Developing a User Interface for DUNE HEPCloud

Elisabeth Petit – Bois
Georgia Institute of Technology – GEM Fellow

Supervisors:
Kenneth Herner
Michael Kirby
Andrew Norman
Fermi National Accelerator Laboratory

Table of Contents

1. Acknowledgements 3
2. Background 4
   2.1. Computing at Fermilab 4
   2.2. Job Submission Services 4
   2.3. HEPCloud 4
      2.3.1. HTCondor 4
      2.3.2. HEPCloud Decision Engine 4
   2.4. Challenges 5
3. Objective 6
4. Project Specification 6
   4.1. Stakeholders 6
   4.2. Application Requirements 6
5. Development 8
   5.1. Identity 8
   5.2. Frontend 8
   5.3. Backend 11
      5.3.1. Server Configuation 11
      5.3.2. API Configuration 11
      5.3.3. Application Data 13
6. Technologies 16
7. Development Challenges 17
8. Future Direction 17
9. References 19

1. Acknowledgements

First and foremost, I would like to thank all three of my supervisors – Kenneth Herner, Michael Kirby, and Andrew Norman – for being great advisors and mentors. They were graciously tolerable of me during my stay by answering questions, scheduling meetings, debugging code, and whatever else despite their busy schedules.

A huge thank you to the Scientific Computing Department (SCD) and, most particularly, the HEPCloud team for all the guidance throughout this summer. Additionally, none of this work would be possible without the work of the SIST committee successfully conducting the first remote intern program this year.

Finally, I would like to extend my gratitude to The National GEM Consortium for their role in making opportunities like these possible for myself and many other minorities in STEM looking to conduct research and pursue graduate education.

2. Background

The Deep Underground Neutrino Experiment (DUNE) is an international experiment headquartered at Fermi National Accelerator Laboratory (Fermilab) in Batavia, Illinois. The collaboration grew in an attempt to support ongoing efforts for studying neutrino physics.

DUNE, although still under construction, is an extremely ambitious neutrino physics plan. The completed project will be housed in the United States; however, there is a prototype at CERN known as ProtoDUNE. The final result is expected to consist of two particle detectors and a proton accelerator, powering the most intense neutrino beam in the world. The neutrino beam will travel through the Earth from Fermilab to the Sanford Underground Research Facility in South Dakota, spanning an impressive distance of around 1,300 kilometers total [1].

2.1 Computing at Fermilab

Because the nature of DUNE's work is complex, the algorithms built to analyze interesting detector events are naturally also incredibly intricate. In fact, scientific computation requirements from experiments like DUNE regularly exceed require hundreds of millions of CPU hours to complete. Therefore, there is an ongoing need for high performance computing (HPC) and dynamic resource provisioning. These technologies maximize efficiency, increasing throughput and resource utilization. They also allow for researchers to worry less about coding for optimal algorithmic complexity and focus more on the objective at hand.

2.2 Job Submission Services

In order to get computing jobs to a HPC cluster, scientists must first submit them through either the Production Operations Management System (POMS) or JobSub [2, 3]. POMS is a graphical user interface that operates as a JobSub wrapper. For those who prefer command line interfaces, JobSub is readily available to perform the same operations as POMS. In these systems, users identify job parameters, or "classads," that inform backend systems on how to handle an incoming computation request.

2.3 HEPCloud

HEPCloud is Fermilab's main gateway to a wide range of computing resources. For researchers who require HPC, this service is tied to the Worldwide LHC Computing Grid (WLCG), allowing for access to varied resources like institutional clusters, supercomputers, and commercial clouds located all around the globe [4, 5, 6].

2.3.1 HTCondor

HTCondor is a batch processing system used to onboard computation jobs to HPC clusters. The system makes use of classads to describe resources necessary to accomplish the computing task of a job. As mentioned in Section 2.2, this metadata is defined via POMS or JobSub and later used by the HEPCloud Decision Engine to pair jobs to an available computing cluster.

2.3.2 HEPCloud Decision Engine

The HEPCloud Decision Engine is a service that performs cluster backend matching for DUNE jobs submitted through POMS or JobSub [7]. The Decision Engine analyzes parameters assigned to each job to identify the most appropriate HPC cluster capable of performing the computation task.

To achieve its goal, the backend of the Engine hosts a set of Logic Rules that define how jobs should be interpreted at the time of submission. It also manages several channels that contain information regarding the status of HPC clusters. This channel data, in combination with job parameter data, informs the Logic Rules about where a particular job should run [8, 9].

2.4 Challenges

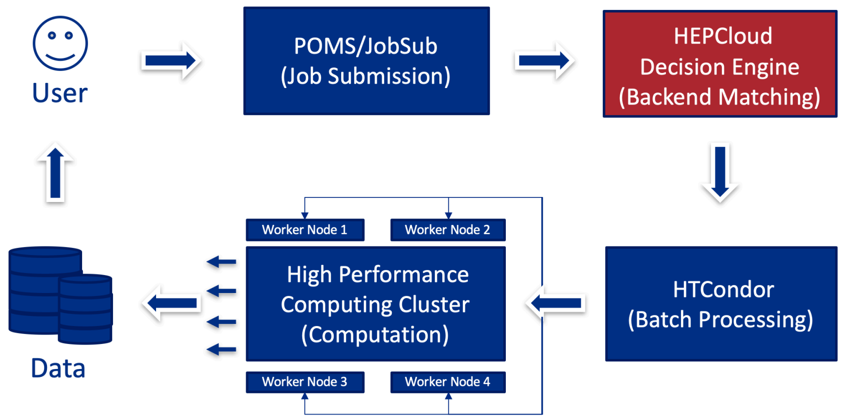


Figure 1: DUNE Workflow Overview

Although the job submission workflow is a solid process, there is more to be desired from the system. Firstly, the HEPCloud Decision Engine presently operates as a black-box for users. After a job is submitted through POMS or JobSub, users cannot modify parameter configurations or cluster preferences. In fact, most scientists are not aware where their jobs run, if they run at all. Furthermore, the state of the Decision Engine is not accessible for users to see or alter.

These challenges pose potential risks for experiment efficiency. For example, in the case of financial resources, ideally, the Decision Engine should direct jobs to the cheapest commercial cluster at the time of submission. However, because the decision-making process is obfuscated, it is unclear that the correct choice is being made every time. The same case can be made for computational resources. The Decision Engine may miss opportunities where it could run a job at a less active cluster versus one that is inundated with computation jobs in its queue.

