernel Configurations
- Build Kernel Configurations Experience- Traditional Approach
- Json Schema
- Build Kernel Configurations Experience- Using Json Schema
- Technical Architecture
- Future Work

Jupyter Notebooks

- Jupyter Ecosystem
- Multiple languages
- Prototyping
- Data Exploration
- Iterative Experiments



Creating a Map

First, let's create a map instance:

```
[3]: unfolded_map = create_map()
```

In environments that support Jupyter Widgets, such as Jupyter Notebook, JupyterLab, and Google Colab, simply put the map variable as the last or only line in a cell:

```
[4]: unfolded_map
```

In Jupyter Lab we also have the option of displaying a map as a separate side pane using the `Sidecar` package. In other environments than Jupyter Lab, using `sidecar` will probably not work.

```
[5]: sc = Sidecar(title='Unfolded Map', anchor='split-right')
     with sc:
         display(unfolded_map)
```

Adding data

We can now add a dataframe as a dataset to the map:

```
[9]: nfolded_map.add_dataset({
        'data': pd.DataFrame({
            'City': ['Buenos Aires', 'Brasilia', 'Santiago', '
            'Country': ['Argentina', 'Brazil', 'Chile', 'Colom
            'Latitude': [-34.58, -15.78, -33.45, 4.60, 10.48],
            'Longitude': [-58.66, -47.91, -70.66, -74.08, -66.
        })
     )

[9]: LocalDataset(id='7d7296c0-fb78-4f08-b8dc-7bc2dd734f49',
```

