# User Storage Quotas and MFT Dashboard

**NOTE: DO NOT CHANGE THE HEADING**

**PROJECT TITLE :**

Implementing Storage limits for multiple types of cloud storage and a dashboard to track file transfer done through MFT.

**GOAL OF THE PROJECT 1:**

Airavata based science gateways store data in the gateway storage which can be mounted in the portal's hosting server or can be stored on third-party cloud storage. Each user's data is organized within user directories on these storage devices. As storage reaches its limits, it often creates an issue in rationing the disks. So, a single user running a finite number of experiments can adversely cause issues for other users in the gateway.

This is very similar to modern memory attacks. A hacker can write a script that creates new experiments on each iteration, causing the storage space to be depleted.

(Jira link: https://issues.apache.org/jira/browse/AIRAVATA-3315)

The goal of the EPIC is to mitigate such scenarios by implementing a storage limit for each user in a gateway. This way, one user can never adversely affect other users.

**GOAL OF THE PROJECT 2:**

MFT is a new service that takes care of transferring files between any two resources. The goal of this project is to develop a new dashboard for MFT to track old requests and initiate new requests. Once the source and destination storages are chosen, the user navigates through the directories of source storage to choose a file and then chooses the destination directory in the destination storage. Post submitting the request to MFT, the dashboard keeps on polling the backend to track the status of the requests.

**DELIVERABLES:**

The work is divided in such a way that each milestone delivers a small functionality.

| TIMELINE | Milestones |
| --- | --- |

| | |
|---|---|
| May 11th - May 15th | **Milestone 0**<br><br>• Setup the codebase.<br>• Update the original GSoC proposal. |
| 16th May - June 29th | **Milestone 1**<br><br>• Interacting with the Mentor and Airavata team.<br>• Familiarizing myself with the Django framework, Apache-Thrift, VueJS, Vuex, JavaScript, and codebases of the Django portal and Airavata.<br>• Finalize requirements for each deliverable and where each will fit within Airavata.<br>• Adding a configurable user-specific storage entity in the gateway's resource profile.<br>• UI changes to accommodate the new entry.<br>• Get the code reviewed and push the changes.<br>• Finalize a tracking mechanism. For now, the first mechanism seems more straight forward and plausible. |
| June 30th - July 27th | **Milestone 2**<br><br>• Complete leftover work and implement code review changes if any.<br>• Delineate all the use cases which require data to be stored in the gateway storage.<br>Ex: An input and Output file stored in the user directory.<br>• Develop the finalized tracking mechanism in Airavata to track all the use cases that require memory on the gateway storage.<br>• If the second tracking mechanism is finalized, there will be more work involved and MIGHT be spilled over to Milestone 3.<br>• Add the mechanism to detect and show an error popup in the UI for the case where the user exceeds his allocated limit.<br>• Along with unit tests (which will be a part of TDD), write integration tests.<br>• Get the code reviewed and push the changes. |

| | |
|---|---|
| July 28th - Aug 24th | **Milestone 3**<br><br>• Complete leftover work and implement code review changes if any.<br>• Come up with a mock UI for the new MFT dashboard.<br>• Come up with new APIs needed for the dashboard and design them.<br>• Integrate these MFT DjangoPortal.<br>• Get everything evaluated and implement code review changes if any. |

**SOLUTION FOR PROJECT 1:**

## Solutions considered and why they weren't viable:

1. Track the user's storage space on the scientific gateway.
    - Add a configurable user-specific storage entry in Storage Preference.
    - Track the size of user storage in the gateway.
    - Validate the storage quota by querying Airavata for the storage limit per Storage preference.
    - This is the easiest solution. But the problem is, every scientific gateway needs to implement its own storage tracking mechanism (though it's fairly simple).
2. Track the storage space in Airavata.
    - Add a configurable user-specific storage entry in Storage Preference.
    - Add a new entry to the User table in experiment_catalog for tracking user storage.
    - Whenever an experiment is created with some inputs, Airavata, using SCP transfer, transfers these files from the storage to compute resources. Before doing this, compute the size of the file being transferred and keep on adding it to the storage tracking number.
    - File transfer is done in a different microservice, so this would require, configuring a new persistent storage to track every user's storage utilization.
    - This microservice will be deprecated in the future and will be replaced by MFT, so this solution isn't viable either.

