STScI | SPACE TELESCOPE SCIENCE INSTITUTE

EXPANDING THE FRONTIERS OF SPACE ASTRONOMY

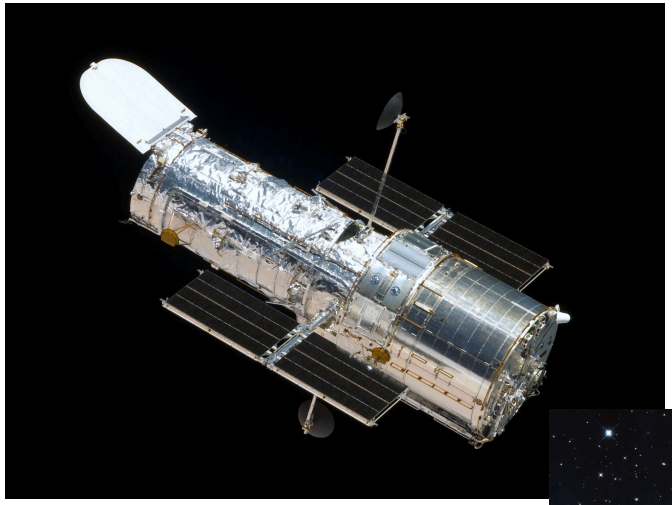
The Hubble Space Telescope (HST) Advanced Camera for Surveys (ACS) Quicklook Application

Matthew Bourque

October 19, 2017

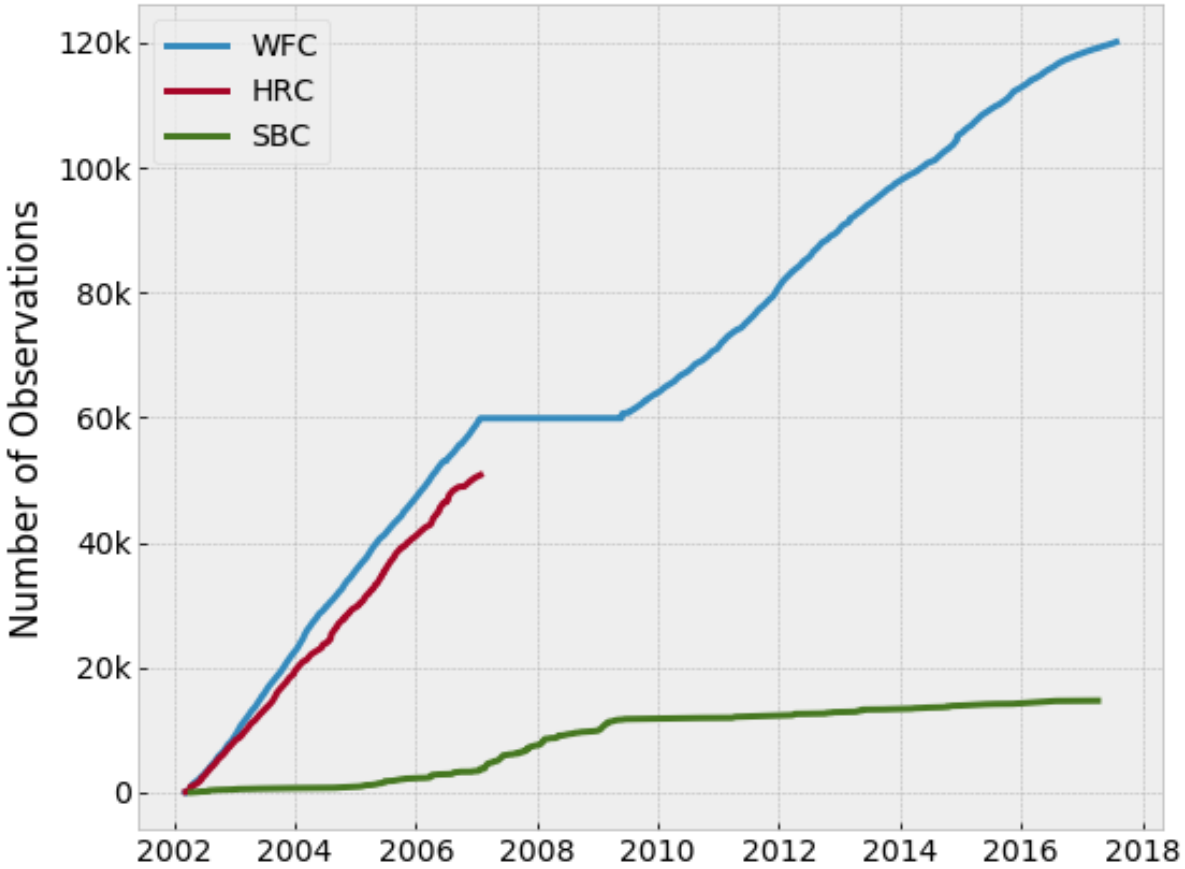
The Hubble Space Telescope (HST)

- HST was launched in 1990
- Over the years it has housed 12 instruments
- 5 Servicing Missions
- Currently there are 4 instruments on board:
  - Cosmic Origins Spectrograph (COS) (2009)
  - Space Telescope Imaging Spectrograph (STIS) (1990)
  - Wide Field Camera 3 (WFC3) (2009)
  - Advanced Camera for Surveys (ACS) (2002)



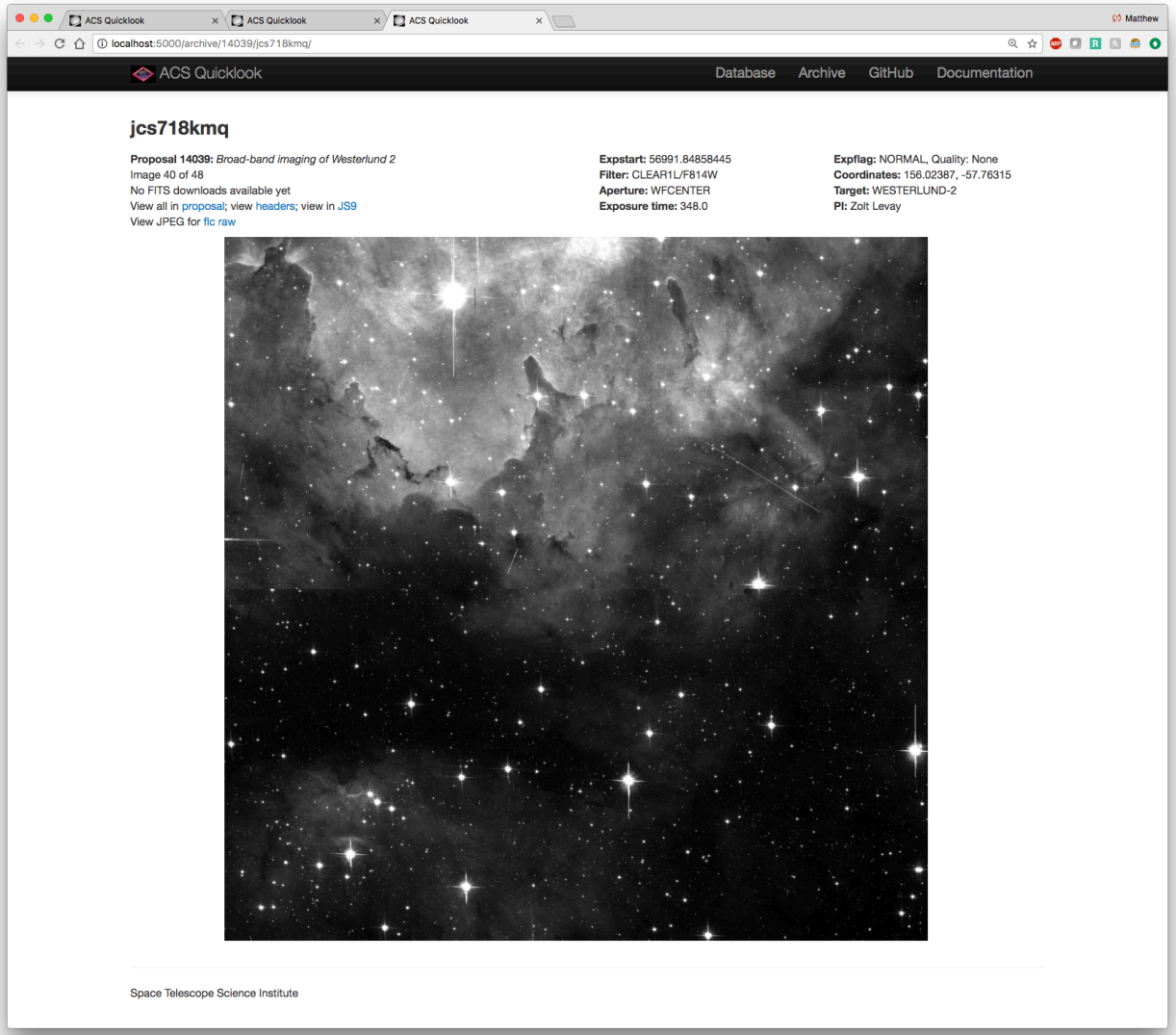
The Advanced Camera for Surveys (ACS)

- Comprised of three detectors:
  - Wide Field Channel (WFC)
  - High Resolution Channel (HRC)
  - Solar Blind Channel (SBC)
- ACS experienced an electronics failure in 2006, temporarily disabling the WFC detector and permanently disabling the HRC detector.



