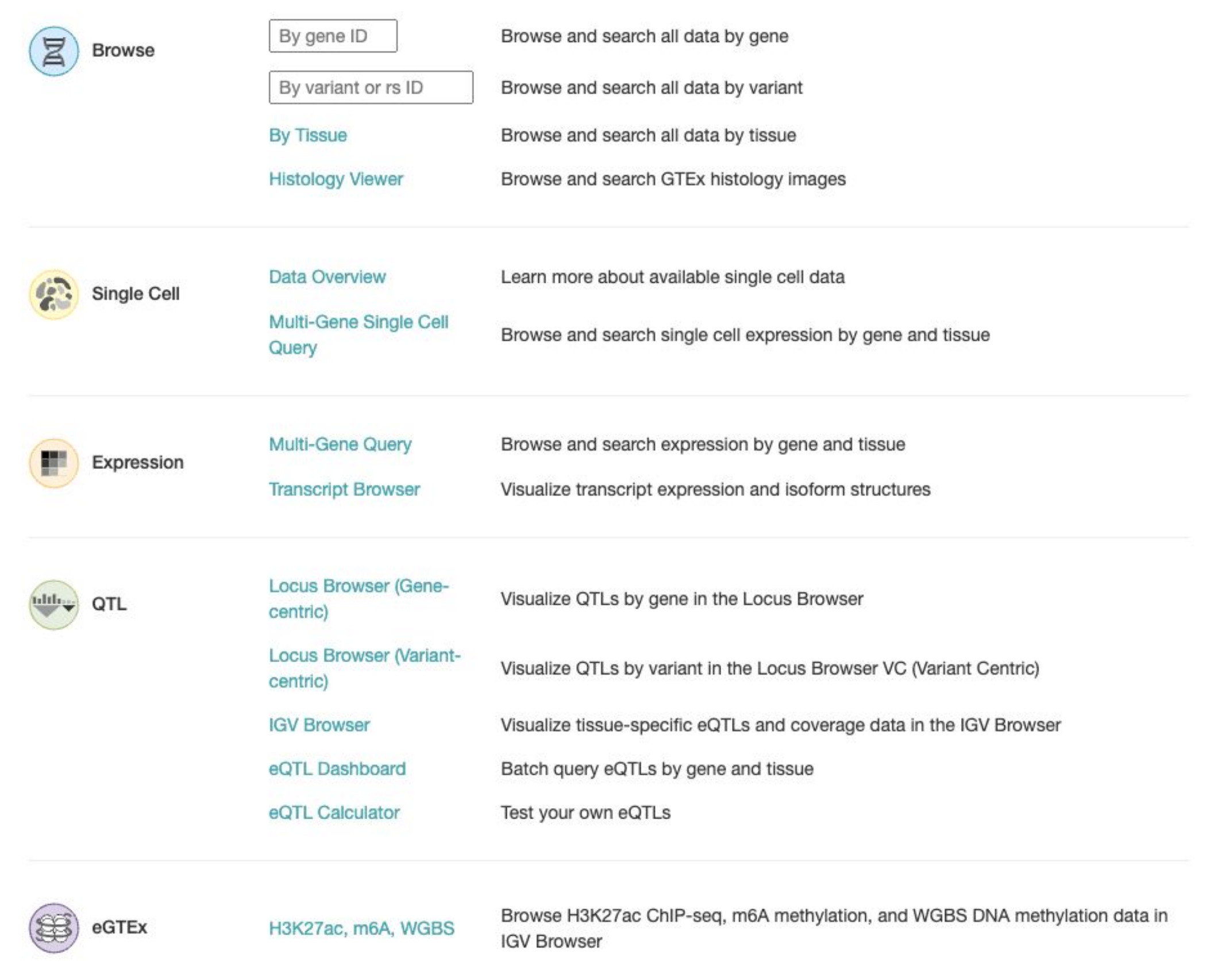
Building a Robust and Reliable API
With Pydantic, FastAPI, and Schemathesis

Joseph Okonda, Qiuyue Liu, Jared Nedzel, Kristin Ardlie

The Broad Institute, Cambridge, MA

Background

- As biological projects continue to produce large amounts of data, it is increasingly important to make this data easily accessible for researchers to utilize.
- One way to achieve this is through the use of well-designed, robust APIs. However, existing data-access APIs such as the RNAGet API or the DRS API, are designed with a single type of data in mind.
- Therefore, to **provide access to complex, multimodal data**, one often has to implement a bespoke API.
- Designing, developing, and maintaining such APIs, however, can be challenging, particularly in resource-constrained environments — with small teams and limited computational resources.