The overall lack of end-user confidence and awareness of what is happening in the system backend has the potential to be detrimental to the bottom line and the quest for new physics.

3. Objective

The objective of this project is to build a web application that allows DUNE users to:

- Read and Write the global state of the HEPCloud Decision Engine.
- Read and write job-specific parameters.
- Read Decision Engine data regarding cluster backend matching.

In order to:

- Create system transparency by opening the Decision Engine's black-box.
- Provide users with more control over their job submissions.

4. Project Specification

Understanding the key participants as well as their needs is crucial to address the challenges faced by the DUNE team and Fermilab at large. This section describes the main stakeholders and defines key project requirements gathered prior to project development.

4.1 Stakeholders

Seeing as this project directly addresses concerns raised by participants within DUNE, the major collaborators originate from this experiment. The DUNE team represents a small group of users who will be able to test and use the application prototype and final product. They understand the overall vision for the project and regularly perform the DUNE workflow in Figure 1. Within this group, there are three main contacts who are also project supervisors:

- **Kenneth Herner** — DUNE Scientist who also provides experiment software support.
- **Michael Kirby** — DUNE Scientist who also coordinates computing services at Fermilab.
- **Andrew Norman** —DUNE Scientist who also serves as the HEPCloud project lead.

Next, the HEPCloud team is responsible for high performance scientific computing capabilities across Fermilab. A subset of the HEPCloud unit hosts the Decision Engine team which manages and develops the software responsible for how backend matching occurs after job submission. Andrew Norman serves as the main point-of-contact for HEPCloud.

4.2 Application Requirements

The application requirements call for three main pages:

1. a **global page** depicting the state of the HEPCloud Decision Engine.
2. a **job parameters page** showcasing information concerning a particular job.
3. an **all jobs page** listing jobs that a user has previously submitted to the job submission services described in Section 2.2.

Table 1 below further elaborates on the main requirements gathered during the elicitation meeting.

Table 1 – Application Requirements	
#	Requirement
1.	The user should be able to view the parameters currently placed in the Decision Engine.
2.	The user should be able to view the parameters relating to a particular Job ID.
3.	The user should be able to modify global parameters within the Decision Engine.
4.	The user should be able to modify job-specific parameters.
5.	The user should be able to add job-specific parameters.
6.	The user should be able to remove job-specific parameters.
7.	The user should be able to view decisions regarding HPC cluster-matching for a particular job.
8.	The user should be able to view all submitted jobs where they are the job owner.
9.	The user should be able to filter jobs by their parameters.
10.	The user should be able to access the job parameters page by clicking on a job listed on the all jobs page.
11.	The user should be able to access the application via Fermilab's Single Sign-On (SSO) Identity Provider.

5. Development

During the first three weeks of the internship, the main focus was establishing a baseline for user needs and creating a visual identity for the application.

For the remainder of the ten weeks, tasks were mostly code-driven, and my supervisors and I largely followed a sprint-like approach to arrive at the final result. Sprint meetings, although unstructured, were a critical time during the week to relay progress, ask questions, provide feedback, and define the next week's objectives.

This section details components of the development process including establishing the application design and identity, creating the frontend, and configuring the backend.

5.1 Identity

The very first task was to assign a name and create a graphic representation for the application. The submitted brand is shown in Figure 2.

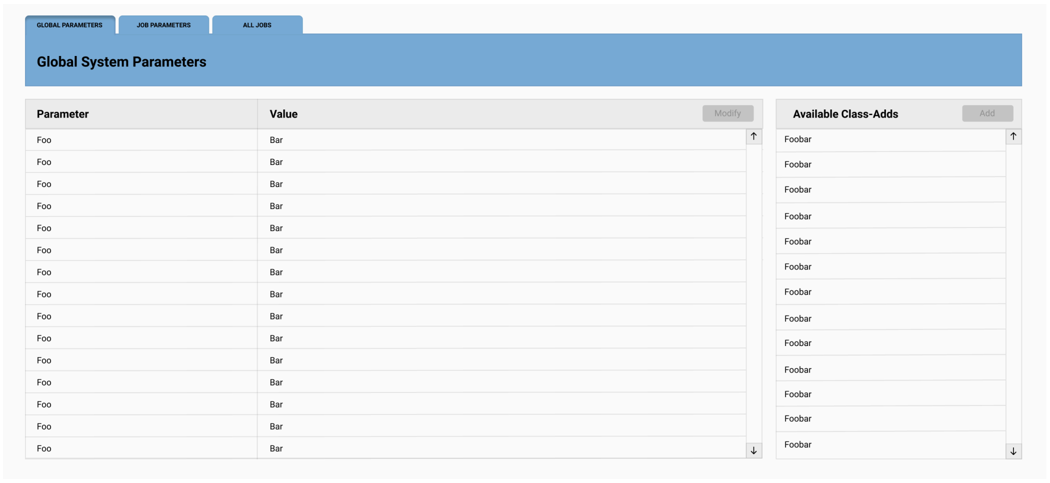
Because the application is mainly concerned with creating transparency regarding which computing clusters jobs are directed to after submission, the graphic tries to depict a "decision tree" itself. The name and brand is suitable because:

Figure 2: Decision Tree Graphic

- Much like how the Decision Engine operates, decision trees use available factors to arrive at a verdict.
- Decision trees elucidate the decision-making process, leaving no room for questionable conclusions.
- "Decision Tree" follows the "Decision" scheme seen in the already-established "Decision Engine" service.