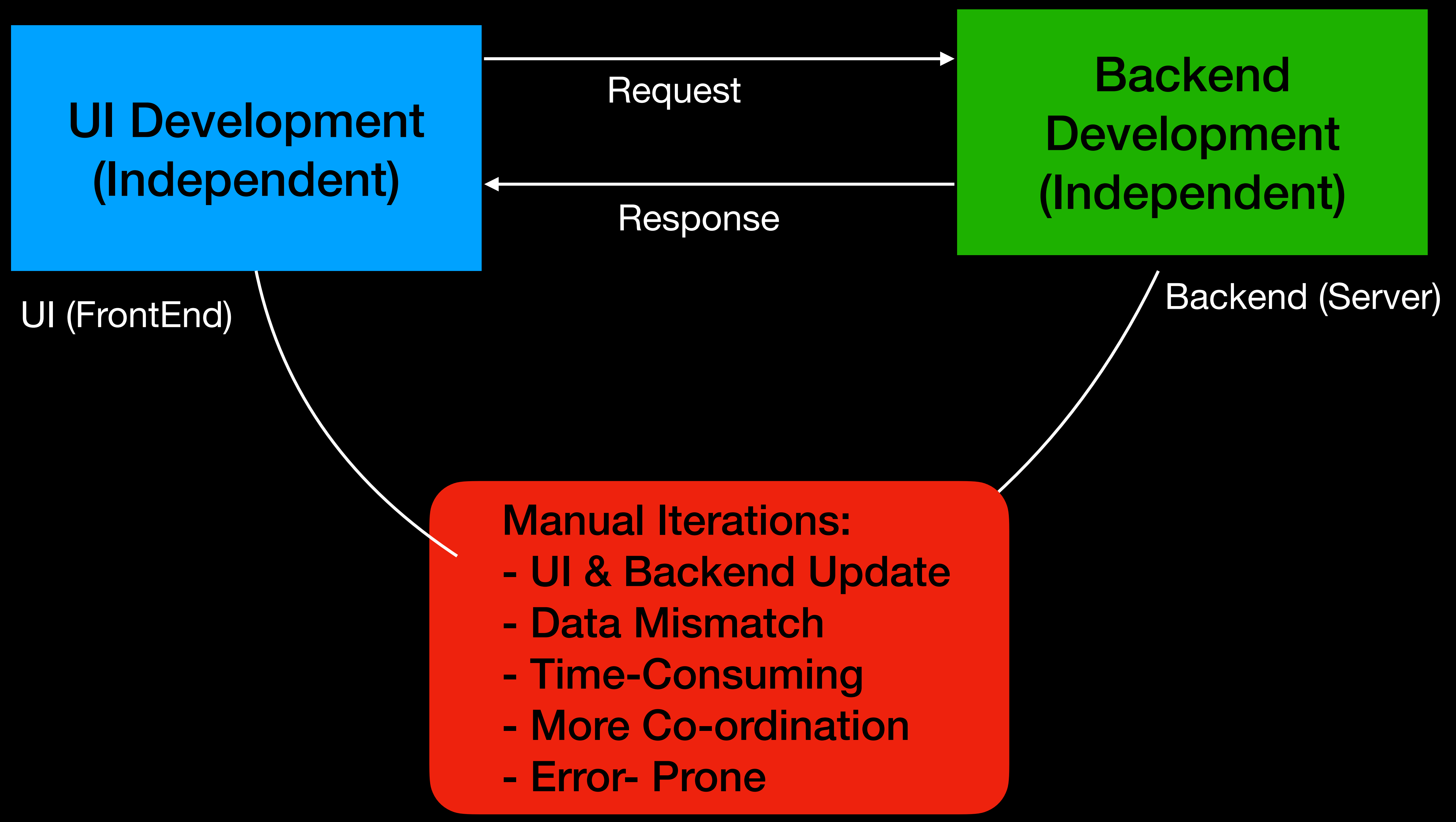
Jupyter Kernel Configurations

- Kernel Specification
- Remote Kernels
- Security Configurations
- Data Access Configurations
- Run Time Configurations

Jupyter Kernel Custom Configurations

- Custom programming language
- Custom environment
- Custom runtime
- Custom data access configurations