The HST/ACS Quicklook Application (`acsql`)

- Database-driven web application for discovering and viewing HST/ACS images and corresponding metadata
- All publicly-available on-orbit data
- Comprised of four major components:
  - NFS Filesystem
  - MySQL Database
  - Python/Flask-based Web Application
  - Software Package
- Inspired by a similar application for the HST/WFC3 instrument (“WFC3 Quicklook”)



Data Structure: FITS Files

FITS = Flexible Image Transport System

Extension	Purpose	Image Dimensions (pixels)	Data Type
0	Primary header	–	String
1	SCI, Chip 2	(4096, 2048)	Float
2	ERR, Chip 2	(4096, 2048)	Float
3	DQ, Chip 2	(4096, 2048)	Integer
4	SCI, Chip 1	(4096, 2048)	Float
5	ERR, Chip 1	(4096, 2048)	Float
6	DQ, Chip 1	(4096, 2048)	Integer

ACS/WFC FITS file structure

Extension	Purpose	Image Dimensions (pixels)	Data Type
0	Primary header	–	String
1	SCI	(1024, 1024)	Float
2	ERR	(1024, 1024)	Float
3	DQ	(1024, 1024)	Integer

ACS/WFC and ACS/SBC FITS file structure

Data Structure: FITS file headers and key metadata

```
SIMPLE  =                    T / data conform to FITS standard
BITPIX  =                   16 / bits per data value
NAXIS   =                    0 / number of data axes
EXTEND  =                    T / File may contain standard extensions
NEXTEND =                    6 / Number of standard extensions
GROUPS  =                    F / image is in group format
DATE    = '2016-09-22'         / date this file was written (yyyy-mm-dd)
FILENAME= 'j6mf16lhq_raw.fits                        ' / name of file
FILETYPE= 'SCI      '          / type of data found in data file

TELESCOP= 'HST'                / telescope used to acquire data
INSTRUME= 'ACS   '             / identifier for instrument used to acquire data
EQUINOX =               2000.0 / equinox of celestial coord. system

             / DATA DESCRIPTION KEYWORDS

ROOTNAME= 'j6mf16lhq                           ' / rootname of the observation set
IMAGETYP= 'DARK              ' / type of exposure identifier
PRIMESI = 'ACS   '             / instrument designated as prime

             / TARGET INFORMATION

TARGNAME= 'DARK                            ' / proposer's target name
RA_TARG =   0.000000000000E+00 / right ascension of the target (deg) (J2000)
DEC_TARG=   0.000000000000E+00 / declination of the target (deg) (J2000)

             / PROPOSAL INFORMATION

PROPOSID=                 9433 / PEP proposal identifier
LINENUM = '16.055        '     / proposal logsheet line number
PR_INV_L= 'Bernstein                       ' / last name of principal investigator
PR_INV_F= 'Gary                ' / first name of principal investigator
PR_INV_M= '                    ' / middle name / initial of principal investigat

             / EXPOSURE INFORMATION

SUNANGLE=            93.563698 / angle between sun and V1 axis
MOONANGL=            33.222004 / angle between moon and V1 axis
SUN_ALT =            68.062172 / altitude of the sun above Earth's limb
FGSLOCK = 'FINE                ' / commanded FGS lock (FINE,COARSE,GYROS,UNKNOWN)
GYROMODE= '3'                  / number of gyros scheduled, T=3+OBAD
REFFRAME= 'GSC1    '           / guide star catalog version
MTFLAG  = ' '                  / moving target flag; T if it is a moving target

DATE-OBS= '2003-01-27'         / UT date of start of observation (yyyy-mm-dd)
TIME-OBS= '15:20:01'           / UT time of start of observation (hh:mm:ss)
EXPSTART=   5.266663890058E+04 / exposure start time (Modified Julian Date)
EXPEND  =   5.266665048715E+04 / exposure end time (Modified Julian Date)
EXPTIME =          1000.000000 / exposure duration (seconds)--calculated
```

`APERTURE` – Portion of detector used during observation
`DATE-OBS` – The date of the start of the observation
`DEC_TARG` – The declination of target in the night sky
`DETECTOR` – The detector used (WFC, HRC, or SBC)
`EXPSTART` – The exposure start time
`EXPFLAG` – The quality of the exposure
`EXPTIME` – The duration of the exposure
`FILTER` – The wavelength filter used during observation
`IMAGTYP` – The type of image (e.g. “EXTERNAL”)
`OBSTYPE` – The type of observation (e.g. “IMAGING”)
`proposal_type` – The type of proposal (e.g. “CAL”)
`PROPID` – The 5-digit proposal ID
`RA_TARG` – The right-ascension of the target in the night sky
`rootname` – The 9-character unique identifier for the observation
`SUBARRAY` – Boolean value for if an image was a full-frame or not
`TARGNAME` – The name of the observation target (e.g. “Saturn”)
`TIMEOBS` – The time of the start of the observation

Data Structure: ACS FITS Filetypes and Rootnames

`<rootname>_<filetype>.fits`

Rootnames consist of a 9-character identifier corresponding to a unique observation

Available filetypes:

`raw` - the raw, uncalibrated data that comes directly from HST
`flt` - nominally calibrated data
`flc` - nominally calibrated data corrected for Charge Transfer Efficiency (CTE) deficits.
`drz` - geometric distortion-corrected data
`drc` - geometric distortion-corrected and CTE corrected data
`spt` - telescope telemetry data
`jit` - telescope pointing data
`jif` - telescope drifting data
`crj` - cosmic ray rejected data
`crc` - cosmic ray rejected plus CTE corrected data
`asn` - observation association table.

Intended Users and Use Cases

- The primary user of the **`acsql`** application are ACS Instrument Analysts at STScI
- They use ACS data to perform instrument calibration and monitoring
- External users to STScI are possible since **`acsql`** is non-proprietary

Use case 1: Visually inspect an image from the ACS archive

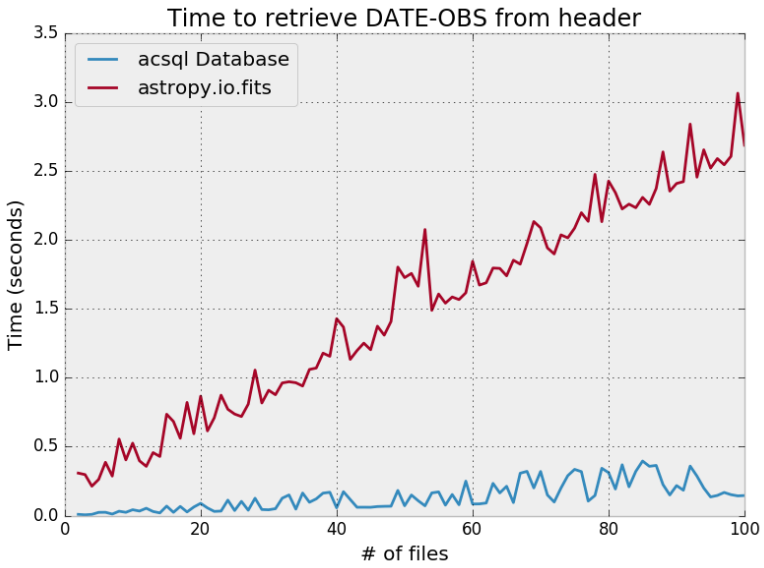
Use case 2: Determine which datasets exist in the ACS archive for a given set of observational parameters

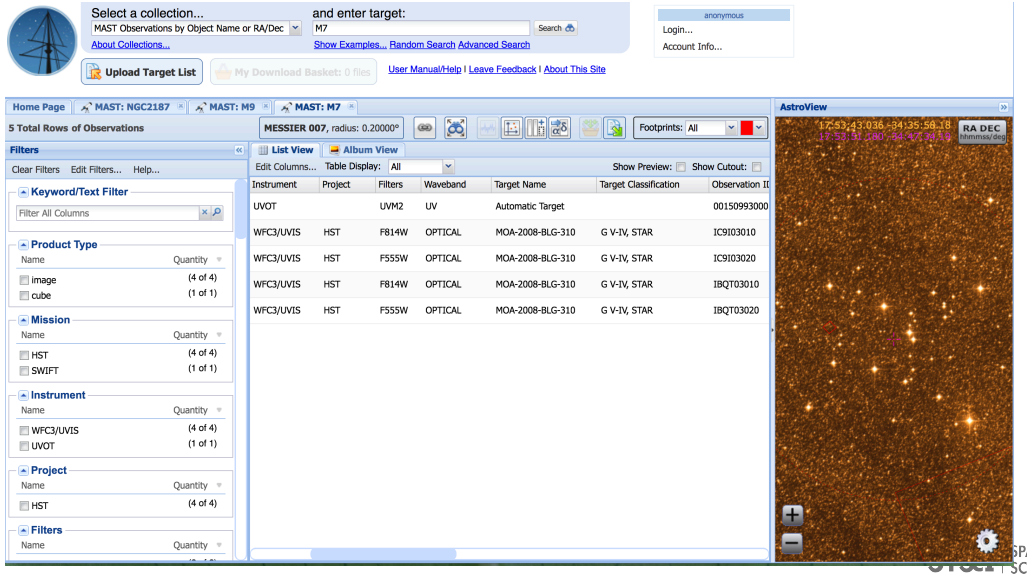
Use case 3: Determine the observational parameters for a given dataset

Use case 4: Programmatically analyze images across custom datasets

Motivation

- Limitations in the FITS file structure (namely File I/O)
- Limitations in the current STScI data archive
  - Data retrieval latency
  - Data discovery
  - Data redundancy
- **`acsql`** is a centrally-located instantly-accessible data archive with one-to-one correspondence with the ACS FITS file headers
  - No complicated data retrieval process
  - No need for file I/O
  - Intuitive database queries