5.2 Frontend

Next, designing the application interface also was a major task. We decided on a look-and-feel that best suited the requirements. Of course, as time elapsed and requirements shifted, our application layouts followed suit. For example, the global state page particularly experienced a huge evolution from design ideation to final arrangement (Figures 3, 4, and 5):

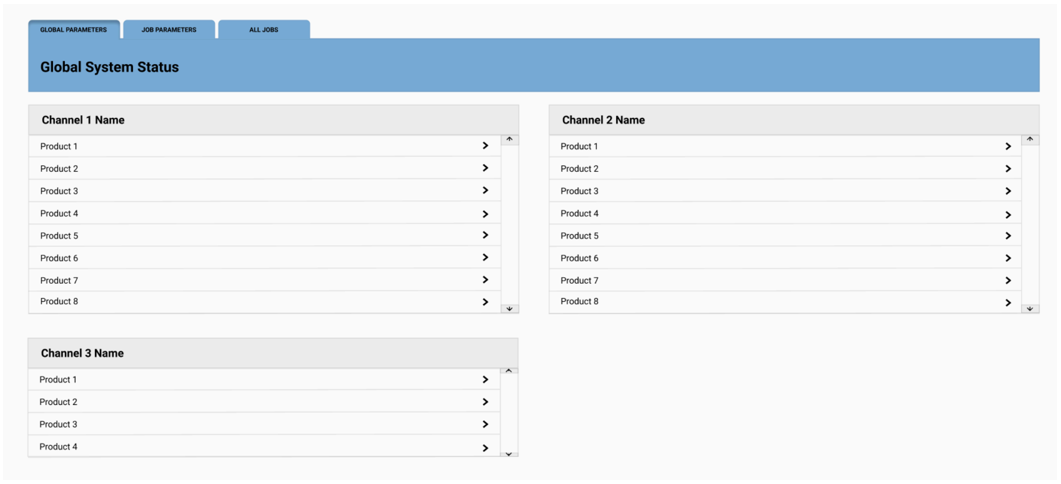


Global System Parameters

Parameter	Value
Foo	Bar
Foo	Bar
Foo	Bar
Foo	Bar
Foo	Bar
Foo	Bar
Foo	Bar
Foo	Bar
Foo	Bar
Foo	Bar
Foo	Bar
Foo	Bar
Foo	Bar
Foo	Bar
Foo	Bar

Available Class-Adds
Foobar
Foobar
Foobar
Foobar
Foobar
Foobar
Foobar
Foobar
Foobar
Foobar
Foobar
Foobar
Foobar

Figure 3: Global Page Prototype 1



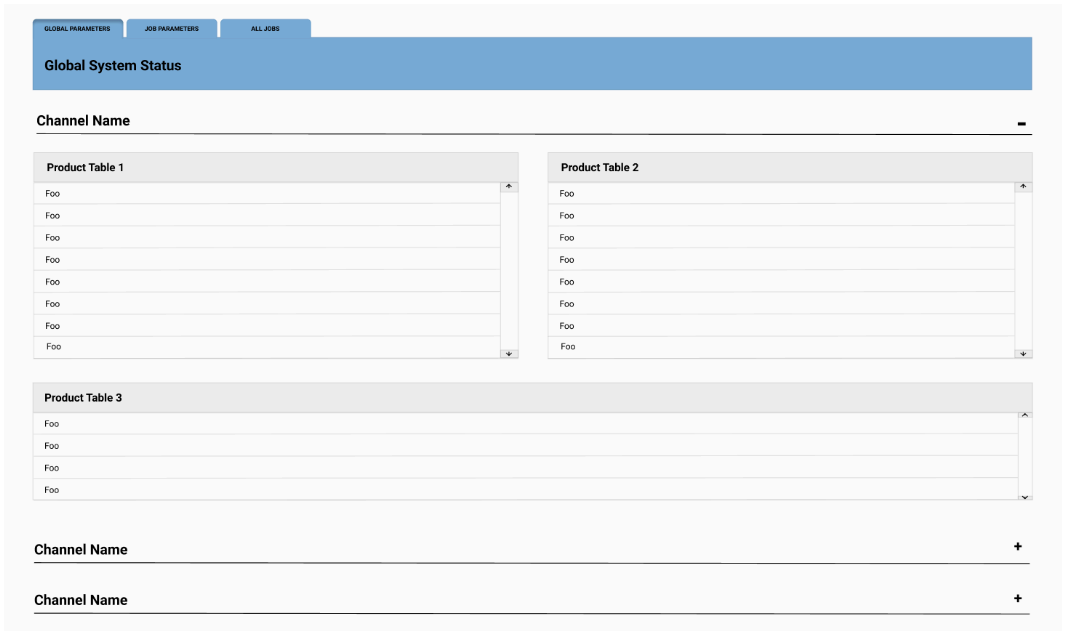
Global System Status

Channel 1 Name
Product 1
Product 2
Product 3
Product 4
Product 5
Product 6
Product 7
Product 8

Channel 2 Name
Product 1
Product 2
Product 3
Product 4
Product 5
Product 6
Product 7
Product 8

Channel 3 Name
Product 1
Product 2
Product 3
Product 4

Figure 4: Global Page Prototype 2



Global System Status

Channel Name -

Product Table 1
Foo
Foo
Foo
Foo
Foo
Foo
Foo
Foo

Product Table 2
Foo
Foo
Foo
Foo
Foo
Foo
Foo
Foo

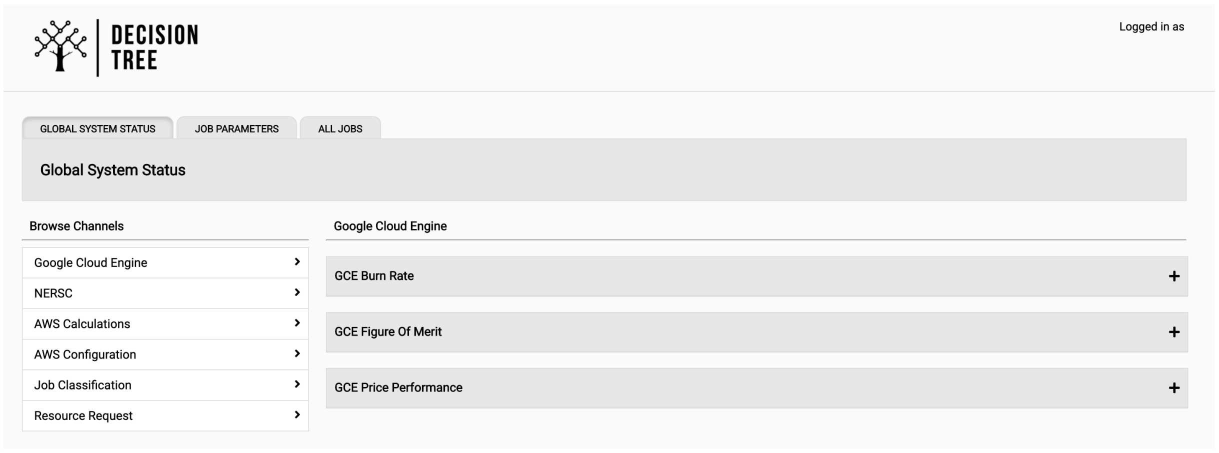
Product Table 3
Foo
Foo
Foo
Foo

Channel Name +

Channel Name +

Figure 5: Global Page Prototype 3

The final designs for the global page, job parameters page and the all jobs page are included as Figures 6, 7, and 8, respectively:



Global System Status

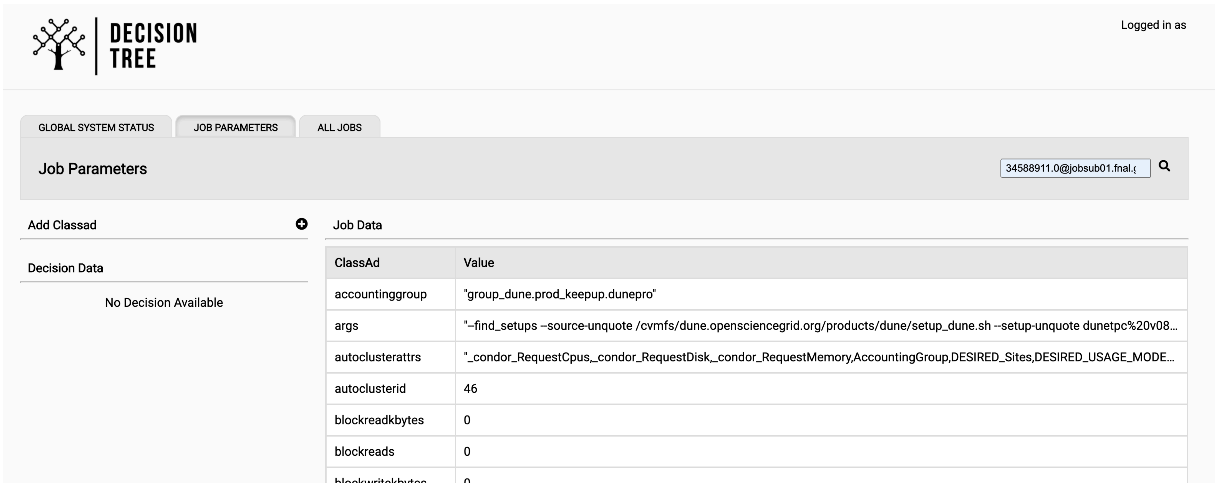
Browse Channels

- Google Cloud Engine
- NERSC
- AWS Calculations
- AWS Configuration
- Job Classification
- Resource Request

Google Cloud Engine

- GCE Burn Rate +
- GCE Figure Of Merit +
- GCE Price Performance +

Figure 6: Global Page Final Design



Job Parameters

34588911.0@jobsub01.fnal.(

Add Classad

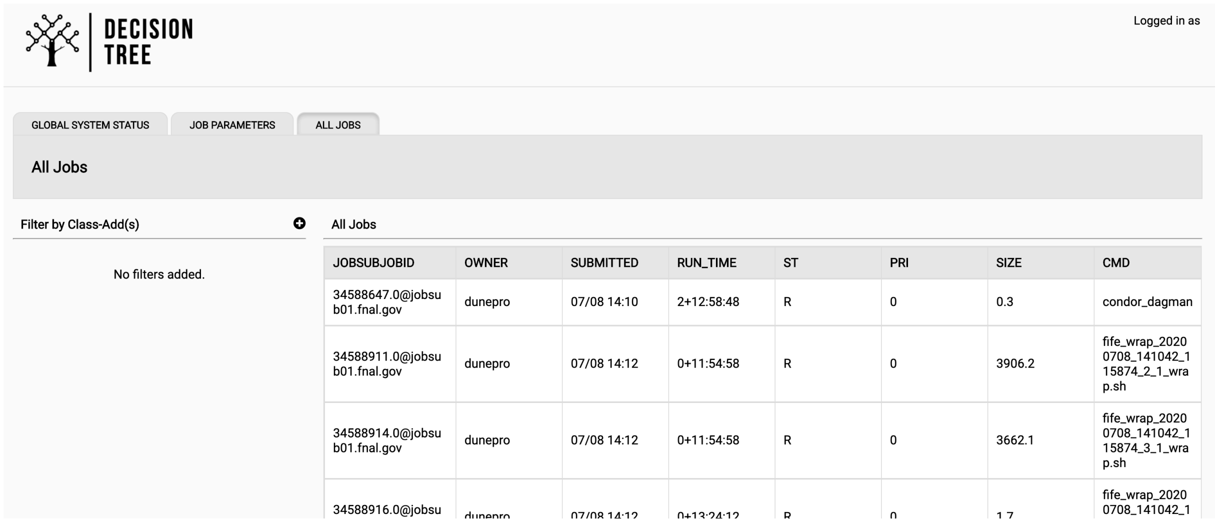
Decision Data

No Decision Available

Job Data

ClassAd	Value
accountinggroup	"group_dune.prod_keepup.dunepro"
args	"--find_setups --source-unquote /cvmfs/dune.opensciencegrid.org/products/dune/setup_dune.sh --setup-unquote dunetpc%20v08…
autoclusterattrs	"_condor_RequestCpus,_condor_RequestDisk,_condor_RequestMemory,AccountingGroup,DESIRED_Sites,DESIRED_USAGE_MODE…
autoclusterid	46
blockreadkbytes	0
blockreads	0
blockwritekbytes	0

Figure 7: Job Parameters Page Final Design



All Jobs

Filter by Class-Add(s)

No filters added.

All Jobs

JOBSUBJOBID	OWNER	SUBMITTED	RUN_TIME	ST	PRI	SIZE	CMD
34588647.0@jobsub01.fnal.gov	dunepro	07/08 14:10	2+12:58:48	R	0	0.3	condor_dagman
34588911.0@jobsub01.fnal.gov	dunepro	07/08 14:12	0+11:54:58	R	0	3906.2	fife_wrap_20200708_141042_115874_2_1_wrap.sh
34588914.0@jobsub01.fnal.gov	dunepro	07/08 14:12	0+11:54:58	R	0	3662.1	fife_wrap_20200708_141042_115874_3_1_wrap.sh
34588916.0@jobsu	dunepro	07/08 14:12	0+13:24:12	R	0	1.7	fife_wrap_20200708_141042_1

Figure 8: All Jobs Page Final Design

5.3 Backend

While not the most outward-facing component, the backend of any application powers the frontend by providing the interface with digestible data for rendering.