**Final solution**:

As part of the Epic, I’ve developed a new tracking capability to implement user-specific quotas within Airavata for each user and track their usage.

This involved:

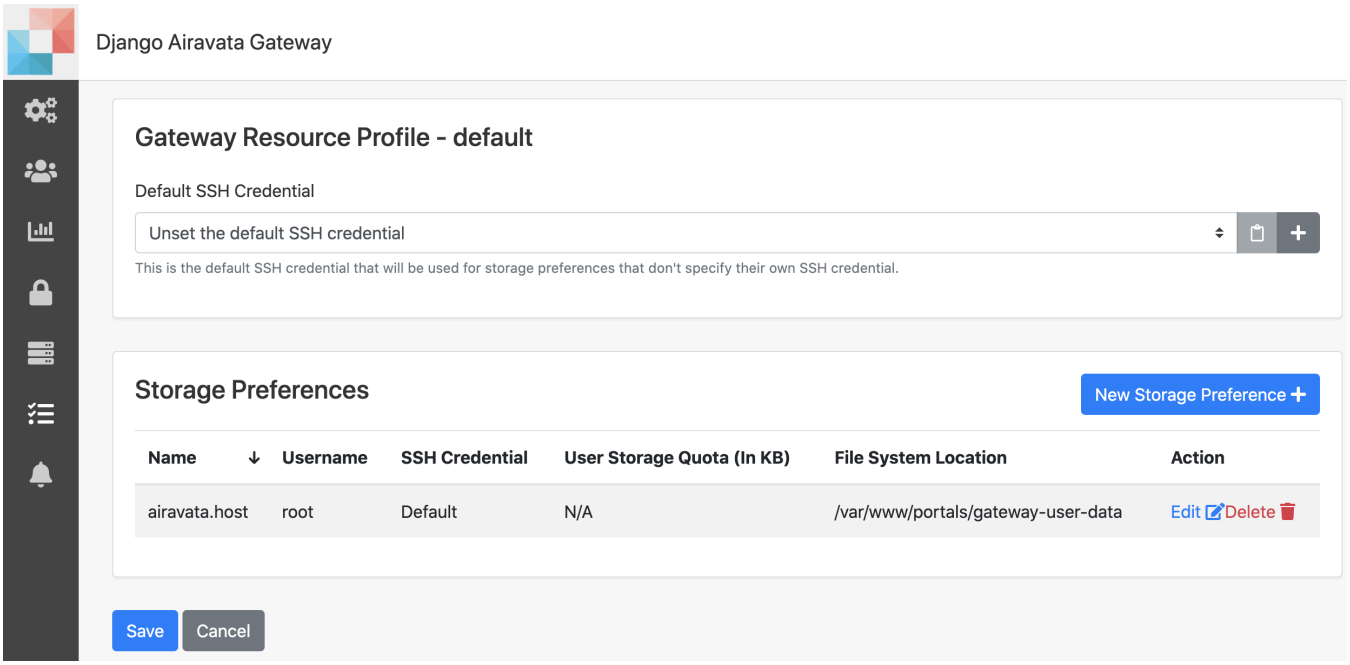
1. Adding a configurable user-specific storage per gateway.
    - This involved changes in Airavata-DjangoPortal to accommodate a new entry, *User Storage Preference*, to take input from the user for the storage limit.
    - Configuring a new entry in STORAGE_PREFERENCE table for the new UserStoragePreference entry.
2. All the user inputs are stored inside a user directory. So, just before creating an experiment, using a pool of SSH connections, Airavata executes a 'du command' on the storage preference to know the size of the user directory.
3. Airavata then validates this size against the storage quota specified in the StoragePreference.