Build Kernel Configurations Experience - Traditional Approach

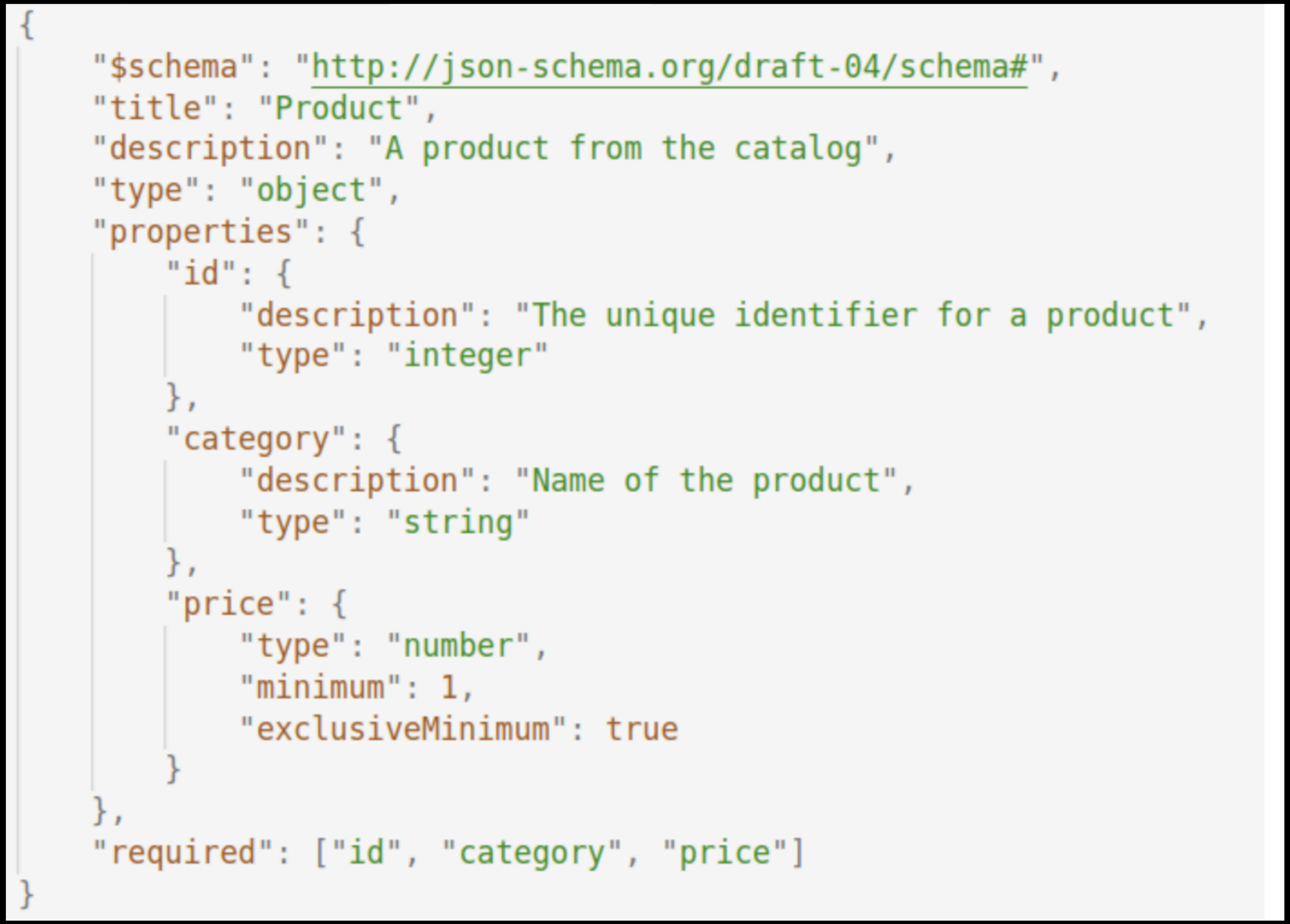


Limitations - Traditional Approach

- Tight Coupling of UI and Logic
- Scalability Challenges
- Reusability and Maintainability
- Lack of Data-Driven UI
- Poor Separation of Concerns
- Limited Flexibility for UI Customization
- Difficulty in Handling Complex UI States

