STScI | SPACE TELESCOPE SCIENCE INSTITUTE

EXPANDING THE FRONTIERS OF SPACE ASTRONOMY

The JWST Quicklook Application (JWQL)

JWQL

Matthew Bourque
(Technical Lead)

Francesca Boffi
(Project Manager)

Lauren Chambers
(FGS)

Misty Cracraft
(MIRI)

Joseph Filippazzo
(NIRISS)

Bryan Hilbert
(NIRCam)

Graham Kanarek
(NIRSpec)

Catherine Martlin
(WFC3)

Sara Ogaz
(DATB)

Johannes Sahlmann
(NIRISS)

Acknowledgements

INS

Tracy Beck

Rosa Diaz

Van Dixon

Scott Friedman

Alex Fullerton

Catherine Gosmeyer

Dean Hines

Sherrie Holfeltz

Margaret Meixner

Massimo Robberto

Linda Smith

DMD

Faith Abney

Anastasia Alexov

Sara Anderson

Clara Brasseur

Matthew Burger

Steven Crawford

James Davies

Tom Donaldson

Lisa Gardner

Joe Hunkeler

Catherine Kaleida

Mark Kyprianou

Karen Levay

Lee Quick

Matt Rendina

Mary Romelfanger

Bernie Shiao

Geoff Wallace

SCOPE

Crystal Mannfolk

SEITO

Maria Nieto-Santisteban

ITSD

Vera Gibbs

Phil Grant

Greg Masci

Prem Mishra

Don Meuller

Anupinder Rai

Matthew Sienkiewicz

Patrick Taylor

Dave Unger

Thomas Walker

Alex Yermolaev

Joe Zahn

OPO

Lara Wilkinson

DSMO

Michael Fox

Arfon Smith

JWSTMO

Joe Pollizzi

Jeff Valenti

JWQL

Goals & Motivation

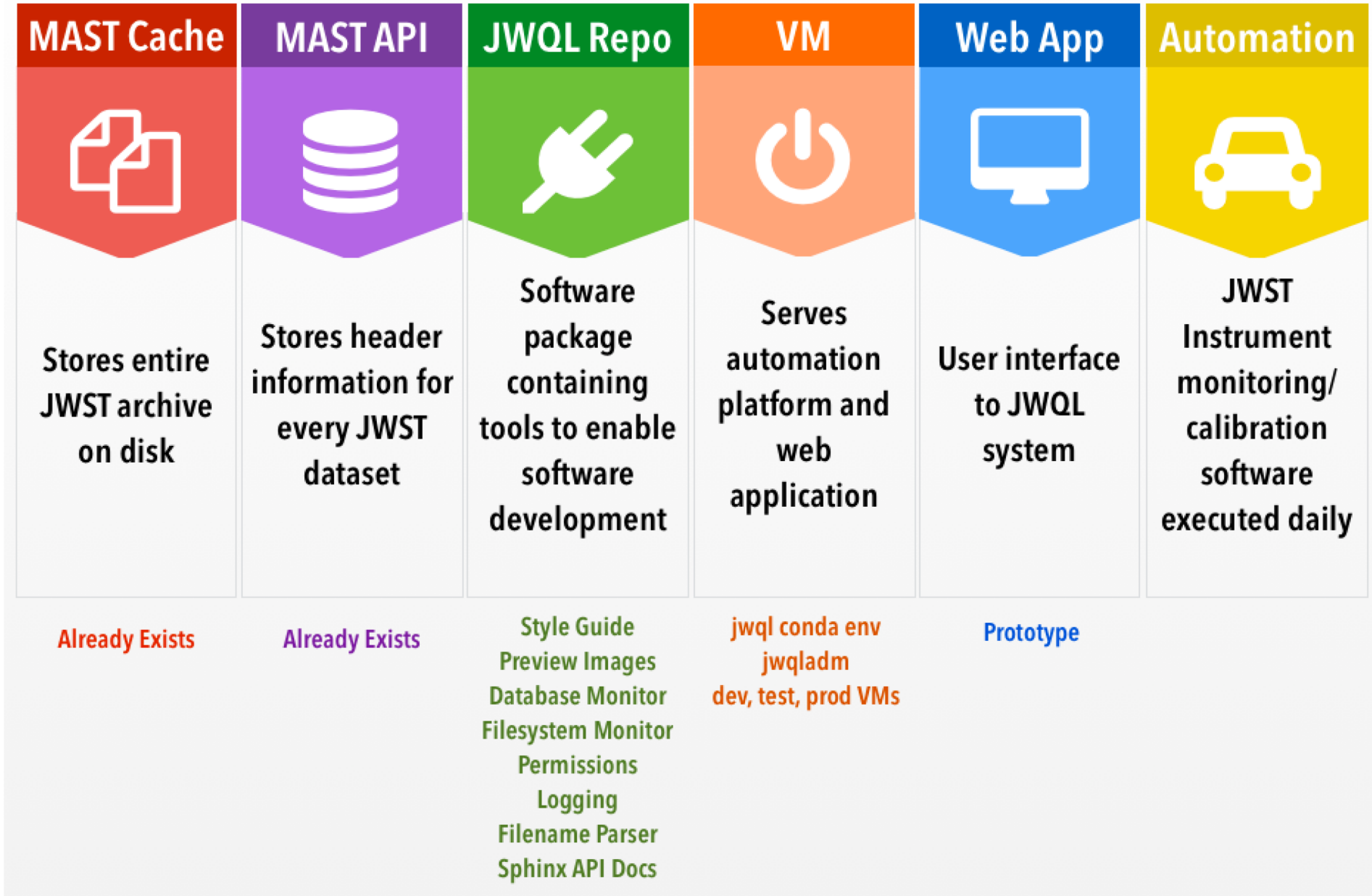
Visually inspect new and archived data

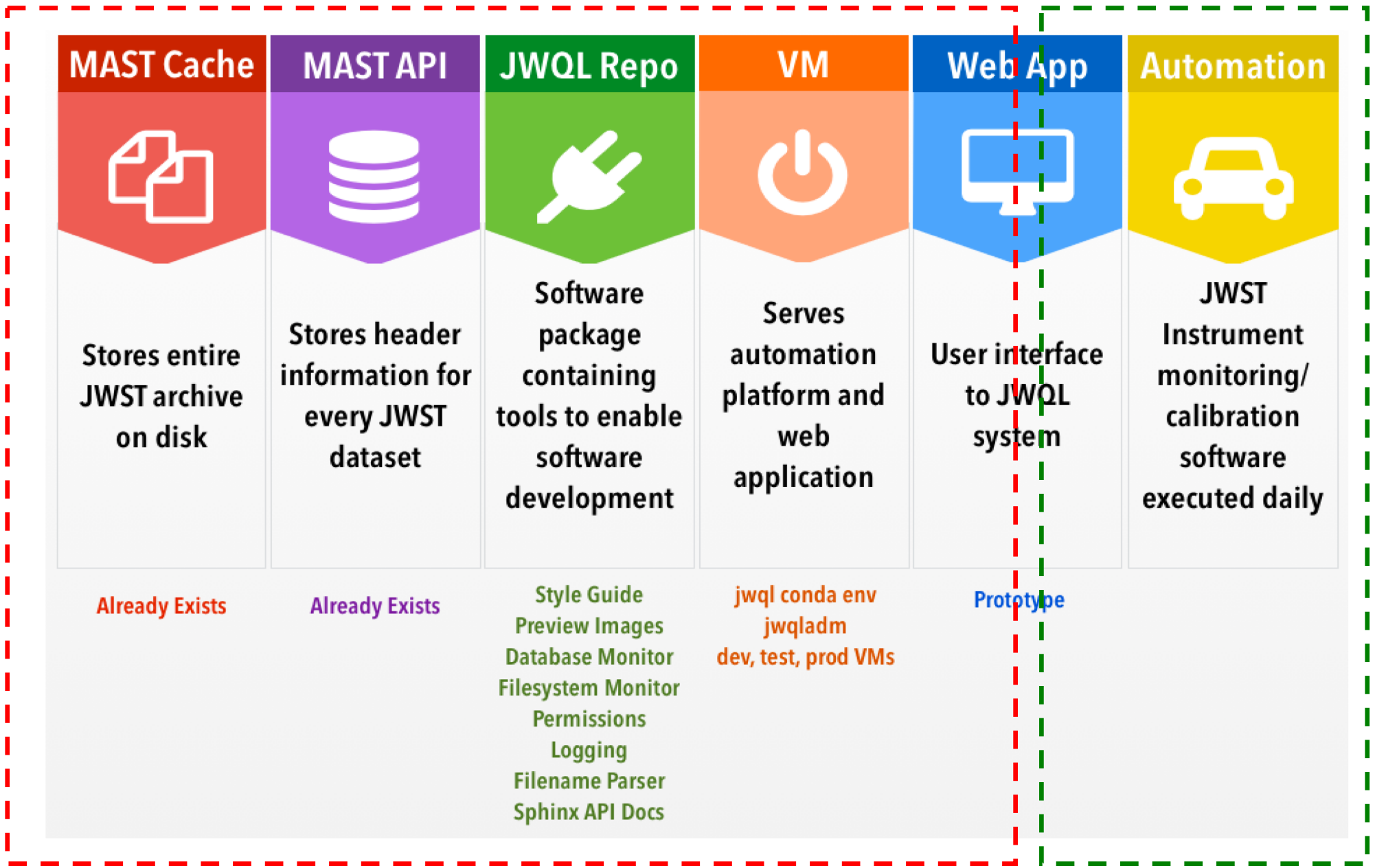
Automated calibration & monitoring software

Convenience of a web browser

One solution for five teams

A database-driven web application and automation framework for analyzing and monitoring JWST instruments and data