The earlier approach needs every gateway to have its own storage tracking mechanism and the second mechanism cannot factor changes in the user directory which doesn't come through an experiment.

Considering all this, the above solution seemed like the best approach.

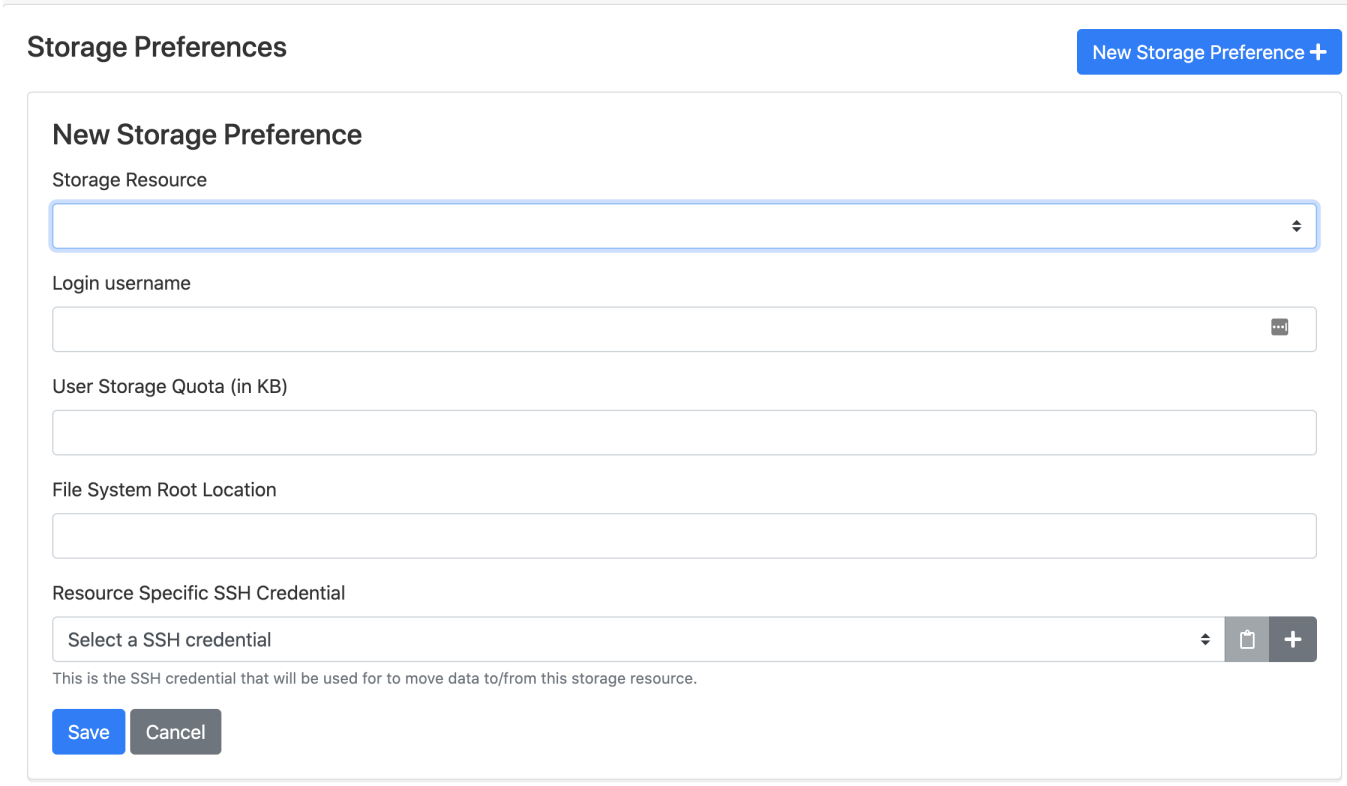
#### Deliverables for 1st Project:

By default, previous entries and gateways that do not have the User Storage quota defined displays the User Storage Quota as 'N/A', which would mean there are no storage restrictions for a user in this gateway.