Browse	By gene ID	Browse and search all data by gene
	By variant or rs ID	Browse and search all data by variant
	By Tissue	Browse and search all data by tissue
	Histology Viewer	Browse and search GTEx histology images
Single Cell	Data Overview	Learn more about available single cell data
	Multi-Gene Single Cell Query	Browse and search single cell expression by gene and tissue
Expression	Multi-Gene Query	Browse and search expression by gene and tissue
	Transcript Browser	Visualize transcript expression and isoform structures
QTL	Locus Browser (Gene-centric)	Visualize QTLs by gene in the Locus Browser
	Locus Browser (Variant-centric)	Visualize QTLs by variant in the Locus Browser VC (Variant Centric)
	IGV Browser	Visualize tissue-specific eQTLs and coverage data in the IGV Browser
	eQTL Dashboard	Batch query eQTLs by gene and tissue
	eQTL Calculator	Test your own eQTLs
eGTEx	H3K27ac, m6A, WGBS	Browse H3K27ac ChIP-seq, m6A methylation, and WGBS DNA methylation data in IGV Browser

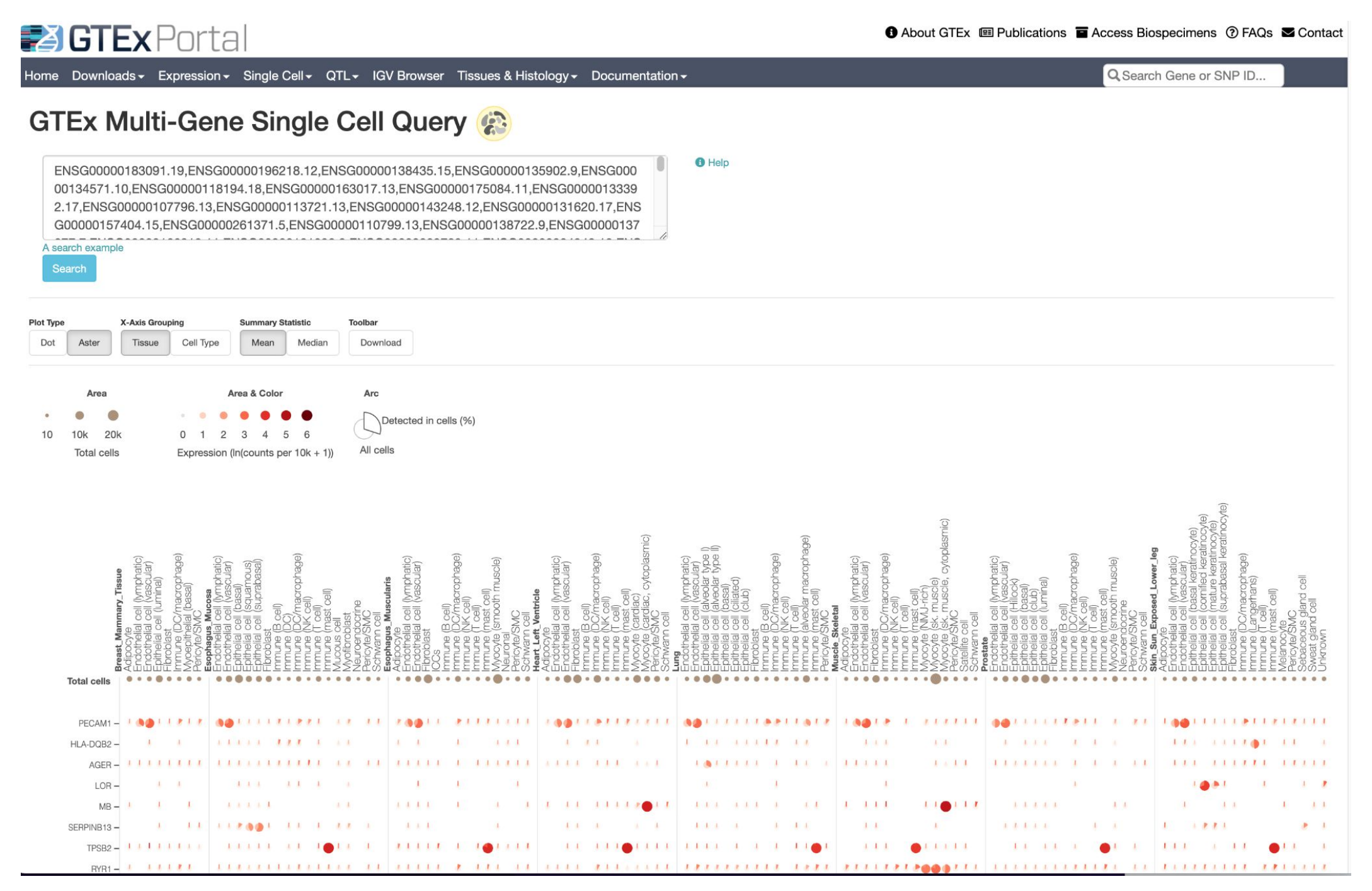
The data exposed by the GTEx API is complex and Multimodal. It ranges from files to dynamic eQTL calculations.