Core Infrastructure
Instrument-specific
MAST Cache
MAST API
JWQL Repo
VM
Web App
Automation
Stores entire JWST archive on disk
Stores header information for every JWST dataset
Software package containing tools to enable software development
Serves automation platform and web application
User interface to JWQL system
JWST Instrument monitoring/ calibration software executed daily
Already Exists
Already Exists
Style Guide
Preview Images
Database Monitor
Filesystem Monitor
Permissions
Logging
Filename Parser
Sphinx API Docs
jwql conda env
jwqladm
dev, test, prod VMs
Prototype


```
(astroconda3) [dljwql:jw00318001001] ls *.fits
jw00318001001_02101_00001_nis_cal.fits
jw00318001001_02101_00001_nis_i2d.fits
jw00318001001_02101_00001_nis_original.fits
jw00318001001_02101_00001_nis_rate.fits
jw00318001001_02101_00001_nis_rateints.fits
jw00318001001_02101_00001_nis_trapsfilled.fits
jw00318001001_02101_00001_nis_uncal.fits
jw00318001001_02102_00001_nis_cal.fits
jw00318001001_02102_00001_nis_i2d.fits
jw00318001001_02102_00001_nis_original.fits
jw00318001001_02102_00001_nis_rate.fits
jw00318001001_02102_00001_nis_rateints.fits
jw00318001001_02102_00001_nis_trapsfilled.fits
jw00318001001_02102_00001_nis_uncal.fits
```

MAST API

Stores header information for every JWST dataset

```
from astroquery.mast import Mast
```

```
service = "Mast.Jwst.Filtered.Nircam"
params = {"columns":"filename, instrume, detector, filter,\
                     expstart, apername, duration",
          "filters":[{"paramName":"filename",
                      "values":['jw94015001002_02104_00001_nrca2_cal.fits']
                     ]}
response = Mast.service_request_async(service,params)
result = response[0].json()['data']
```

```
result
```

```
[{'apername': 'NRCA2_FULL',
  'detector': 'NRCA2',
  'duration': 64.42056000000001,
  'expstart': 57410.63391019028,
  'filename': 'jw94015001002_02104_00001_nrca2_cal.fits',
  'filter': 'F115W',
  'instrume': 'NIRCAM'}]
```

We use a standardized, open development workflow using `git`/GitHub that:

- **Enables Collaboration** – Many people contributing to the code base simultaneously
- **Supports Continuous Delivery** – "Production" version of code that is well tested and works
- **Aids in code readability & maintainability**
- **Avoids redundancy** – One solution instead of five
- **Facilitates knowledge transfer** – At least two people looking at every line of code
- **Promotes industry and institute best practices**
- **Conforms to institute security policies**

This is accomplished by:

- Cross-instrument / cross-division development team
- Development standards
- Code Review
- Documentation for onboarding and contributing

Version Control Policy

Source Code Control

Created by Glenn Miller, last modified on Apr 05, 2018

Purpose	Source code control is a best practice for software engineering. Source code control has evolved to embrace the iterative and collaborative models of code development seen across the industry. The products and services we provide to our customer, NASA, are embodied in the software and other products that we develop. Rights to these products are retained by NASA, and all of the Intellectual Property rights are retained by AURA. Therefore, we have an obligation to retain our software and other products in perpetuity. Benefits of this policy are: • Cross-Institute support for best practices • Decreased cost of infrastructure support • Ability to meet contractual requirements to deliver, store, track changes to, and archive our products Please direct questions about this policy to the IT Service Desk, itsupport.stsci.edu. Read more about enforcement or exceptions to policy.

GitHub Policy

GitHub

Created by Glenn Miller, last modified on Feb 23, 2018

Purpose	Producing software (in the form of source code and compiled binaries), is a core part of what we do at STScI. Sometimes this software needs to be shared with collaborators and the wider community. This policy is designed to govern how we make use of the GitHub platform for collaborating on public and private repositories. Please direct questions about this policy to the IT Service Desk, itsupport.stsci.edu. Read more about enforcement or exceptions to policy.

Continuous Integration

Continuous Integration (Draft)