The backend configuration for "Decision Tree" is suited for development purposes only. This includes the overall system architecture, API implementation, and data sources. As the application transitions from proof-of-concept to production work, the backend must be altered within all three sections to more efficiently handle incoming data and return it to the application in an acceptable format.

5.3.1 Server Configuration

There are four main entities in the server configuration as shown in Figure 9: The Decision Tree Application, the local API, the local data directory, and Fermilab's Shibboleth Identity Provider.

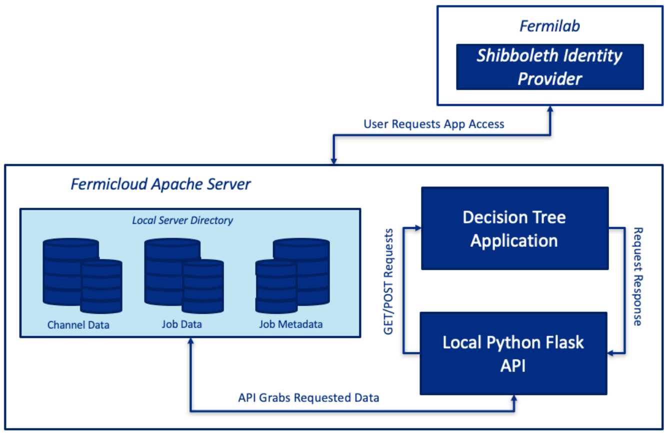


Figure 9: Development Server Architecture

The Decision Tree Application is the user-facing element of the four units. As the user interacts with the interface, the application communicates with the local API to retrieve data and present it to the user. The system also connects with the Fermilab to authenticate users prior to their access to the application. It is also by this means by which the API has the ability to query jobs according to a user's Fermilab ID.

5.3.2 API Configuration

The API is one of the most important components of the software architecture. It transmits and transforms the data from the local server directory to the application backend. Currently, the API serves as a hub for access to local Decision Engine channel data and HTCondor job data and metadata.

In total, there are six endpoints built into this service detailed in Table 2:

Table 2 – API Endpoints

Endpoint	Request Type	Purpose
/get-resource/:channelName	GET	Gets the data associated with a channel
/get-resource/job-data	POST	Gets a specific job
/get-resource/channel-list	GET	Gets the list of channels in the Decision Engine
/jobs	POST	Gets a paginated list of jobs associated with a user
/update-job	POST	Adds, removes, or modifies a classad
/update-channel	POST	Modifies a channel property

Table 2 – API Endpoints (continued)

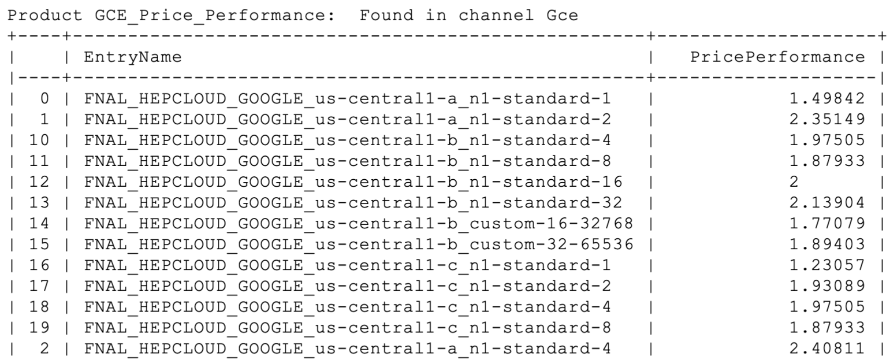
Endpoint	Parameters	Example
/get-resource/:channelName	**:channelName** – (string) name of channel to query	N/A
/get-resource/job-data	**Id** – (string) jobsubjobid	{ “id”: “0000@jobs.fnal.gov” }
/get-resource/channel-list	None	N/A
/jobs	**start** – (int) pagination starting point **count** – (int) number of jobs to return **filters** – (array) array of user-defined filters	{ start: 0, count: 20, filters: [{“hello”: “world”}] }
/update-job	**action** – “add”, “modify”, or “remove” **key** – classad key **value** – classad value **id** – jobsubjobid	{ “action”: “add”, “key”: “hello”, “value”: “world”, “id”: “0000@jobs.fnal.gov” }
/update-channel	**file** – (string) name of channel **action** – (string) “modify” **table** – (string) name of channel **row_index** – (int) index of modified row **obj** – (dict) key-value representation of changed row item	{ “file”: “gce-transforms” “action”: “modify” “table”: “AWS_Burn_Rate” “row_index”: 0 “obj”: { “BurnRate”: 0 } }

The API also communicates with the application's own web server in order to retrieve authentication information stored by Shibboleth identity provider. Authentication information is available by making a GET request to the following example URL: *http://sample-web-application.com/Shibboleth.sso/Session.*

5.3.3 Application Data

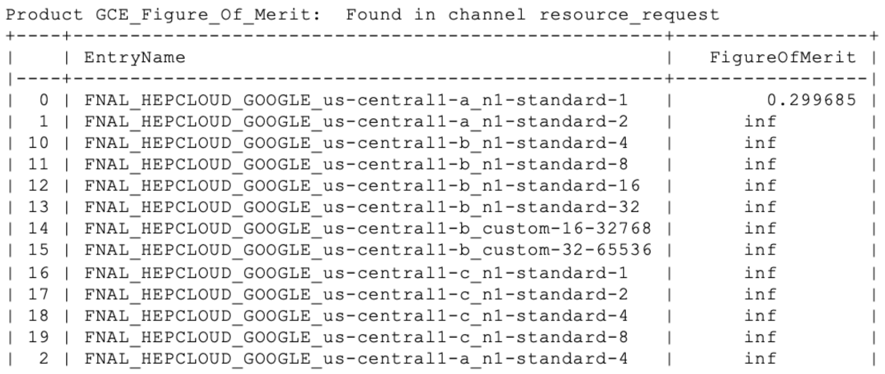
All of the data rendered on the webpages is static. Information was pulled directly from the Decision Engine and HTCondor to emulate a working piece of software. After harvesting the data, it was then converted to JSON format for easy importing and parsing.