Goals

1. Make the API more **Reliable, and Robust** in order to minimize downtime.
2. Simplify the Codebase in order to make it **easier to maintain**.
3. Improve the API documentation to make it more **Interoperable and Reusable.**

Key Design Changes in the new API

1. **Paginate the Endpoints.** This would improve the API's robustness by preventing requests from taxing the database too much.
2. Perform Aggressive **Data Validation**. This would enable us to catch errors early.
3. Automatically generate the **documentation**, and ensure that it's alway up to date.
4. Ensure that we have extensive API tests — Use **Property Based Testing** for wider test coverage.



A screen shot of the GTEx Portal Showing the Single Cell Gene Search Feature.

FastAPI

FastAPI

- In GTEX V2, we switched **from Flask to FastAP**I for writing our web server.
- FastAPI provides a set of tools that allowed us to write a Clean and Simple web server.
- Further, it **automatically Generates OpenAPI** documentation.

Pydantic

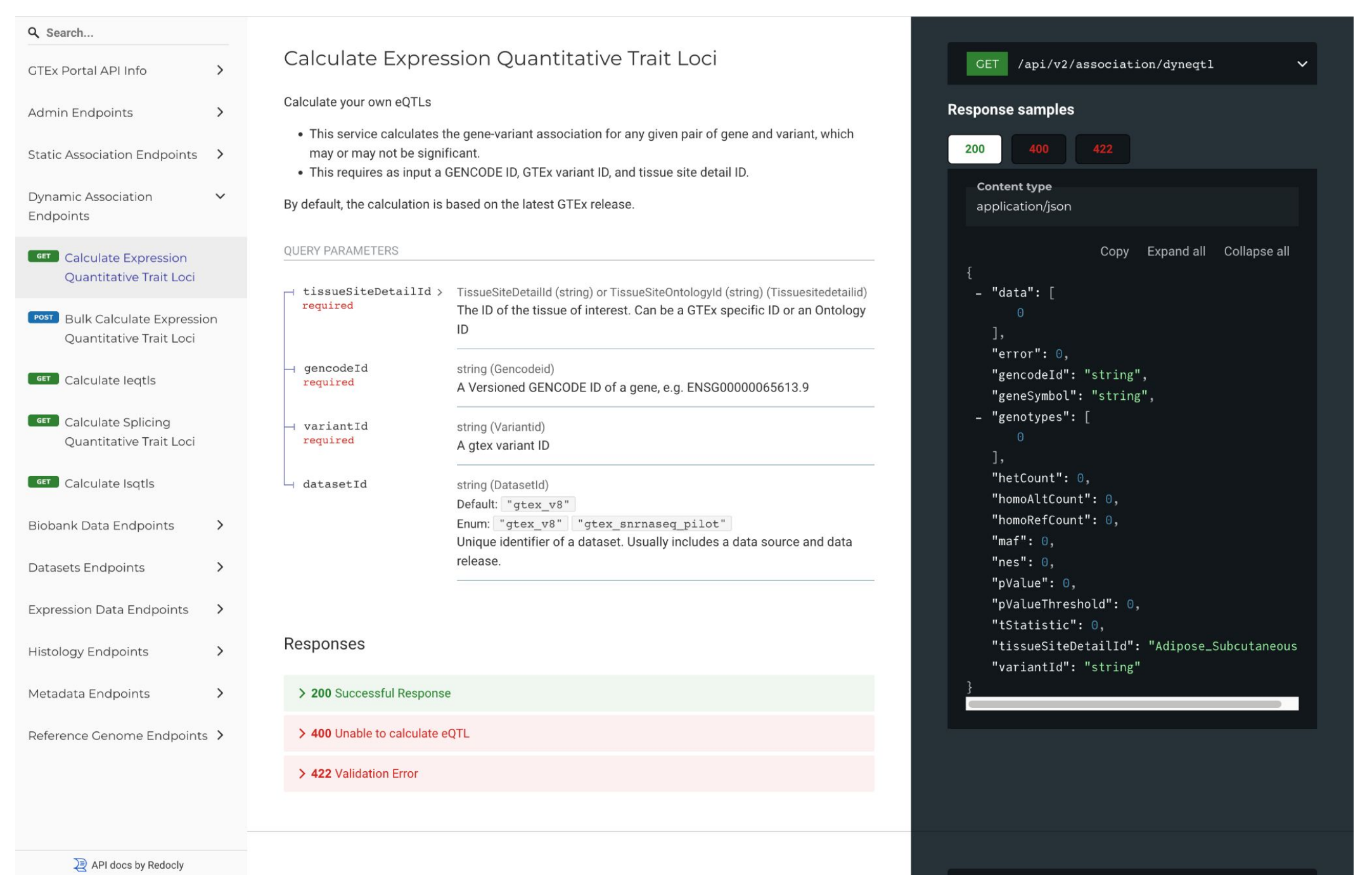
- To reach our goal of validating our data, we used Pydantic for **defining all our data models**.
- Pydantic allowed us to **add some form of type checking** (albeit at runtime) to User Input and the servers Responses.
- Further, it **automatically Generates model** documentation.