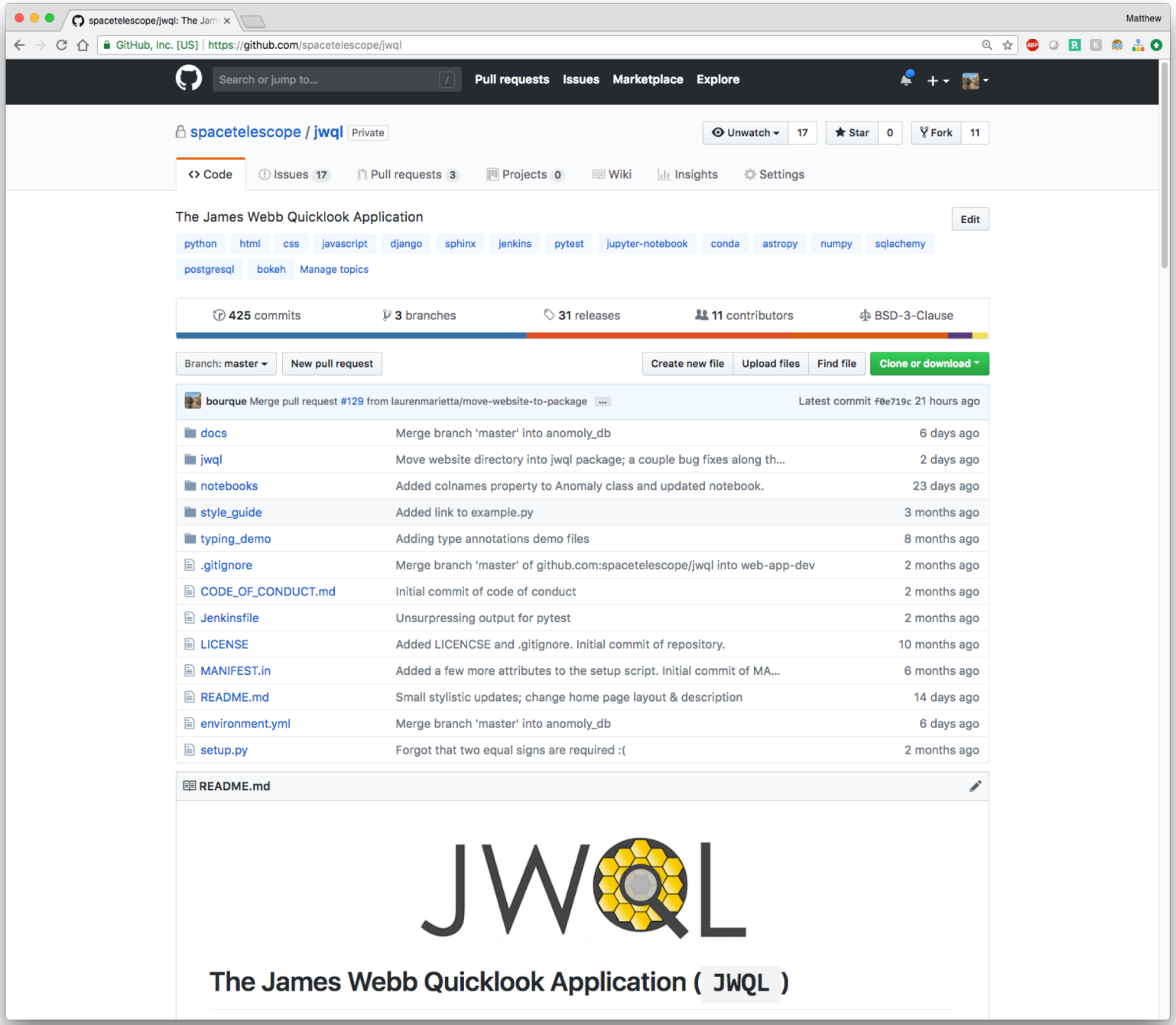
Created by Arfon Smith, last modified by Glenn Miller on Feb 27, 2018

Purpose	Continuous Integration (CI), is a development practice whereby developers check-in code to a shared repository regularly (usually several times per day). Each time this code is checked-in, the automated test suite is executed, thereby allowing teams to detect problems with their codebase early. Continuous Integration, combined with automated test suites and Source Code Control, are the foundations of modern best practice Software Engineering. Our customer, NASA, requires us to produce reliable, high-quality software, delivered on a regular schedule. Implementing this policy can only serve to help us deliver on this core competency of the Institute. Please direct questions about this policy to the IT Service Desk, itsupport.stsci.edu. Read more about enforcement or exceptions to policy.