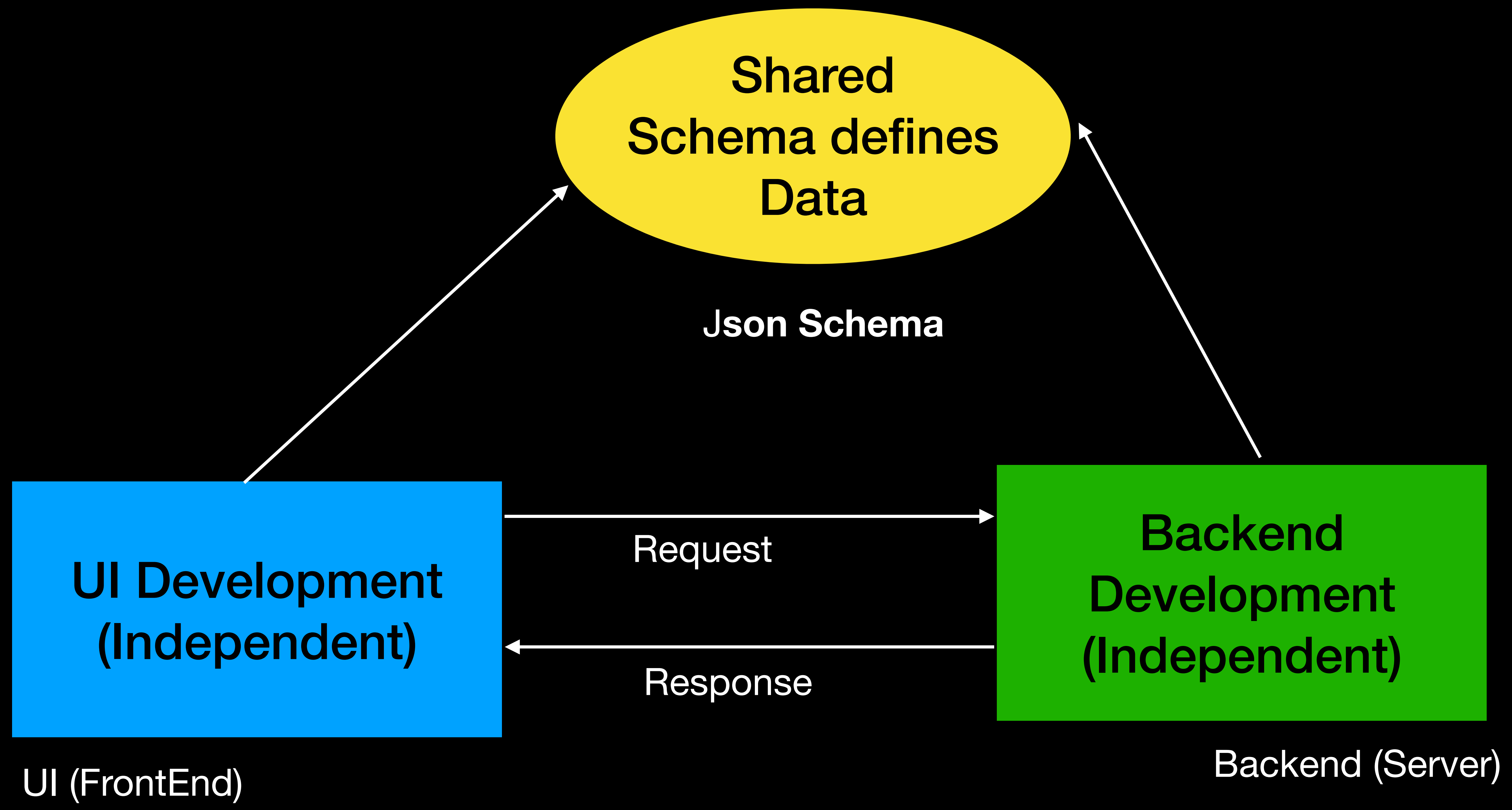
Json Schema

- Describe JSON data
- Data Validation
- Describing Data Structure
- API Contract Definition



```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "title": "Product",
    "description": "A product from the catalog",
    "type": "object",
    "properties": {
        "id": {
            "description": "The unique identifier for a product",
            "type": "integer"
        },
        "category": {
            "description": "Name of the product",
            "type": "string"
        },
        "price": {
            "type": "number",
            "minimum": 1,
            "exclusiveMinimum": true
        }
    },
    "required": ["id", "category", "price"]
}
```

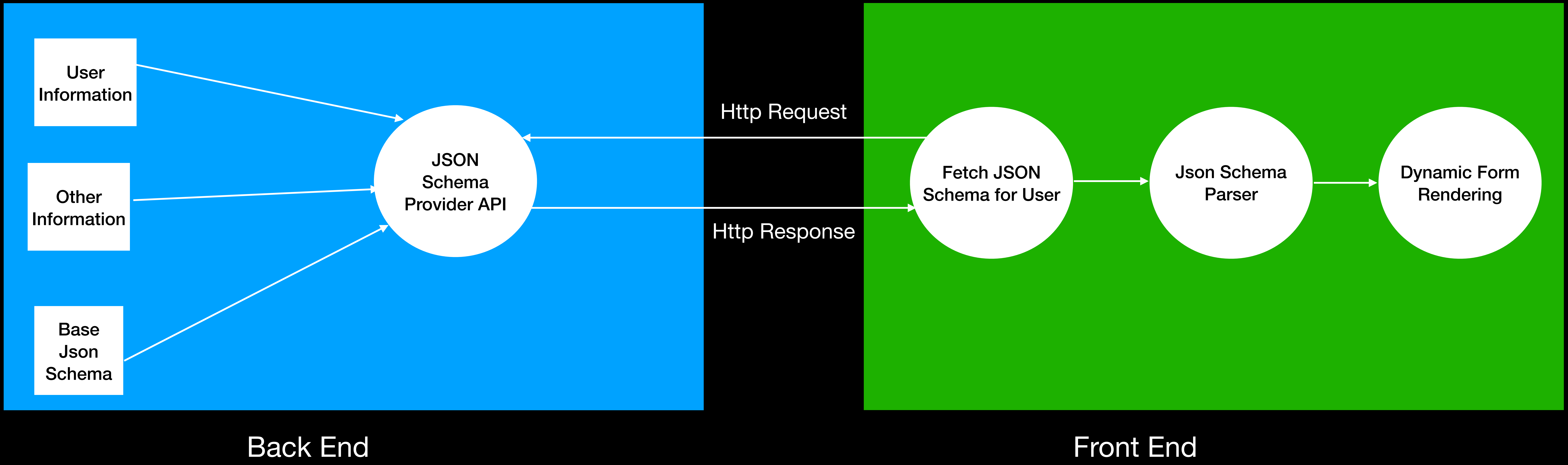
Build Kernel Configurations Experience- Using Json Schema


Shared Schema defines Data
Json Schema
UI Development (Independent)
Request
Response
Backend Development (Independent)
UI (FrontEnd)
Backend (Server)