When retrieving channel data from the Decision Engine, this information often produces several text files as depicted in Figures 10 and 11. This portion of data collectively belongs to the Google Cloud Engine channel.



```
Product GCE_Price_Performance:  Found in channel Gce
+----+---------------------------------------------------+--------------------+
|    | EntryName                                         |   PricePerformance |
|----+---------------------------------------------------+--------------------|
|  0 | FNAL_HEPCLOUD_GOOGLE_us-central1-a_n1-standard-1   |            1.49842 |
|  1 | FNAL_HEPCLOUD_GOOGLE_us-central1-a_n1-standard-2   |            2.35149 |
| 10 | FNAL_HEPCLOUD_GOOGLE_us-central1-b_n1-standard-4   |            1.97505 |
| 11 | FNAL_HEPCLOUD_GOOGLE_us-central1-b_n1-standard-8   |            1.87933 |
| 12 | FNAL_HEPCLOUD_GOOGLE_us-central1-b_n1-standard-16  |            2       |
| 13 | FNAL_HEPCLOUD_GOOGLE_us-central1-b_n1-standard-32  |            2.13904 |
| 14 | FNAL_HEPCLOUD_GOOGLE_us-central1-b_custom-16-32768 |            1.77079 |
| 15 | FNAL_HEPCLOUD_GOOGLE_us-central1-b_custom-32-65536 |            1.89403 |
| 16 | FNAL_HEPCLOUD_GOOGLE_us-central1-c_n1-standard-1   |            1.23057 |
| 17 | FNAL_HEPCLOUD_GOOGLE_us-central1-c_n1-standard-2   |            1.93089 |
| 18 | FNAL_HEPCLOUD_GOOGLE_us-central1-c_n1-standard-4   |            1.97505 |
| 19 | FNAL_HEPCLOUD_GOOGLE_us-central1-c_n1-standard-8   |            1.87933 |
|  2 | FNAL_HEPCLOUD_GOOGLE_us-central1-a_n1-standard-4   |            2.40811 |
```

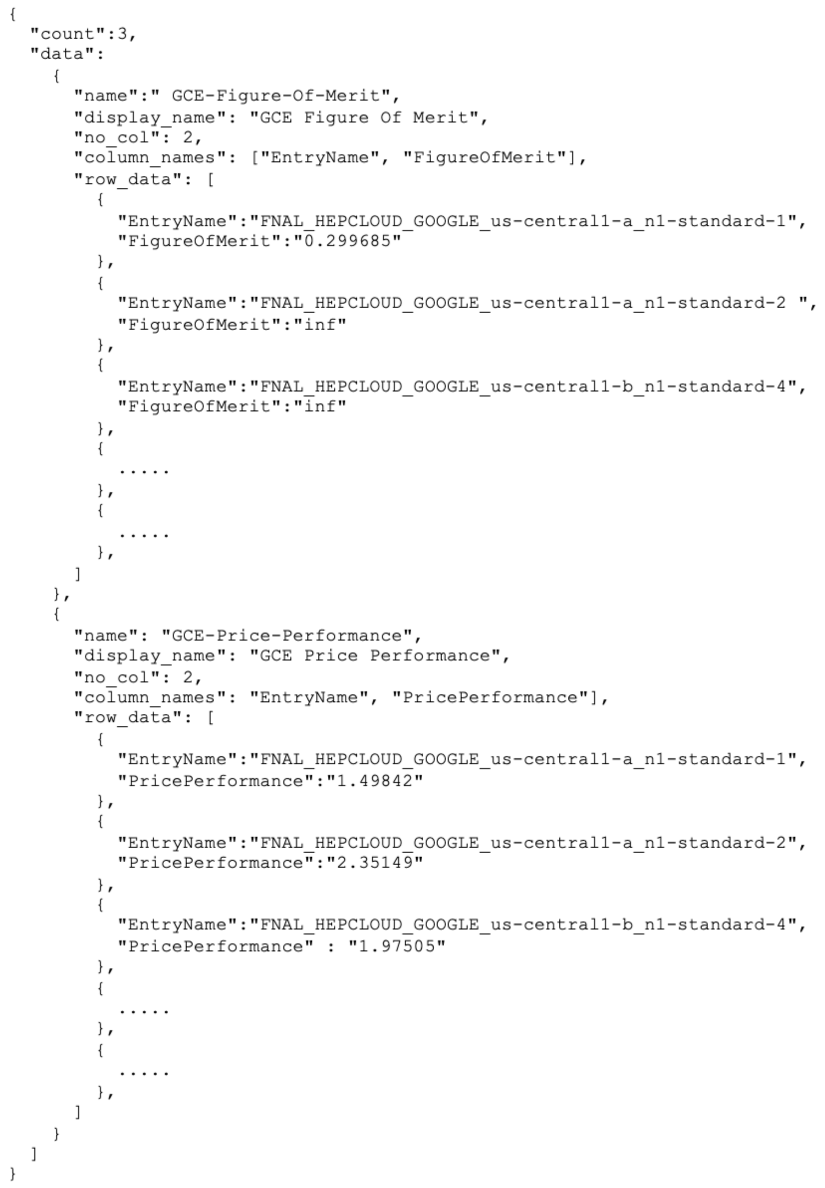
Figure 10: Price Performance Text File



```
Product GCE_Figure_Of_Merit:  Found in channel resource_request
+----+---------------------------------------------------+-----------------+
|    | EntryName                                         |   FigureOfMerit |
|----+---------------------------------------------------+-----------------|
|  0 | FNAL_HEPCLOUD_GOOGLE_us-central1-a_n1-standard-1   |        0.299685 |
|  1 | FNAL_HEPCLOUD_GOOGLE_us-central1-a_n1-standard-2   |      inf        |
| 10 | FNAL_HEPCLOUD_GOOGLE_us-central1-b_n1-standard-4   |      inf        |
| 11 | FNAL_HEPCLOUD_GOOGLE_us-central1-b_n1-standard-8   |      inf        |
| 12 | FNAL_HEPCLOUD_GOOGLE_us-central1-b_n1-standard-16  |      inf        |
| 13 | FNAL_HEPCLOUD_GOOGLE_us-central1-b_n1-standard-32  |      inf        |
| 14 | FNAL_HEPCLOUD_GOOGLE_us-central1-b_custom-16-32768 |      inf        |
| 15 | FNAL_HEPCLOUD_GOOGLE_us-central1-b_custom-32-65536 |      inf        |
| 16 | FNAL_HEPCLOUD_GOOGLE_us-central1-c_n1-standard-1   |      inf        |
| 17 | FNAL_HEPCLOUD_GOOGLE_us-central1-c_n1-standard-2   |      inf        |
| 18 | FNAL_HEPCLOUD_GOOGLE_us-central1-c_n1-standard-4   |      inf        |
| 19 | FNAL_HEPCLOUD_GOOGLE_us-central1-c_n1-standard-8   |      inf        |
|  2 | FNAL_HEPCLOUD_GOOGLE_us-central1-a_n1-standard-4   |      inf        |
```