Implementation: Version Control, Programming Standards, and API Documentation

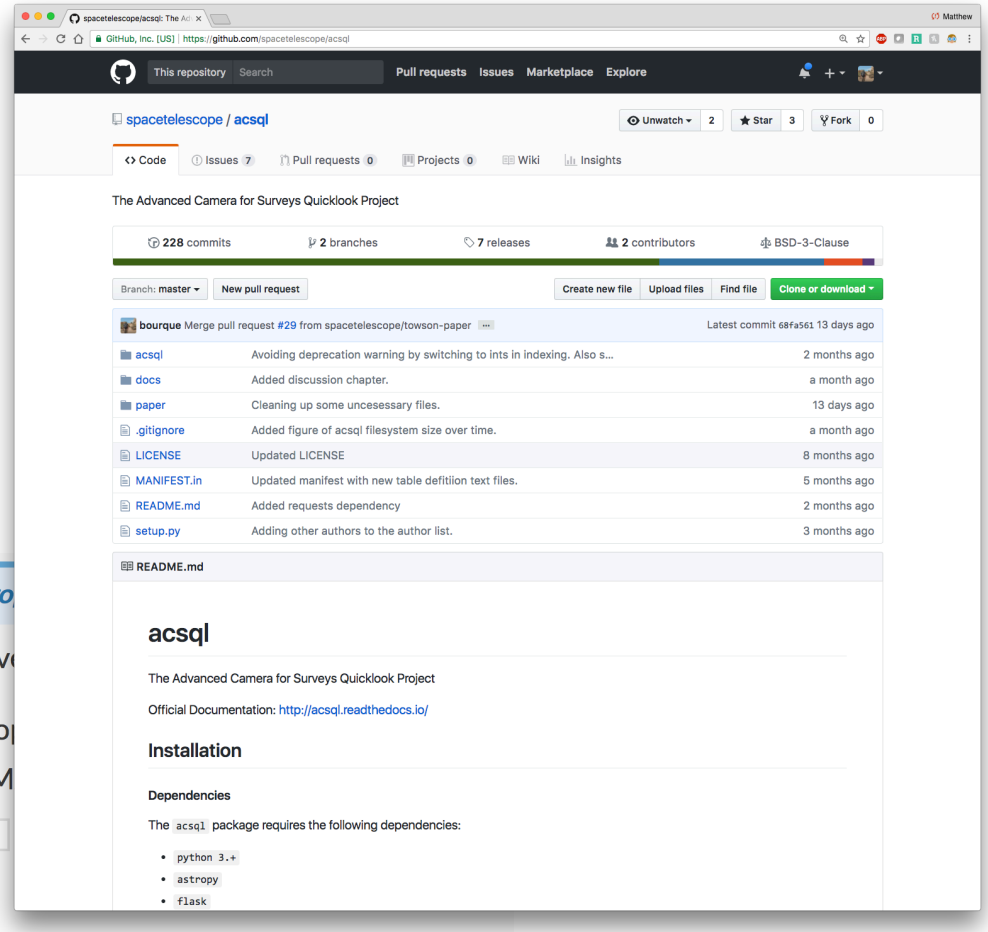
- Style Guide for Python (PEP8)
- Style Guide for Python Docstrings (PEP257)
- `numpydoc` Documentation Convention
- `git` + `GitHub`
- `readthedocs`

```
def get_proposal_type(proposid):
    """Return the ``proposal_type`` for the given ``proposid``.

    The ``proposal_type`` is the type of proposal (e.g. ``CAL``,
    ``GO``, etc.).  The ``proposal_type`` is scraped from the MAST
    proposal status webpage for the given ``proposid``.  If the
    ``proposal_type`` cannot be determined, a ``None`` value is returned.

    Parameters
    ----------
    proposid : str
        The proposal ID (e.g. ``12345``).

    Returns
    -------
    proposal_type : int or None
        The proposal type (e.g. ``CAL``).
    """
```



ingest.ingest.get_proposal_type(*pro*)

Return the `proposal_type` for the giv

The `proposal_type` is the type of pro

`proposal_type` is scraped from the M

given `proposid`. If the `proposal_type`

returned.

Parameters: **proposid** : str

The proposal ID (e.g. `12345`).

Returns: **proposal_type** : int or None

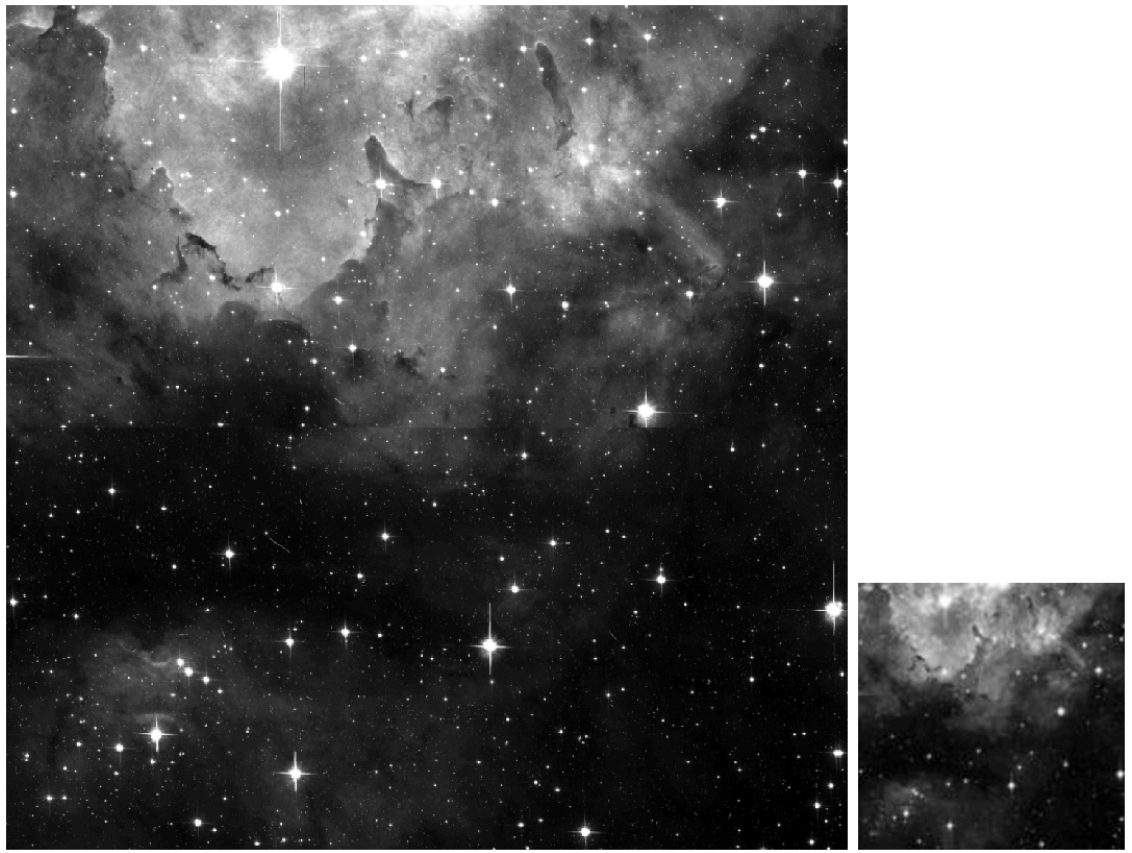
The proposal type (e.g. `CAL`).