Benefits:
* No Manual Updates
* Shared Schema
* Error-Free
* Consistent

Demo

Technical Architecture



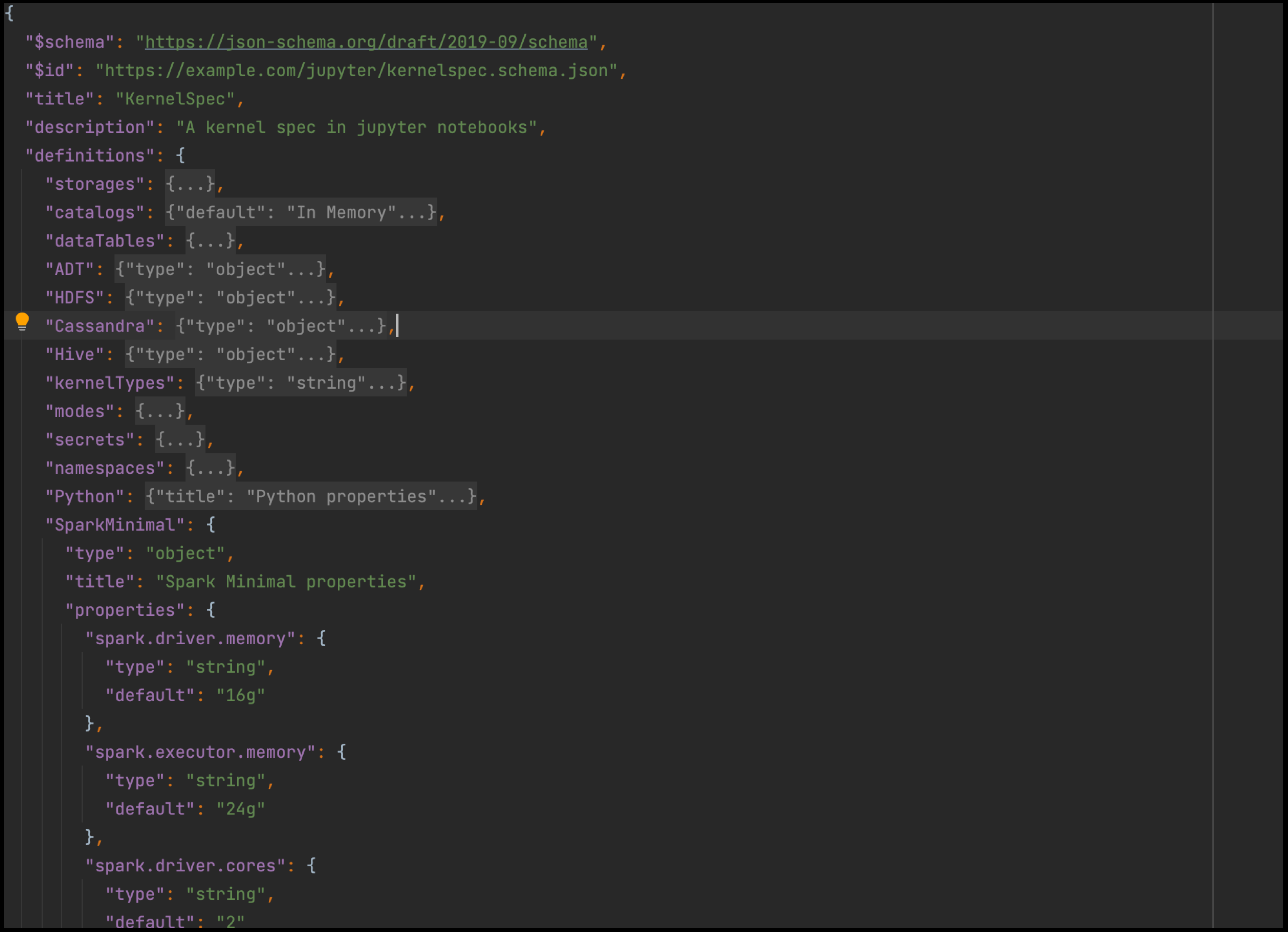
Components

- Backend
  - Json Schema Provider
- FrontEnd
  - Json Schema Parser
  - Dynamic Form Rendering

Json Schema Provider

- Dynamically Generate schema
- Data Driven
- Customizable rules
- Data Validation
- Real Time Updates

Base Schema



```
{
  "$schema": "https://json-schema.org/draft/2019-09/schema",
  "$id": "https://example.com/jupyter/kernelspec.schema.json",
  "title": "KernelSpec",
  "description": "A kernel spec in jupyter notebooks",
  "definitions": {
    "storages": {...},
    "catalogs": {"default": "In Memory"...},
    "dataTables": {...},
    "ADT": {"type": "object"...},
    "HDFS": {"type": "object"...},
    "Cassandra": {"type": "object"...},
    "Hive": {"type": "object"...},
    "kernelTypes": {"type": "string"...},
    "modes": {...},
    "secrets": {...},
    "namespaces": {...},
    "Python": {"title": "Python properties"...},
    "SparkMinimal": {
      "type": "object",
      "title": "Spark Minimal properties",
      "properties": {
        "spark.driver.memory": {
          "type": "string",
          "default": "16g"
        },
        "spark.executor.memory": {
          "type": "string",
          "default": "24g"
        },
        "spark.driver.cores": {
          "type": "string",
          "default": "2"
```

Future Work

- Open Source
  - Inviting Collaborators
  - Jupyter Meetings
    - https://jupyter-server.readthedocs.io/en/latest/contributors/team-meetings.html

Thank You

Questions?