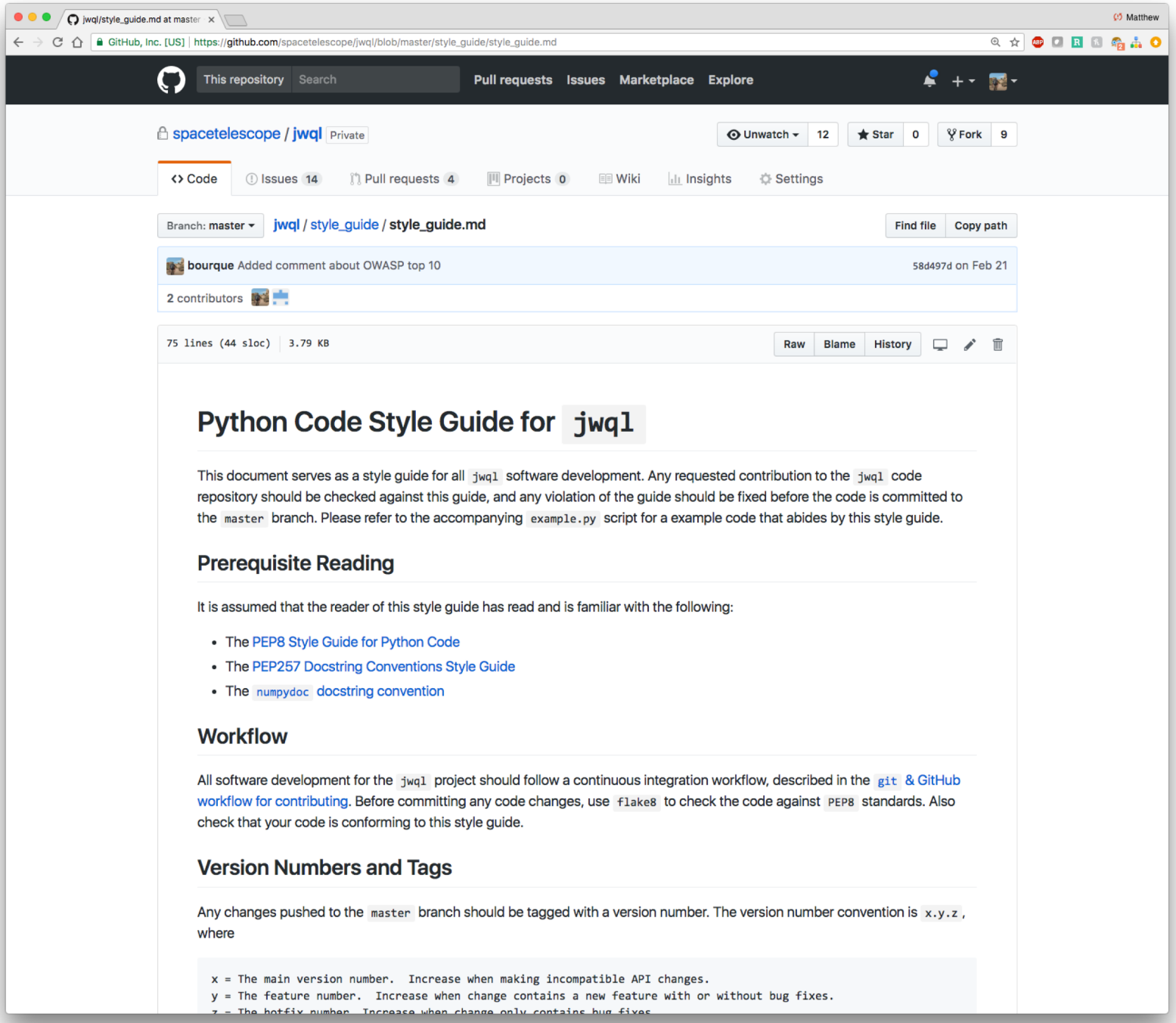
JWQL Repo

Software package containing tools to enable software development

github.com/spacetelescope/jwql



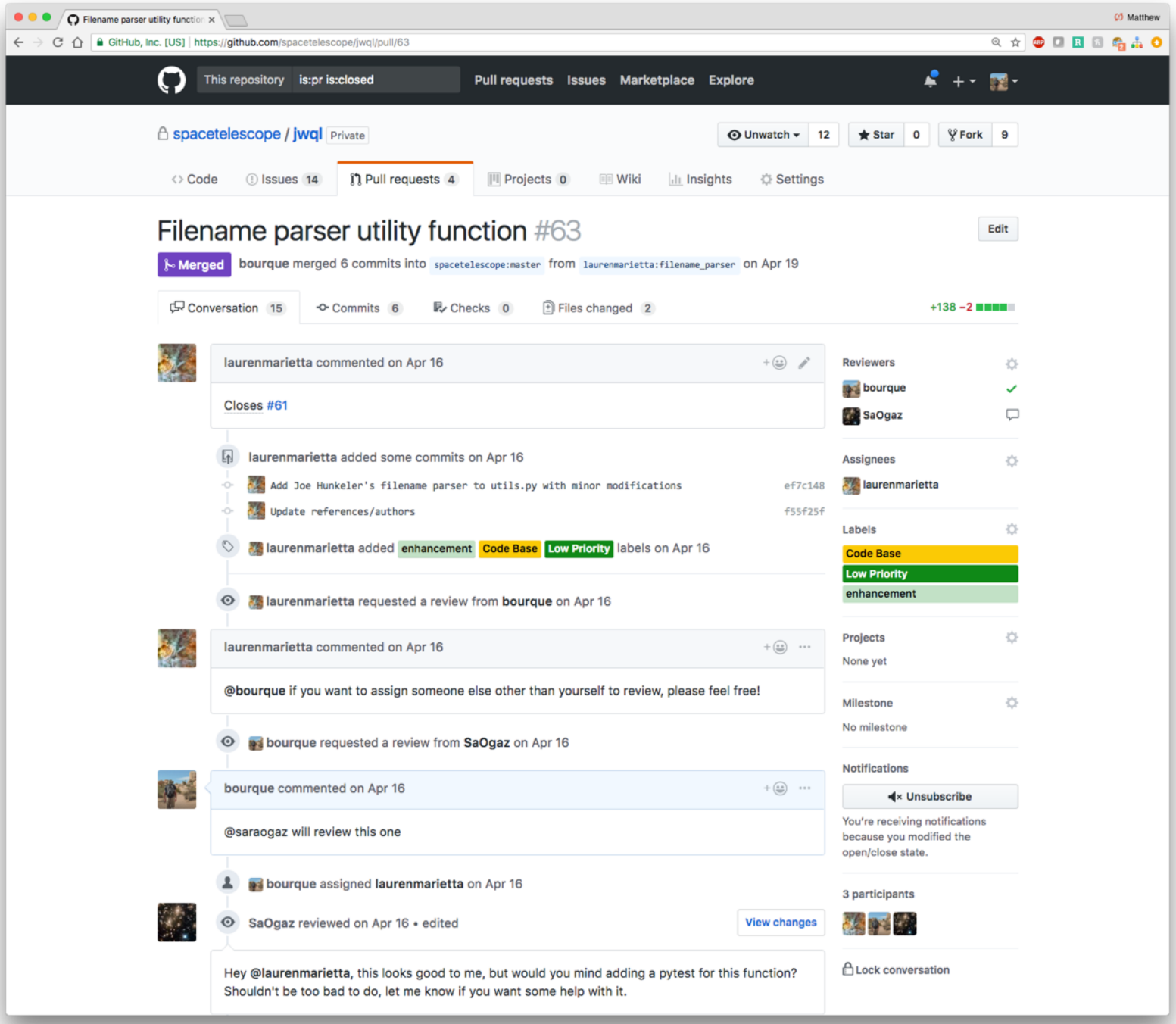
- PEP8 Python Standards
- PEP257 Documentation Standards
- numpydoc API doc conventions