Implementation: Filesystem

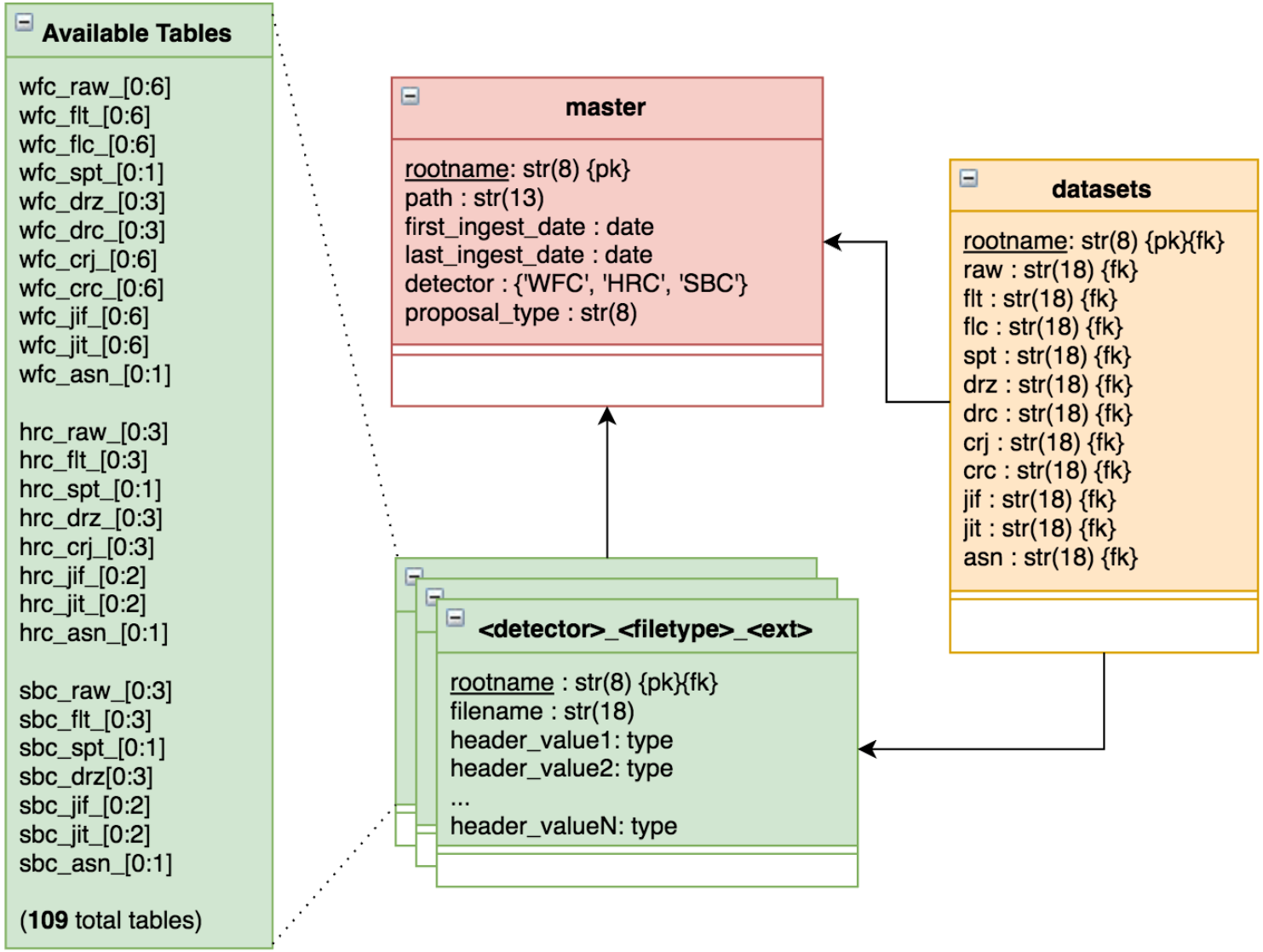
```
filesystem/
    jcp0/
        jcp001kwq/
            jcp001kwq_flc.fits
            jcp001kwq_flt.fits
            jcp001kwq_raw.fits
            jcp001kwq_spt.fits
            jcp001kwq_jif.fits
            jcp001kwq_jit.fits
        jcp001010/
            jcp001010_asn.fits
            jcp001010_drc.fits
            jcp001010_drz.fits
            jcp001010_jif.fits
            jcp001010_jit.fits
        jcp014tyq/
            ...
        jcp001kt1/
            ...
        ...
    jcp3/
        ...
    jcp7/
        ...
    ...
```

```
jpegs/
    12780/
        jbw901jtq_flc.jpg
        jbw901jtq_flt.jpg
        jbw901jtq_raw.jpg
        jbw901jxq_flc.jpg
        jbw901jxq_flt.jpg
        jbw901jxq_raw.jpg
        ...
    12781/
        jbx101f5q_flc.jpg
        jbx101f5q_flt.jpg
        jbx101f5q_raw.jpg
        jbx101f7q_flc.jpg
        jbx101f7q_flt.jpg
        jbx101f7q_raw.jpg
        ...
    12783/
        ...
    12787/
        ...
    ...
```

```
thumbnails/
    12780/
        jbw901jtq_flt.thumb
        jbw901jxq_flt.thumb
        ...
    12781/
        jbx101f5q_flt.thumb
        jbx101f7q_flt.thumb
        ...
    12783/
        ...
    12787/
        ...
    ...
```



Implementation: Database Schema



Implementation: MySQL and SQLAlchemy

- **SQLAlchemy**: Python **SQL** toolkit and Object Relational Mapper

```python
class Master(base):
    """ORM for the  master table."""
    def __init__(self, data_dict):
        self.__dict__.update(data_dict)

    __tablename__ = 'master'
    rootname = Column(String(8), primary_key=True, index=True, nullable=False)
    path = Column(String(15), unique=True, nullable=False)
    first_ingest_date = Column(Date, nullable=False)
    last_ingest_date = Column(Date, nullable=False)
    detector = Column(Enum('WFC', 'HRC', 'SBC'), nullable=False)
    proposal_type = Column(Enum('CAL/ACS', 'CAL/OTA', 'CAL/STIS', 'CAL/WFC3',
                                'ENG/ACS', 'GO', 'GO/DD', 'GO/PAR', 'GTO/ACS',
                                'GTO/COS', 'NASA', 'SM3/ACS', 'SM3/ERO',
                                'SM4/ACS', 'SM4/COS', 'SM4/ERO', 'SNAP'),
                           nullable=True)
```

```python
def load_connection(connection_string):
    """Return ``session``, ``base``, and ``engine`` objects for
    connecting to the ``acsql`` database.

    Create an ``engine`` using an given ``connection_string``. Create a
    ``base`` class and ``session`` class from the ``engine``. Create an
    instance of the ``session`` class. Return the ``session``,
    ``base``, and ``engine`` instances.

    Parameters
    ----------
    connection_string : str
        The connection string to connect to the ``acsql`` database. The
        connection string should take the form:
        ``dialect+driver://username:password@host:port/database``

    Returns
    -------
    session : sesson object
        Provides a holding zone for all objects loaded or associated
        with the database.
    base : base object
        Provides a base class for declarative class definitions.
    engine : engine object
        Provides a source of database connectivity and behavior.
    """

    engine = create_engine(connection_string, echo=False, pool_timeout=100000)
    base = declarative_base(engine)
    Session = sessionmaker(bind=engine)
    session = Session()

    return session, base, engine
```

Implementation: MySQL and SQLAlchemy

```
def orm_factory(class_name):
    """Create a SQLAlchemy ORM Classes with the given ``class_name``.

    Parameters
    ----------
    class_name : str
        The name of the class to be created

    Returns
    -------
    class : obj
        The SQLAlchemy ORM
    """

    data_dict = {}
    data_dict['__tablename__'] = class_name.lower()
    data_dict['rootname'] = Column(String(8), ForeignKey('master.rootname'),
                                   primary_key=True, index=True,
                                   nullable=False)
    data_dict['filename'] = Column(String(18), nullable=False, unique=True)
    data_dict = define_columns(data_dict, class_name)
    data_dict['__table_args__'] = {'mysql_row_format': 'DYNAMIC'}

    return type(class_name.upper(), (base,), data_dict)
```

```
WFC_raw_0 = orm_factory('WFC_raw_0')
WFC_raw_1 = orm_factory('WFC_raw_1')
WFC_raw_2 = orm_factory('WFC_raw_2')
WFC_raw_3 = orm_factory('WFC_raw_3')
WFC_raw_4 = orm_factory('WFC_raw_4')
WFC_raw_5 = orm_factory('WFC_raw_5')
WFC_raw_6 = orm_factory('WFC_raw_6')

WFC_flt_0 = orm_factory('WFC_flt_0')
WFC_flt_1 = orm_factory('WFC_flt_1')
WFC_flt_2 = orm_factory('WFC_flt_2')
WFC_flt_3 = orm_factory('WFC_flt_3')
WFC_flt_4 = orm_factory('WFC_flt_4')
WFC_flt_5 = orm_factory('WFC_flt_5')
WFC_flt_6 = orm_factory('WFC_flt_6')

WFC_flc_0 = orm_factory('WFC_flc_0')
WFC_flc_1 = orm_factory('WFC_flc_1')
WFC_flc_2 = orm_factory('WFC_flc_2')
WFC_flc_3 = orm_factory('WFC_flc_3')
WFC_flc_4 = orm_factory('WFC_flc_4')
WFC_flc_5 = orm_factory('WFC_flc_5')
WFC_flc_6 = orm_factory('WFC_flc_6')
```

Implementation: Data Ingestion Software Algorithm

1. Identify newly available data
2. Loop over **`rootnames`** in parallel
    3. Update **`master`** table with information about **`rootname`**
    4. Loop over available filetypes for given **`rootname`**
        5. Create Python dictionary with metadata about file
        6. For each FITS extension, extract header information and update **`header`** table
        7. Update **`datasets`** table
        8. If filetype is **`raw`**, **`flt`**, or **`flc`**, then create JPEG
        9. If filetype is **`flt`**, then create Thumbnail

Encapsulated within **`ingest.py`**, **`make_file_dict.py`**, **`make_jpeg.py`**, **`make_thumbnail.py`**

Implementation: Logging

```
08/15/2017 11:05:26 INFO: User: bourque
08/15/2017 11:05:26 INFO: System: plhstins1.stsci.edu
08/15/2017 11:05:26 INFO: Python Version: 3.5.2
08/15/2017 11:05:26 INFO: Python Path: /envs/anaconda3/envs/astroconda3/bin/python
08/15/2017 11:05:26 INFO: Numpy Version: 1.11.2
08/15/2017 11:05:26 INFO: Numpy Path: /site-packages/numpy
08/15/2017 11:05:26 INFO: Astropy Version: 1.2.1
08/15/2017 11:05:26 INFO: Astropy Path: /site-packages/astropy
08/15/2017 11:05:26 INFO: SQLAlchemy Version: 1.1.4
08/15/2017 11:05:26 INFO: SQLAlchemy Path: /site-packages/SQLAlchemy-1.1.4-py3.5-linux-x86_64.egg/sqlalchemy
08/15/2017 11:05:26 INFO: Gathering files to ingest
08/15/2017 11:06:00 INFO: j8zh21xv: Updated master table.
08/15/2017 11:06:01 INFO: j8zh21xv: Updated HRC_flt_0 table.
08/15/2017 11:06:01 INFO: j8zh21xv: Updated HRC_flt_1 table.
08/15/2017 11:06:01 INFO: j8zh21xv: Updated HRC_flt_2 table.
08/15/2017 11:06:01 INFO: j8zh21xv: Updated HRC_flt_3 table.
08/15/2017 11:06:01 INFO: j8zh21xv: Updated datasets table for flt.
08/15/2017 11:06:01 INFO: j8zh21xv: Creating JPEG
08/15/2017 11:06:02 INFO: j8zh21xv: Creating Thumbnail
08/15/2017 11:06:02 INFO: j8zh21xv: Updated HRC_raw_0 table.
08/15/2017 11:06:02 INFO: j8zh21xv: Updated HRC_raw_1 table.
08/15/2017 11:06:03 INFO: j8zh21xv: Updated HRC_raw_2 table.
08/15/2017 11:06:03 INFO: j8zh21xv: Updated HRC_raw_3 table.
08/15/2017 11:06:03 INFO: j8zh21xv: Updated datasets table for raw.
08/15/2017 11:06:03 INFO: j8zh21xv: Creating JPEG
08/15/2017 11:06:04 INFO: j8zh21xv: Updated HRC_spt_0 table.
08/15/2017 11:06:05 INFO: j8zh21xv: Updated HRC_spt_1 table.
08/15/2017 11:06:05 INFO: j8zh21xv: Updated datasets table for spt.
08/15/2017 11:06:05 INFO: j8zh21xv: End ingestion
```