Figure 11: Figure of Merit Text File

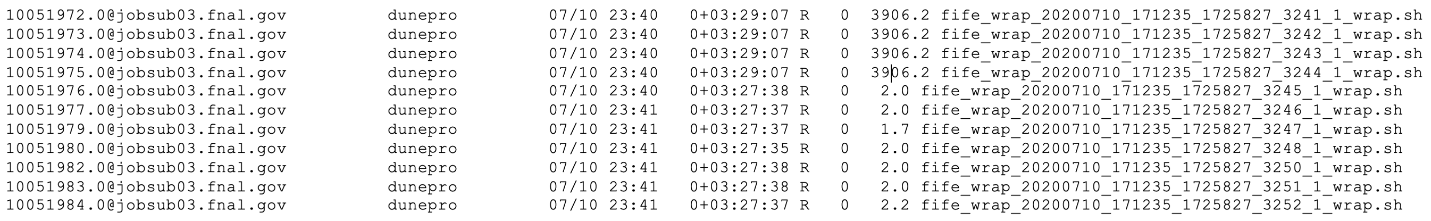
The channel files were then cleaned and converted into a much more workable format. In the case of the two files above, the expected JSON for this channel is:



```
{
  "count":3,
  "data":
    {
      "name":" GCE-Figure-Of-Merit",
      "display_name": "GCE Figure Of Merit",
      "no_col": 2,
      "column_names": ["EntryName", "FigureOfMerit"],
      "row_data": [
        {
          "EntryName":"FNAL_HEPCLOUD_GOOGLE_us-central1-a_n1-standard-1",
          "FigureOfMerit":"0.299685"
        },
        {
          "EntryName":"FNAL_HEPCLOUD_GOOGLE_us-central1-a_n1-standard-2 ",
          "FigureOfMerit":"inf"
        },
        {
          "EntryName":"FNAL_HEPCLOUD_GOOGLE_us-central1-b_n1-standard-4",
          "FigureOfMerit":"inf"
        },
        {
          .....
        },
        {
          .....
        },
      ]
    },
    {
      "name": "GCE-Price-Performance",
      "display_name": "GCE Price Performance",
      "no_col": 2,
      "column_names": "EntryName", "PricePerformance"],
      "row_data": [
        {
          "EntryName":"FNAL_HEPCLOUD_GOOGLE_us-central1-a_n1-standard-1",
          "PricePerformance":"1.49842"
        },
        {
          "EntryName":"FNAL_HEPCLOUD_GOOGLE_us-central1-a_n1-standard-2",
          "PricePerformance":"2.35149"
        },
        {
          "EntryName":"FNAL_HEPCLOUD_GOOGLE_us-central1-b_n1-standard-4",
          "PricePerformance" : "1.97505"
        },
        {
          .....
        },
        {
          .....
        },
      ]
    }
  ]
}
```

Figure 12: JSON-Formatted Channel Data

HTCondor, on the other hand, presents job data and job metadata in key-value and comma-separated value formats, respectively. HTCondor job data is represented in Figure 13 while the job metadata is in Figure 14.



```
10051972.0@jobsub03.fnal.gov          dunepro         07/10 23:40   0+03:29:07 R   0  3906.2 fife_wrap_20200710_171235_1725827_3241_1_wrap.sh
10051973.0@jobsub03.fnal.gov          dunepro         07/10 23:40   0+03:29:07 R   0  3906.2 fife_wrap_20200710_171235_1725827_3242_1_wrap.sh
10051974.0@jobsub03.fnal.gov          dunepro         07/10 23:40   0+03:29:07 R   0  3906.2 fife_wrap_20200710_171235_1725827_3243_1_wrap.sh
10051975.0@jobsub03.fnal.gov          dunepro         07/10 23:40   0+03:29:07 R   0  3906.2 fife_wrap_20200710_171235_1725827_3244_1_wrap.sh
10051976.0@jobsub03.fnal.gov          dunepro         07/10 23:40   0+03:27:38 R   0    2.0 fife_wrap_20200710_171235_1725827_3245_1_wrap.sh
10051977.0@jobsub03.fnal.gov          dunepro         07/10 23:41   0+03:27:37 R   0    2.0 fife_wrap_20200710_171235_1725827_3246_1_wrap.sh
10051979.0@jobsub03.fnal.gov          dunepro         07/10 23:41   0+03:27:37 R   0    1.7 fife_wrap_20200710_171235_1725827_3247_1_wrap.sh
10051980.0@jobsub03.fnal.gov          dunepro         07/10 23:41   0+03:27:35 R   0    2.0 fife_wrap_20200710_171235_1725827_3248_1_wrap.sh
10051982.0@jobsub03.fnal.gov          dunepro         07/10 23:41   0+03:27:38 R   0    2.0 fife_wrap_20200710_171235_1725827_3250_1_wrap.sh
10051983.0@jobsub03.fnal.gov          dunepro         07/10 23:41   0+03:27:38 R   0    2.0 fife_wrap_20200710_171235_1725827_3251_1_wrap.sh
10051984.0@jobsub03.fnal.gov          dunepro         07/10 23:41   0+03:27:37 R   0    2.2 fife_wrap_20200710_171235_1725827_3252_1_wrap.sh
```

Figure 13: HTCondor Job Data Text File

```
JobsubJobId = "001.0@jobs.fnal.gov"
JobsubJobSection = "4"
JobsubParentJobId = "001.0@jobs.fnal.gov"
JobsubServerVersion = "1.3.2.1"
JobUniverse = 5
KeepClaimIdle = 20
LastHoldReasonCode = 26
LastHoldReasonSubCode = 8
LastJobLeaseRenewal = 1594454895
LastJobStatus = 1
LastMatchTime = 1594406734
```