JWQL Repo

Software package containing tools to enable software development

All code is reviewed through Pull Requests



git & GitHub workflow for contributing

GitHub, Inc. [US] | https://github.com/spacetelescope/jwql/wiki/git-&-GitHub-workflow-for-contributing

This repository Search Pull requests Issues Marketplace Explore

spacetelescope / jwql Private

Unwatch 12 Star 0 Fork 9

Code Issues 14 Pull requests 4 Projects 0 Wiki Insights Settings

git & GitHub workflow for contributing

Edit New Page

Matthew Bourque edited this page on Mar 13 · 3 revisions

The best method for contributing software to the `jwql` project is a workflow that involves forking the `jwql` repository, developing changes on personal branches, and opening pull requests through GitHub.

The first question you will have to figure out is whether you should open an issue for this update - if you think that this change will be solving a significant problem or add a significant enhancement to the project then it would be advantageous to open an issue ticket here. This will allow both individuals and the team as a whole to keep track of the project and our progress as we go. Any appropriate individuals should be assigned to the issue, and a label(s) should be tagged.

Following that, any changes that you want to eventually make to the `master` branch should be done through the workflow where you create a fork and work on your own branch before submitting those changes to be reviewed through a pull request. Instructions on how to do those things can be found below:

1. Create a personal fork of the `jwql` repository by visiting the `spacetelescope` repository and clicking the `Fork` button. Note that this only has to be done once.
2. Make a local copy of your personal fork by cloning the repository (e.g. `git clone https://github.com/username/jwql.git`, found by clicking the green "clone or download" button.). Note that, unless you explicitly delete your clone of the fork, this only has to be done once.
3. Ensure that the personal fork is pointing to the `upstream` `spacetelescope/jwql` repository with `git remote add upstream https://github.com/spacetelescope/jwql.git`. Note that this only has to be done once.
4. Create a branch on the cloned personal fork to develop software changes on. Branch names should be short but descriptive (e.g. `new-database-table` or `fix-ingest-algorithm`) and not too generic (e.g. `bug-fix`). Also consistent use of hyphens is encouraged.
    i. `git branch <branchname>` - you only need to do this when you first create your branch.
    ii. `git checkout <branchname>` - you can use this command to switch back and forth between existing branches.
    iii. Perform local software changes using the nominal `git add` / `git commit -m` cycle.
        a. `git status` - allows you to see which files have changed.

Pages 8

- Home
- Config file
- Environment Installation
- Feature Requests
- git & GitHub workflow for contributing
- JWQL Logging Function: The Why's and The How
- Useful Links
- Web Application