Implementation: Web Application

Built using Flask, a Python-based web framework

`acsql_webapp.py` — Runs web application on **`localhost`** or server

`data_containers.py` — Various functions for gathering and returning data used by application

`form_options.py` — Stores data needed for database query form

`query_form.py` — Builds and validates database queries through web application

`query_lib.py` — Contains tools for executing database queries through web application

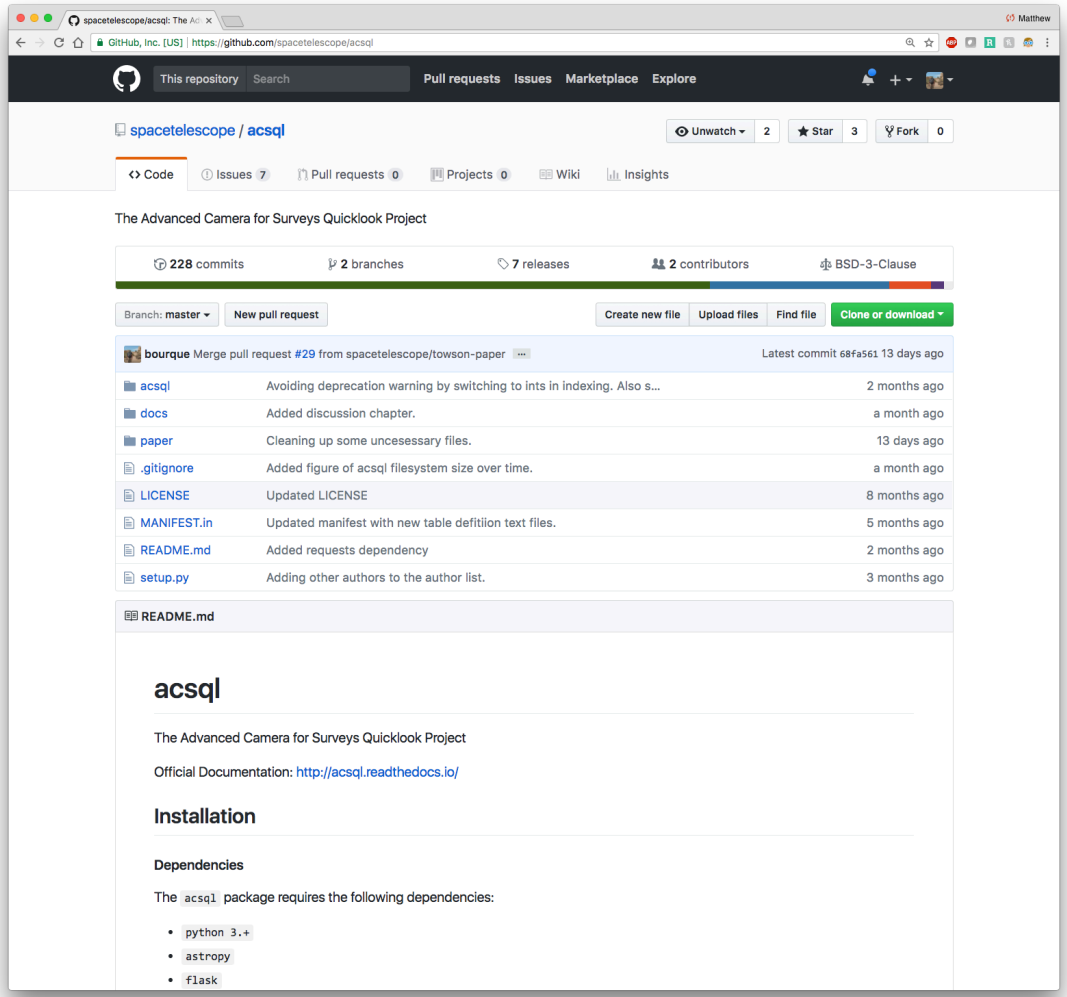
`static/css/*.css` — **`CSS`** templates

`static/js/*.js` — **`JavaScript`** functions

`templates/*.html` — **`HTML`** templates

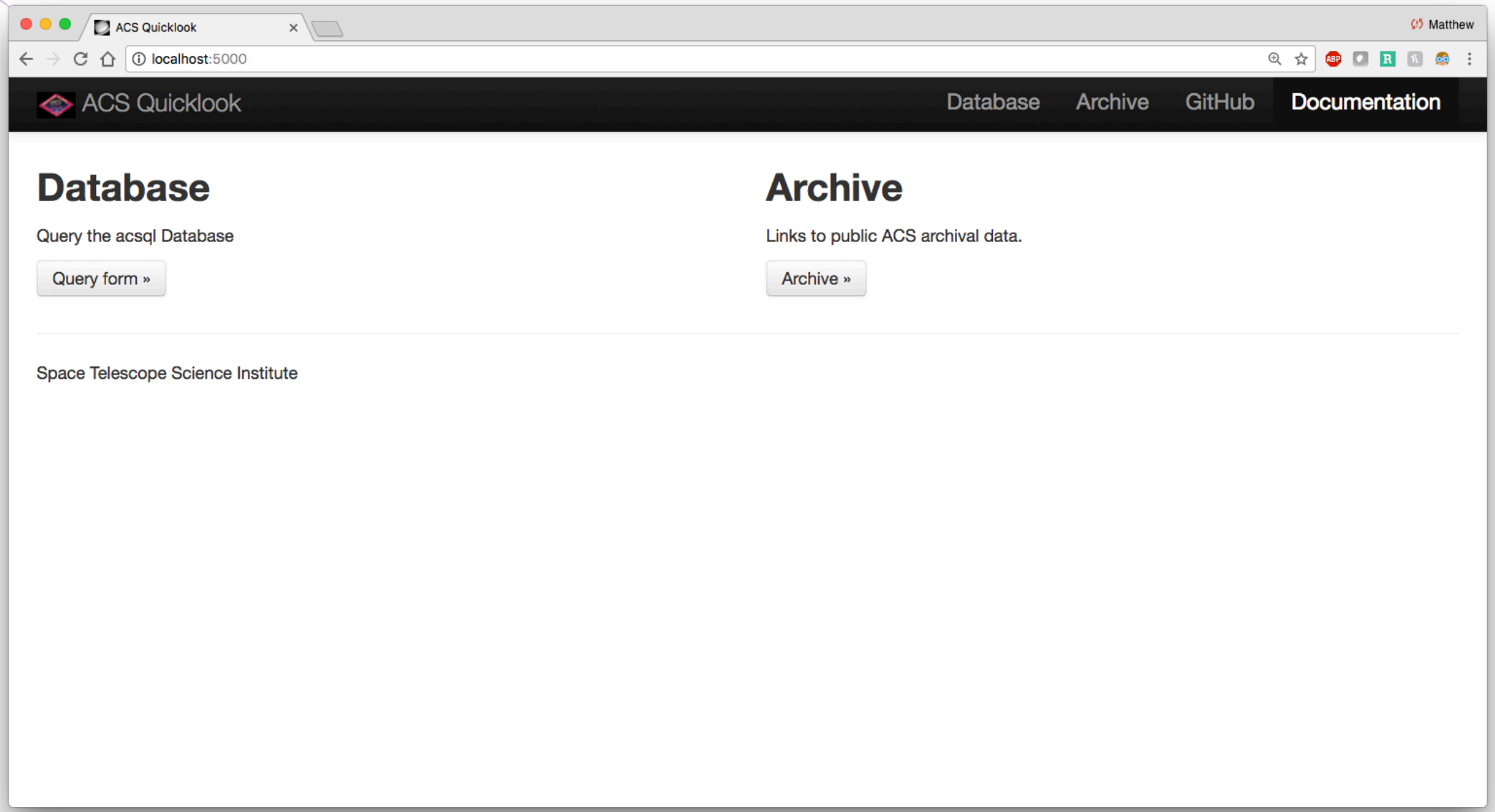
Implementation: acsql Package

```
acsql/
    __init__.py
    database/
        __init__.py
        database_interface.py
        make_tabledefs.py
        queries.py
        reset_database.py
        table_definitions/
            *.txt
        update_tabledefs.py
    ingest/
        __init__.py
        ingest.py
        make_file_dict.py
        make_jpeg.py
        make_thumbnail.py
    scripts/
        __init__.py
        ingest_production.py
    utils/
        __init__.py
        config.yaml
        utils.py
    website/
        __init__.py
        acsql_webapp.py
        data_containers.py
        form_options.py
        query_form.py
        query_lib.py
        static/
            css/
                *.css
            img/
                jpegs
                thumbnails
            js/
                *.js
        templates/
            *.html
```