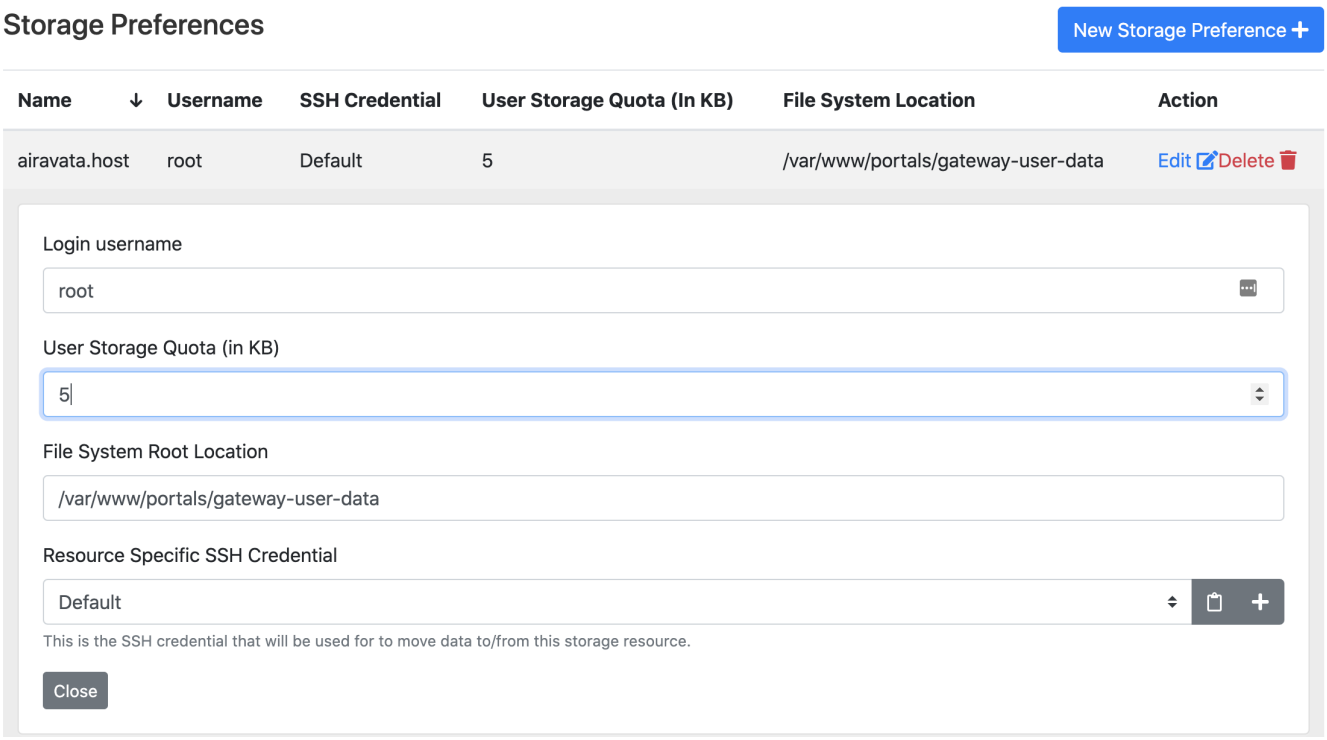

## New Storage Preference:

A new User Storage Quota entry is added to the New Storage preference component. Restrictions are added so that the User Storage Quota entry can only be a positive number. Input '0' would display 'N/A' resulting in no storage restrictions.