Add a custom sidebar

Clone this wiki locally

https://github.com/spacetel

Clone in Desktop

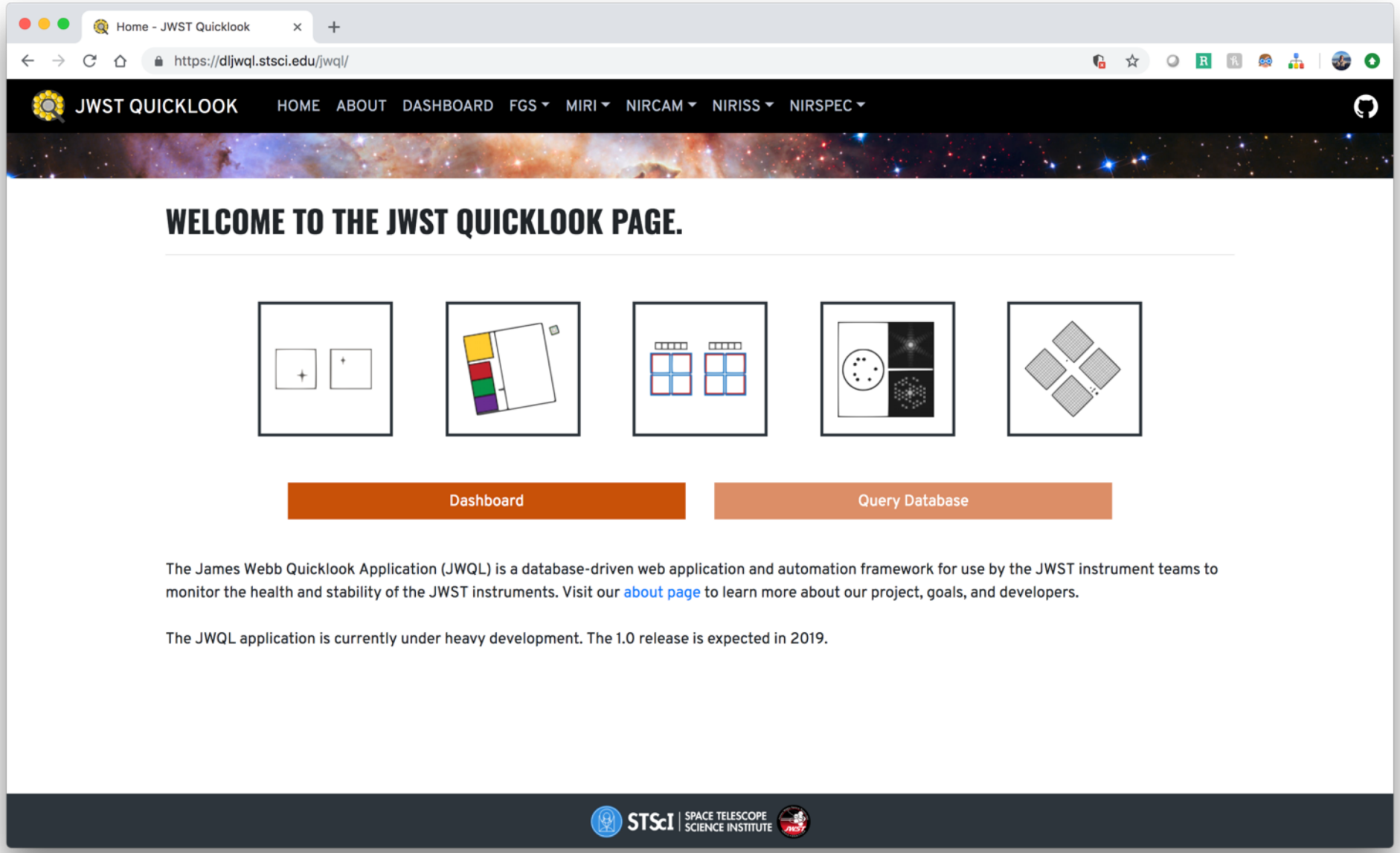
Core Infrastructure Development Progress

Provides tools necessary for future instrument-specific monitoring/calibration scripts

Module/Tool Name	Description
`utils.py`	Various utility functions (e.g. filename parser, config file parser)
`permissions.py`	UNIX permission management for JWQL-related files
`preview_image.py`	Creates preview images for JWST FITS files for web application
`logging_functions.py`	Logs execution of scripts
`monitor_mast.py`	Monitor the contents of the MAST database
`filesystem_monitor.py`	Monitor the contents of the MAST data cache
`monitor_template.py`	Provides examples of how to build an instrument monitor
`jwql_webapp`	`Django` web application
`pytest` unit tests	Tests for individual pieces of code
Jenkins Continuous Integration	Automatically runs tests when new code is introduced to repo
`sphinx` API Documentation	Automatically generates code documentation