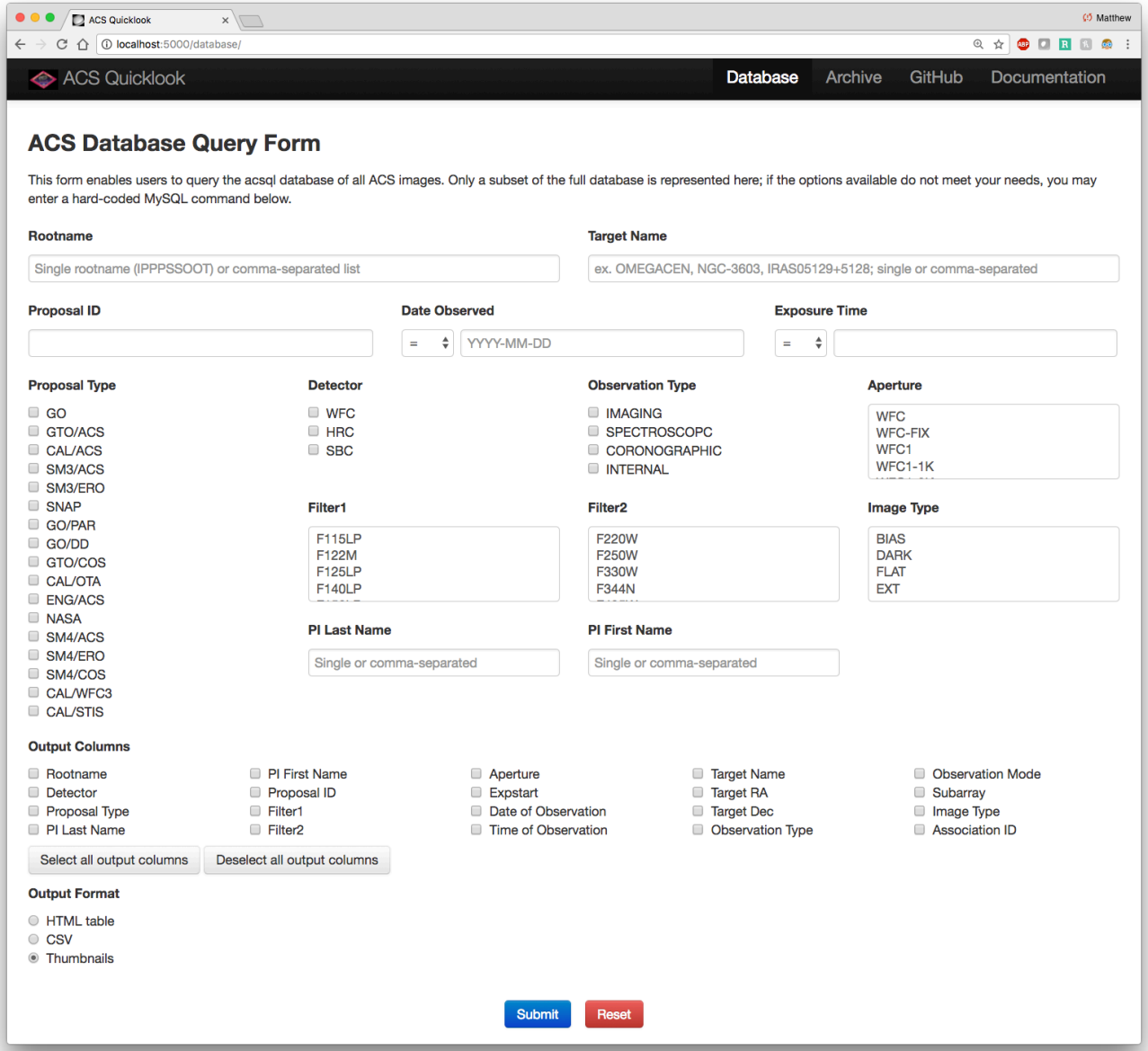


Available at http://github.com/spacetelescope/acsql

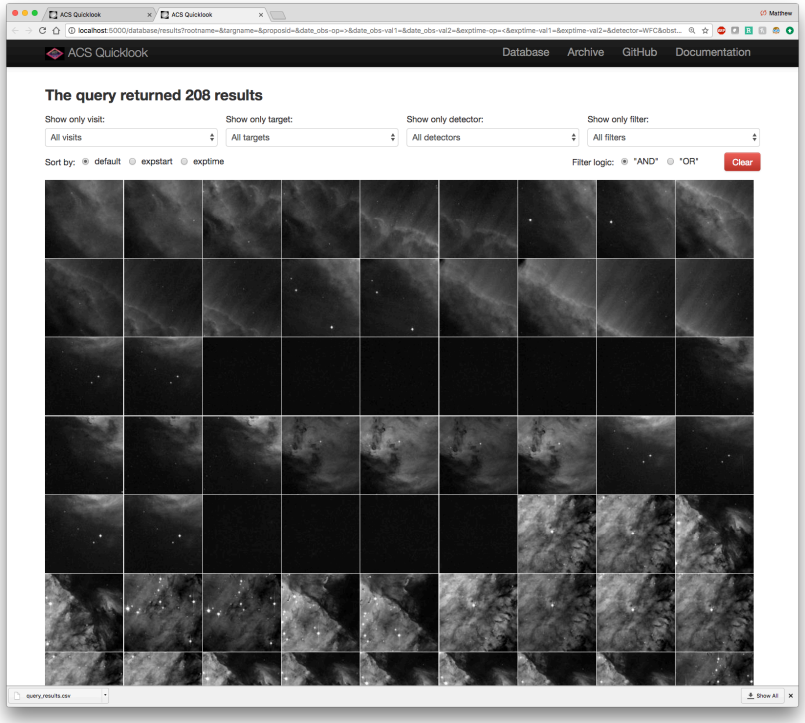
Web Application: Homepage



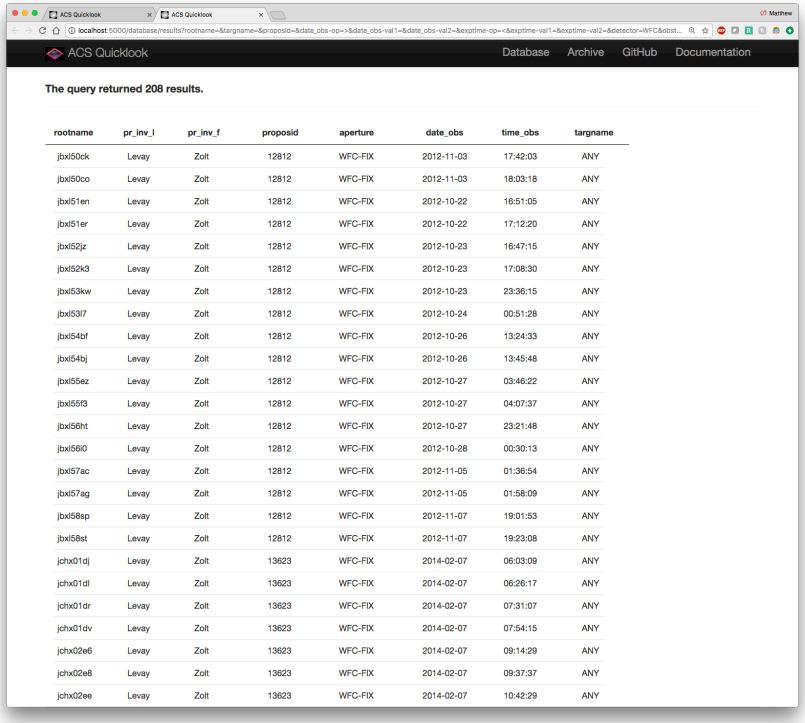
Web Application: Database Query Page