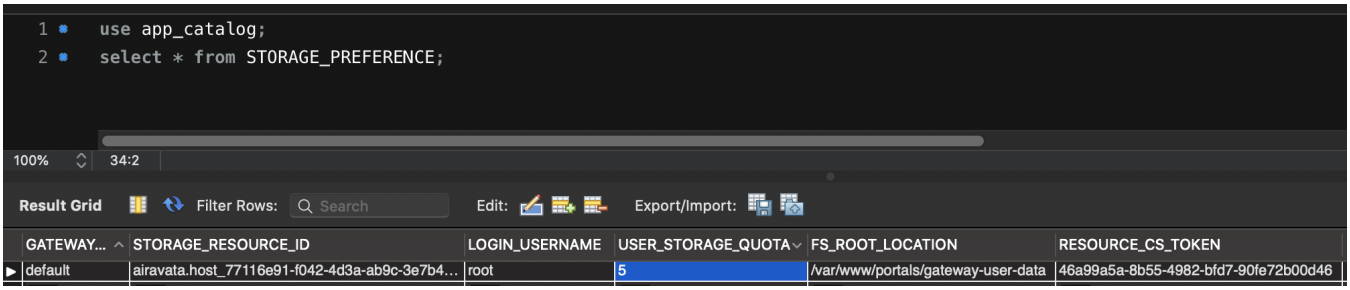

## Edit existing storage preference:

A new User Storage Quota entity is added to edit an existing Storage quota.




## Changes in Airavata - StoragePreference table:

Changes are made to propagate the new entry from Frontend DjangoPortal Airavata and then store it in the STORAGE_PREFERENCE table.




## First-month report:

| Dates (Monday - Friday) | Work | References (Can include mails and PRs) |
| --- | --- | --- |
| May 15th to May 29th | • Learning Python, Django Framework, Thrift. | |
| June 1st to June 5th | • Learning JavaScript, VueJS, Vuex, and understanding the codebase | |
| June 8th to June 12th | • Started coding.<br>• This week involved a lot of figuring out the codebase and functionalities of Airavata.<br>• Basic changes to input a new entry to the Gateway Resource Profile page in StoragePreference were made.<br>• Changes to Thrift generated files in Airavata to read the new entry.<br>• Manually made changes to *ttypes.py* to read *UserStorageQuota* value sent from the frontend. | • https://markmail.org/message/e5n653nf23yp3czc?q=shresta+order:date-backward&page=1<br>• https://markmail.org/message/iefe6fnp4bjllkfd?q=shresta+order:date-backward&page=1 |
| June 15th to June 19th | | |

- Figured out how client stubs are generated and used in the Django portal. So instead of manually changing *ttypes.py*, replaced it with the newly generated thrift files.
- Most of my work this week was figuring out how to make a schema change in Airavata. Several conversations took place to get a proper understanding of how an extra entry is added to a table without any data loss in Airavata.
- Added an extra functionality where if an admin inputs the value 0 for UserStorageQuota, Airavata makes sure not to apply any storage restrictions and instead displays the value 'N/A' in frontend.

- https://markmail.org/message/ugomc6qf2i5ybnje?q=shresta+order:date-backward&page=1
- https://github.com/apache/airavata/pull/254
- https://markmail.org/message/dde573xpgp6ok7sd?q=shresta+order:date-backward&page=1

  **AIRAVATA-3343** - Add configurable user-specific storage quota entry OPEN
- https://markmail.org/message/pameotck52czrccf?q=shresta+order:date-backward&page=1

- https://markmail.org/message/nh3qeumx2yibozxm?q=shresta+order:date-backward&page=1
- https://markmail.org/message/7uft7lcfzi6mwrzo?q=shresta+order:date-backward&page=1
- https://markmail.org/message/rmszakbizxkylwf5?q=vivekshresta+order:date-backward
- https://markmail.org/message/hnuqtmnfwv5fezsg?q=vivekshresta+order:date-backward
- https://markmail.org/message/jxy7ik6piximeht2?q=vivekshresta+order:date-backward

| | | |
|---|---|---|
| | | • https://markmail.org/message/g43zrm33lbkz4wlu?q=vivekshresta+order:date-backward |
| June 22nd to June 26th | • Got a better understanding on how to introduce schema changes in Airavata, addressed the code review changes, and got them verified.<br>• Finalized and raised Pull Requests to Airavata Django Portal and Airavata codebases.<br>• Completed milestone 1. | • https://github.com/apache/airavata-django-portal/pull/42 (Merged)<br>• https://github.com/apache/airavata/pull/255 (Merged) |