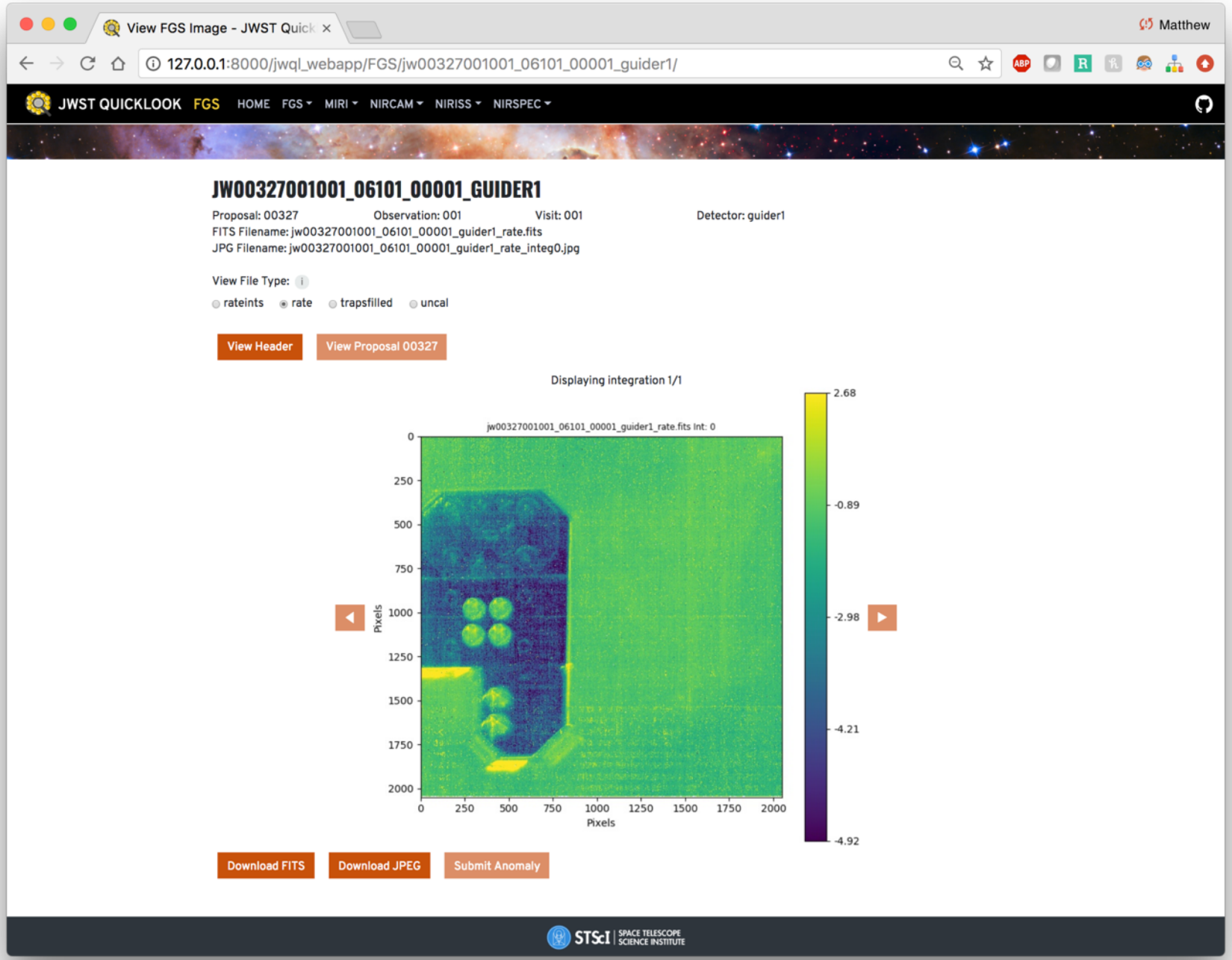
Web App

User interface to JWQL system



Web App

User interface to JWQL system



Bias Monitor
Readnoise Monitor
Gain Level Monitor
Dark Current Monitor
Hot Pixel Monitor
Photometric Stability Monitor
Dead/Bad Pixel Population Monitor
Flat Field Monitor
Interpixel Capacitance/Crosstalk Monitor
Snowball Monitor
Wide Field Slitless Dispersion Calibration Monitor
Pedestal Monitor
1/f Noise Monitor
Average Closure Phase of AMI Calibrators Monitor
RMS for the derivative of TSO Observation Monitor
Engineering Database Mnemonics
Sky/telescope Background Monitor
PSF Characterization
Photometric Calibration
Internal Flats Monitor
Target Acquisition Offsets/Failure Monitor
Fringe Phases/Amplitudes AMI observation Monitor
WCS Outputs Monitor

Optical Short Monitor
Saturation Level Monitor
Reference Pixel Monitor
Internal Lamp Monitor
Instrument Model Updates
Failed-open Shutter Monitor
MSA Contrast
Detector Monitoring via Fixed-slit Subarrays
Readout Transition Monitor
BOTS Monitoring
Line Spread Function Monitor
IFU Leakage Characterization
Partial Calibration
Cosmic Ray Monitoring
Blind Pointing Accuracy
Filter and Calibration Lamp Monitor
Telescope Thermal Emission Monitoring
MRS Straylight Monitor
MRS Astrometric Monitor
Geometric Distortion Monitor
FGS-SI Alignment I Monitor
FGS-SI Alignment II Monitor
Focus Monitor

List of monitors can be viewed/updated at:

https://confluence.stsci.edu/display/JWQLPROJ/Instrument+Monitor+Overview

JWQL

jwql@stsci.edu

https://github.com/spacetelescope/jwql

JWQLPROJ