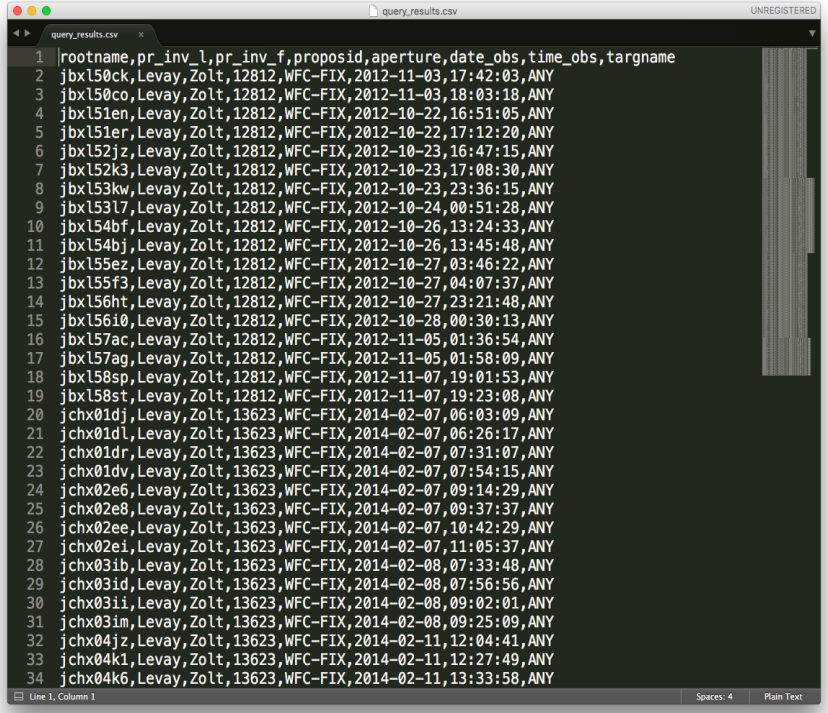
Web Application: Query Results



Thumbnails

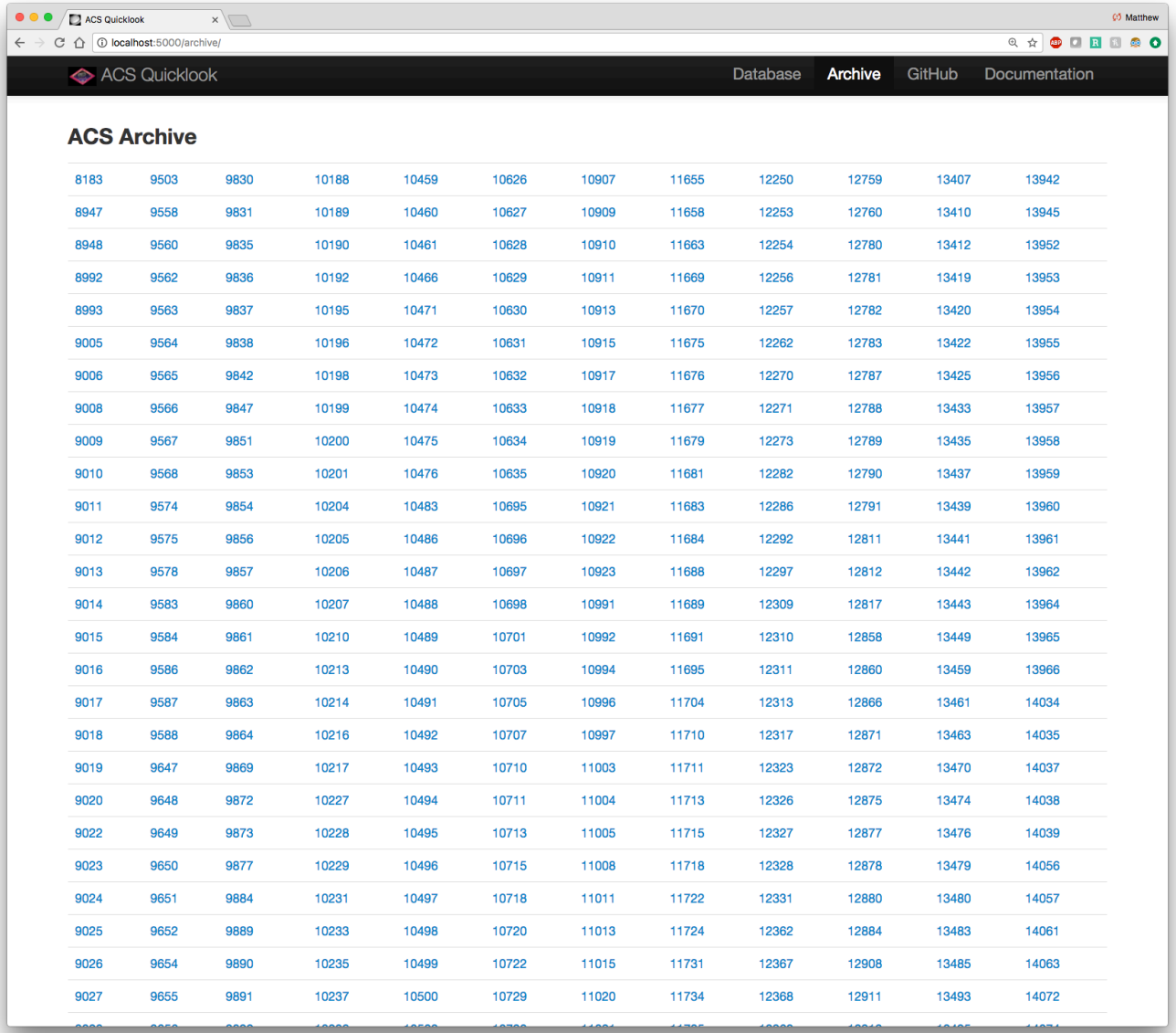


HTML Table

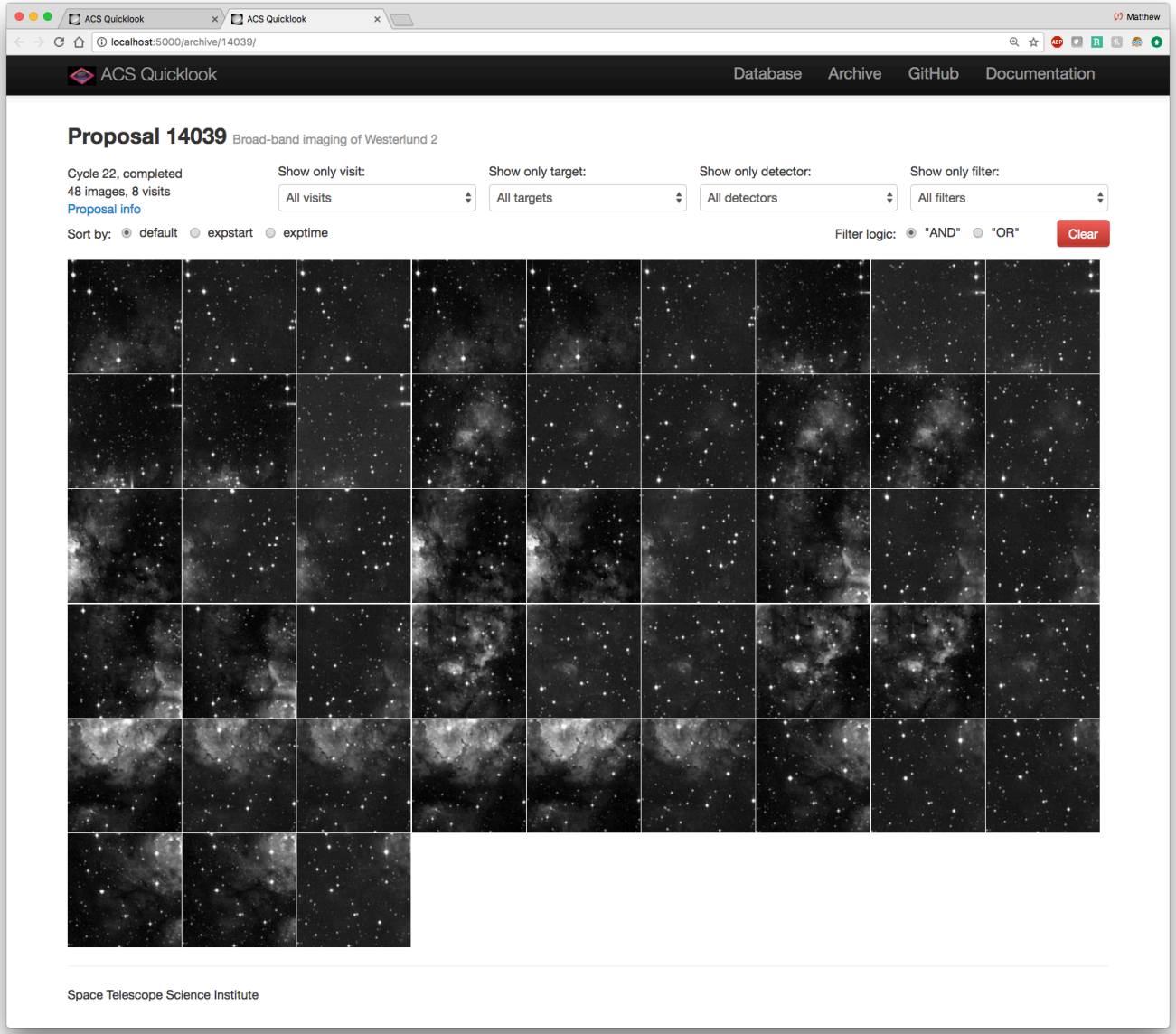


CSV File

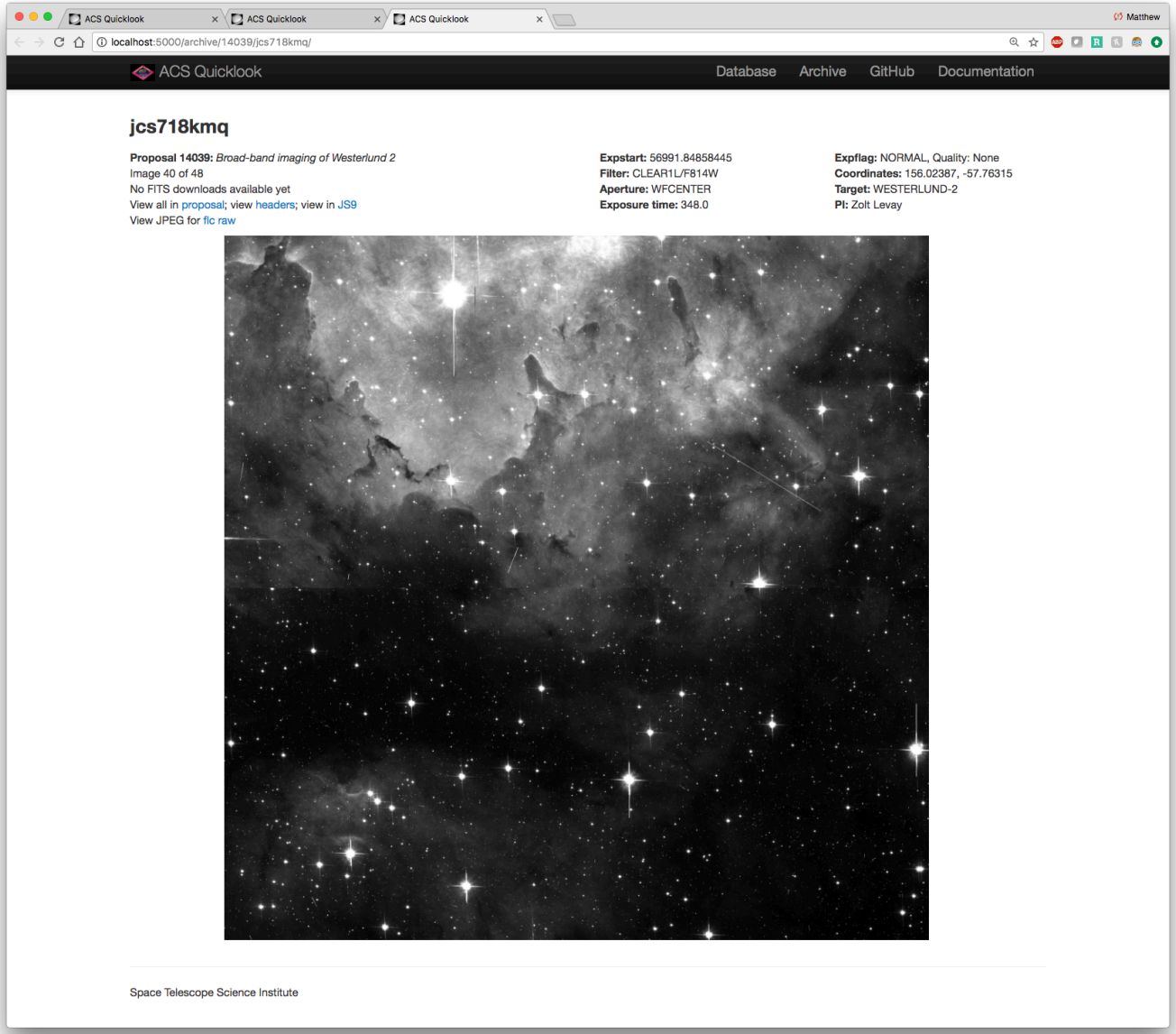
Web Application: ACS Archive