SchemaThesis

- We used Schemathesis, a Property Based Testing library, in order to provide wider test coverage without writing too many boilerplate unit tests.
- Coupled with the OpenAPI spec. generated by FastAPI, and the Validations of Pydantic, we were able to discover and fix a lot of errors at test time.

Results

1. The rewrite resulted in a highly improved documentation page which is helping to drive adoption.
2. Further, refactoring and maintaining the API has become much easier.


Search...
GTEx Portal API Info
Admin Endpoints
Static Association Endpoints
Dynamic Association Endpoints
GET Calculate Expression Quantitative Trait Loci
POST Bulk Calculate Expression Quantitative Trait Loci
GET Calculate Ieqtls
GET Calculate Splicing Quantitative Trait Loci
GET Calculate Isqtls
Biobank Data Endpoints
Datasets Endpoints
Expression Data Endpoints
Histology Endpoints
Metadata Endpoints
Reference Genome Endpoints
API docs by Redocly
Copy
Expand all
Collapse all

Calculate Expression Quantitative Trait Loci

Calculate your own eQTLs

- This service calculates the gene-variant association for any given pair of gene and variant, which may or may not be significant.
- This requires as input a GENCODE ID, GTEx variant ID, and tissue site detail ID.

By default, the calculation is based on the latest GTEx release.

QUERY PARAMETERS

`tissueSiteDetailId` required	TissueSiteDetailId (string) or TissueSiteOntologyId (string) (Tissuesitedetailid) The ID of the tissue of interest. Can be a GTEx specific ID or an Ontology ID
`gencodeId` required	string (Gencodeid) A Versioned GENCODE ID of a gene, e.g. ENSG00000065613.9
`variantId` required	string (Variantid) A gtex variant ID
`datasetId`	string (DatasetId) Default: "gtex_v8" Enum: "gtex_v8" "gtex_snrnaseq_pilot" Unique identifier of a dataset. Usually includes a data source and data release.

Responses

- 200 Successful Response
- 400 Unable to calculate eQTL
- 422 Validation Error

GET /api/v2/association/dyneqtl

Response samples

200 400 422

Content type application/json

```
{
  - "data": [
        0
    ],
    "error": 0,
    "gencodeId": "string",
    "geneSymbol": "string",
  - "genotypes": [
        0
    ],
    "hetCount": 0,
    "homoAltCount": 0,
    "homoRefCount": 0,
    "maf": 0,
    "nes": 0,
    "pValue": 0,
    "pValueThreshold": 0,
    "tStatistic": 0,
    "tissueSiteDetailId": "Adipose_Subcutaneous
    "variantId": "string"
}
```

A screenshot showing a portion of the new, improved documentation page.