**Second-month report:**

| Work | References (Can include mails and PRs) |
|---|---|

| | |
|---|---|
| • Learning the codebase around storage preferences in Airavata (References include issues that I've raised which were used in the codebase learning) | • https://apache.markmail.org/message/rwveaccv4ufakr2o?q=shresta+order:date-backward&page=2<br>• https://apache.markmail.org/message/bcdxnbtgmokm263w?q=shresta+order:date-backward&page=3<br>• https://apache.markmail.org/message/msnkqjhyrn5rkhdm?q=shresta+order:date-backward&page=3<br>• https://apache.markmail.org/message/fgyovutukaj2mvl6?q=shresta+order:date-backward&page=3<br>• https://apache.markmail.org/message/grpvxhbqzkah47cq?q=shresta+order:date-backward<br>• https://apache.markmail.org/message/n2mihsfw4u5vmspa?q=shresta+order:date-backward&page=3<br>• https://apache.markmail.org/message/x2nc5sn3aopueoze?q=shresta+order:date-backward&page=3#query:shresta%20order%3Adate-backward+page:3+mid:yyeklkfg7vd7rqce+state:results<br>• https://apache.markmail.org/message/x2nc5sn3aopueoze?q=shresta+order:date-backward&page=3#query:shresta%20order%3Adate-backward+page:3+mid:rwveaccv4ufakr2o+state:results<br>• https://apache.markmail.org/message/x2nc5sn3aopueoze?q=shresta+order:date-backward&page=3#query:shresta%20order%3Adate-backward+page:3+mid:toko4rmjspqb4nk5+state:results<br>• https://apache.markmail.org/message/my5w6cw7ia7hskga?q=shresta+order:date-backward&page=2 |

| | |
|---|---|
| • Discussions with the team during the development of the storage tracking mechanism. | • https://apache.markmail.org/message/z2t5vmsifeecklxa?q=shresta+order:date-backward&page=2<br>• https://apache.markmail.org/message/z2t5vmsifeecklxa?q=shresta+order:date-backward&page=2#query:shresta%20order%3Adate-backward+page:2+mid:eazttm3z2724c7gz+state:results<br>• https://apache.markmail.org/message/z2t5vmsifeecklxa?q=shresta+order:date-backward&page=2#query:shresta%20order%3Adate-backward+page:2+mid:opplnx5xidfoskkt+state:results<br>• https://apache.markmail.org/message/z2t5vmsifeecklxa?q=shresta+order:date-backward&page=2#query:shresta%20order%3Adate-backward+page:2+mid:4lbxwkchn3akxypd+state:results<br>• https://apache.markmail.org/message/z2t5vmsifeecklxa?q=shresta+order:date-backward&page=2#query:shresta%20order%3Adate-backward+page:2+mid:k3dxxqmqqhzeuznt+state:results<br>• https://apache.markmail.org/message/mqvyasg3feenbcfo?q=shresta+order:date-backward&page=1 |
| • PR's (Not including any commits since the Pull requests have all the commits and changes made) | • https://github.com/apache/airavata-django-portal/pull/48<br>• https://github.com/apache/airavata/pull/260<br>• https://github.com/apache/airavata/pull/257 |

**Communication before and during the community bonding period:**

https://markmail.org/message/aoagl3mqehhzfsdp?q=shresta+order:date-backward&page=3

https://markmail.org/message/mmfiwg2a2kfqs6j7?q=shresta+order:date-backward&page=2

https://markmail.org/message/jpyushuqxuix7zn2?q=shresta+order:date-backward&page=2

https://markmail.org/message/7odulcua6bxoktvb?q=shresta+order:date-backward&page=3

https://markmail.org/message/bpzkagpjvowwesch?q=shresta+order:date-backward&page=3

https://markmail.org/message/vn4zjvqmnamyoqxy?q=shresta+order:date-backward&page=5

https://markmail.org/message/oojdzfmczss2gepr?q=shresta+order:date-backward&page=4

https://markmail.org/message/6cupd6ac3m2ggs7t?q=shresta+order:date-backward&page=6

https://markmail.org/message/gddwtkvtgnhjjt5x?q=shresta+order:date-backward&page=6

https://markmail.org/message/xyvtsr6zss5m6hvy?q=shresta+order:date-backward&page=6