Figure 14: HTCondor Job Metadata Text File

When cleaned and converted into JSON, the application backend receives job data in a format as shown in Figure 15 and job metadata in a format shown in Figure 16.

```
[
  {
    "JOBSUBJOBID":"000.0@jobsub01.fnal.gov",
    "OWNER":"dunepro",
    "SUBMITTED":"07\/08 14:10",
    "RUN_TIME":"2+12:58:48",
    "ST":"R",
    "PRI":0,
    "SIZE":0.3,
    "CMD":"condor_dagman"
  },
  {
   "JOBSUBJOBID":"0001.0@jobsub01.fnal.gov",
   "OWNER":"dunepro",
   "SUBMITTED":"07\/08 14:12",
   "RUN_TIME":"0+11:54:58",
   "ST":"R",
   "PRI":0,
   "SIZE":3906.2,
   "CMD":"fife_wrap_20200708_141042_115874_2_1_wrap.sh”
  }
]
```

Figure 15: JSON-Formatted Job Data

```
[
  {
    ...
    “JobsubJobId” = "001.0@jobsub01.fnal.gov"
    “JobsubJobSection” = "4"
    “JobsubParentJobId” = "001.0@jobsub01.fnal.gov"
    “JobsubServerVersion” = "1.3.2.1"
    “JobUniverse” = 5
    “KeepClaimIdle” = 20
    ...
  },
  {
    ...
    “JobsubJobId” = "002.0@jobsub01.fnal.gov"
    “JobsubJobSection” = "4"
    “JobsubParentJobId” = "002.0@jobsub01.fnal.gov"
    “JobsubServerVersion” = "1.3.2.1"
    “JobUniverse” = 5
    “KeepClaimIdle” = 20
    ...
  }
  ...
]
```

Figure 16: JSON-Formatted Job Metadata

Regardless of the data's nature, it is important that the JSON returned to the application is in accordance with expected guidelines; otherwise, the frontend will not interpret the object correctly and will fail to display any information.

6. Technologies

To accomplish our goal of building a web application, we used a variety of technologies as listed and described in Table 3.

Table 3 – Development Technologies

Name	Purpose
Angular + TypeScript	Web Framework
HTML + CSS + JavaScript	Application Structure, Styles, and Behavior
Python	API Configuration
Apache	Web Server
Figma	Prototyping Tool
Git	Version Control

7. Development Challenges

Aside from the abrupt adjustment to remote work, the most challenging area in this project was gathering data.

The information required to make this application functional is not currently exposed by services at Fermilab via an API. The Decision Engine, in particular, does not have a method of retrieving decision data; therefore, the current iteration of this system does not include decision data for any of the jobs. Furthermore, although HTCondor can pull job data and metadata from its system, it also is not immediately available to external systems.

Additionally, finding a reasonable JSON schema for the Decision Engine's text representation of internal data proved difficult. The JSON format that the application currently uses is my interpretation of what the data *might* and *should* look like. This discretionary power may pose challenges when trying to integrate what is actually embedded in the Decision Engine versus what the application currently needs to function correctly. Any major discrepancies may require an overhaul of the frontend templating implementation.

8. Future Direction

Ideally, in the future, we would like to obtain Decision Engine and job data from their respective Fermilab services. This means that the application would become dynamic and also include the missing Decision Data we could not obtain to fulfil requirement 7 identified in Section 4.2. Figure 17 depicts an updated version of Figure 9 in Section 5.3.1 where data is now pulled from HTCondor and the HEPCloud Decision engine as opposed to a local server directory.

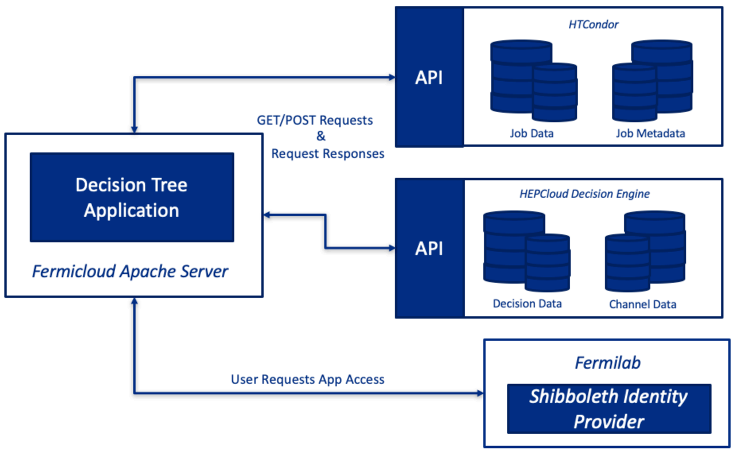


Figure 17: Future Server Architecture

Additionally, we hope to construct a system of privileges depending on the type of authentication attributed to a logged in user. In this scenario, certain users will be able to modify the global state of the Decision Engine and/or configure jobs that belong to other accounts. By building this new system we hope to further maintain system integrity by giving power to those who may require more oversight. Finally, expanding access to this application beyond DUNE is a definite roadmap item, especially considering that the black-box challenges that end-users face is universal and not experiment-specific.

9. References

[1] Deep Underground Neutrino Experiment (DUNE) Homepage: https://www.dunescience.org/

[2] K. Herner *et al.*, J. Phys.: Conf. Ser. **898**, 052026 (2017).

[3] D. Box, J. Phys.: Conf. Ser. 513, 032010 (2014).

[4] B. Holzman *et al.*, Comp. Softw. for Big Sci. **1**, 1 (2017).

[5] Worldwide Large Hadron Collider Computing Grid Homepage: https://wlcg.web.cern.ch/

[6] HEPCloud Homepage: https://hepcloud.fnal.gov/

[7] P. Mhashilkar *et al.*, EPJ Web of Conf. **214**, 03060 (2019).

[8] Fermilab HEPCloud Facility Decision Engine Design: https://lss.fnal.gov/archive/test-tm/2000/fermilab-tm-2654-cd.pdf

[9] HEPCloud Decision Engine GitHub Page: https://github.com/HEPCloud/decisionengine