**Note**: Once the code review comments are addressed, project 1 should be completed. My development of this project can be found here:
https://github.com/vivekshresta/airavata
https://github.com/vivekshresta/airavata-django-portal

**SOLUTION FOR PROJECT 2:**

As part of developing MFT dashboard, develop all the backend gRPC APIs that are required.

The main features in this dashboard are:

1. List all the transfers that are done through MFT and that are currently in progress.
2. Given a list of storage resources, choose the source and destination storage resource.
3. Once the storages are chosen, navigate through the directories of the source storage to choose a file.
4. Navigate through the destination storage to choose the final path of the file.

Changes that were done as part of this development:

- **Persistent storage**: Previously, only transfers in progress could be tracked through MFT. Added persistent storage support in controller service to store TransferRequests which are initiated. DB's can be easily changed since I used jpa to talk to DB. Also, changes are made to track the status of a request, and the time taken to reach that status are also added.
- **Grpc support for controller service**: The current communication between the API service and controller service is asynchronous and done through consul. When a client requests for the details of all transfer requests or a single transfer request, the call needs to be synchronous. Added gRPC support for controller service. Made a slight change in the way the gRPC stubs are stored.

**Completed work**:

- Sat with the team and came up with a mock UI for the dashboard along with another GSoC member.
- The finalized mock UI can be found here: https://drive.google.com/file/d/1FfvjnMeXemQq7X1KcmHjVm3NICBLIP-w/view?usp=sharing
- Reviewed requirements and uses of various functionalities in MFT Portal.
- Came up with the list of APIs that are required and are present to deliver the first cut of the dashboard.
- Finalized the new necessary APIs. Understood the codebase.
- Added persistent storage to the controller service. The configured DB can be easily changed.

- Added synchronous communication for the controller service by developing gRPC support.
- Added retrieving all transfers and fetching a particular request APIs.
- Changes are yet to be merged.

**Pending work**:

- Integration with MFT Django-Portal.
- API to get all the storage preferences.

The remaining development can be tracked here: https://github.com/vivekshresta/airavata-mft

**References**:

https://apache.markmail.org/search/?q=shresta#query:shresta%20order%3Adate-backward+page:1+mid:re7curjrjujdmiwf+state:results

https://github.com/apache/airavata-mft/pull/26

**RELATED WORK:**

As part of the GOAL of making Pega Cloud Compatible (the company I was working in before joining IU), I developed a system to track the number of Concurrent sessions a user can have.

Pega can be used to develop various applications or websites. The above-mentioned feature helps Pega the following ways:

1. Depending on the user privileges, the user should be able to use or host only a fixed number of pega instances at a time.
2. Pega sessions consume a lot of memory. A hacker armed with this knowledge can create multiple pega sessions with the same user credentials to cause a memory attack and creating multiple EC2 because of the load. This feature helps in mitigating such use cases.

**BIOGRAPHICAL INFORMATION:**

Name: Vivek Shresta Bandaru

Email: vivekshrestabandaru@gmail.com, vivband@iu.edu

## Education :

| | |
|---|---|
| August 2019 - May 2021 (expected ) | **Indiana University Bloomington**<br>MS in Computer Science - 3.9/4 |
| August 2012 - May 2016 | **National Institute of Technology, Warangal (NITW)**<br><br>B.Tech in Computer Science |

**Programming Languages:** Java, Python, Javascript, React, C++, C, SQL, HTML5

**Technologies/Frameworks:** Spring, SpringBoot, Google Guice, Docker, Kubernetes, Apache Kafka, MongoDB, PostgreSQL, MySQL, Heroku, Jenkins, Gradle, Maven, JUnit, Mockito, Git, YourKit, Log4j, PRPC, Fiddler