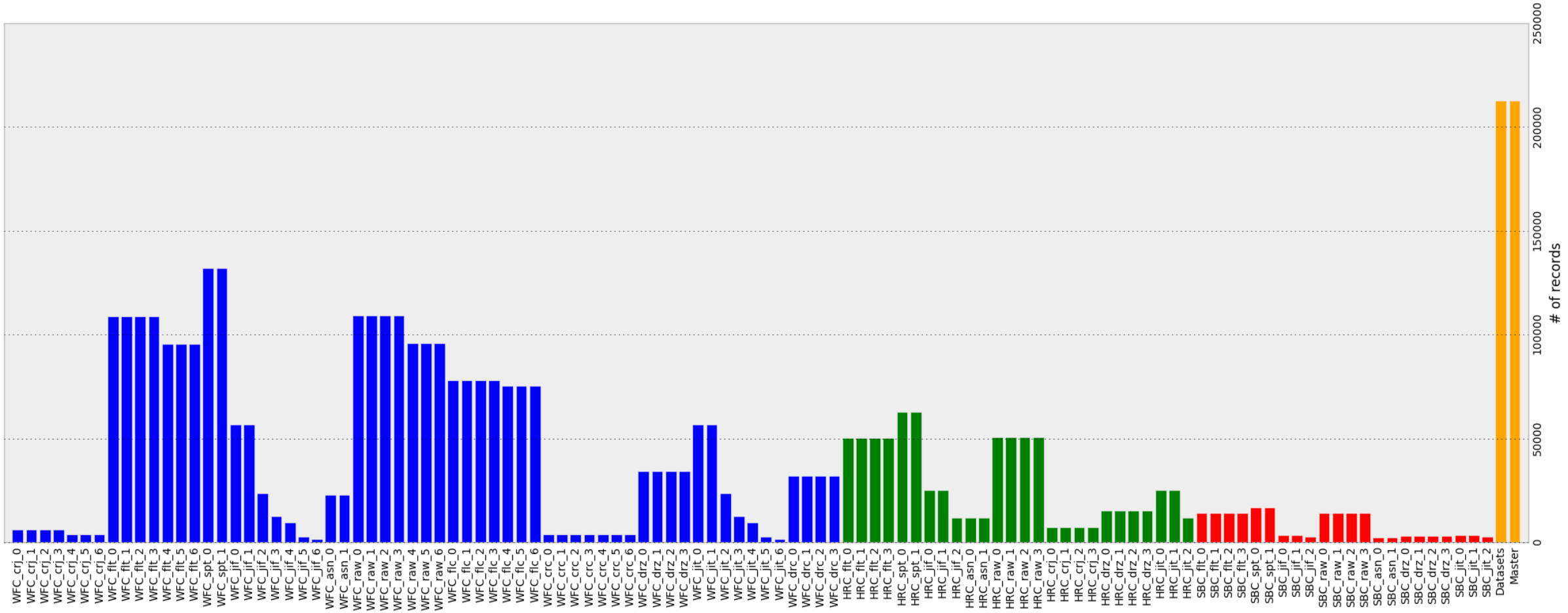
Web Application: View Proposal



Web Application: View Image

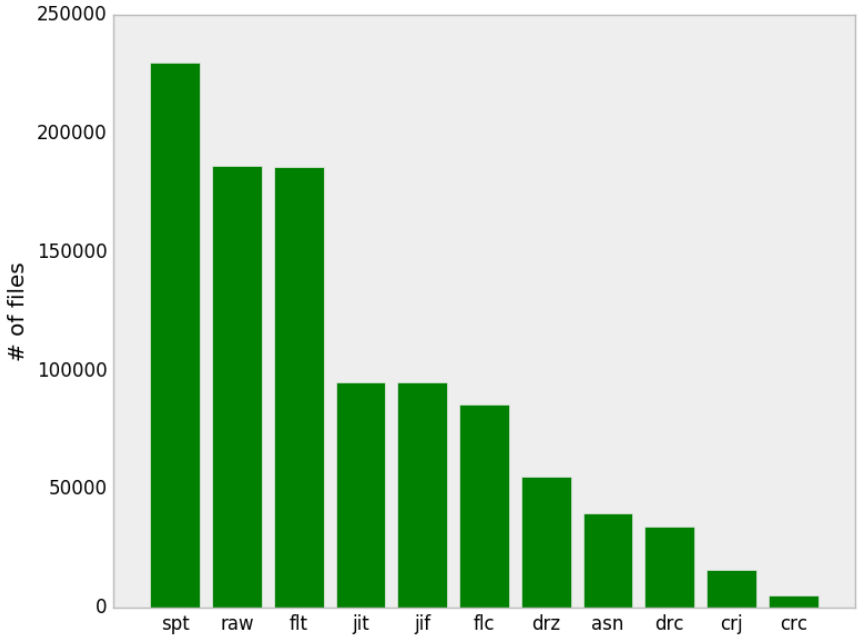


Deliverables

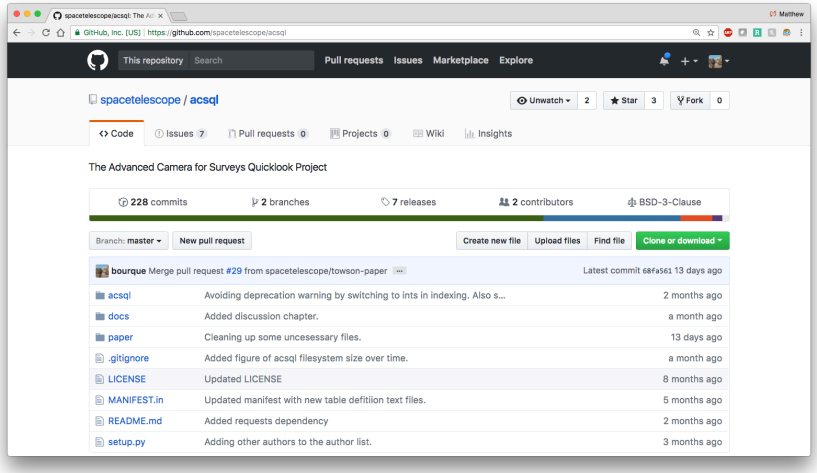


Database Deliverable

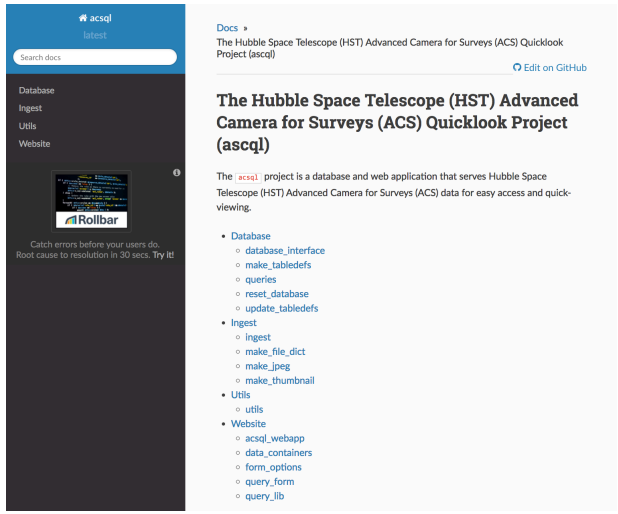
Deliverables



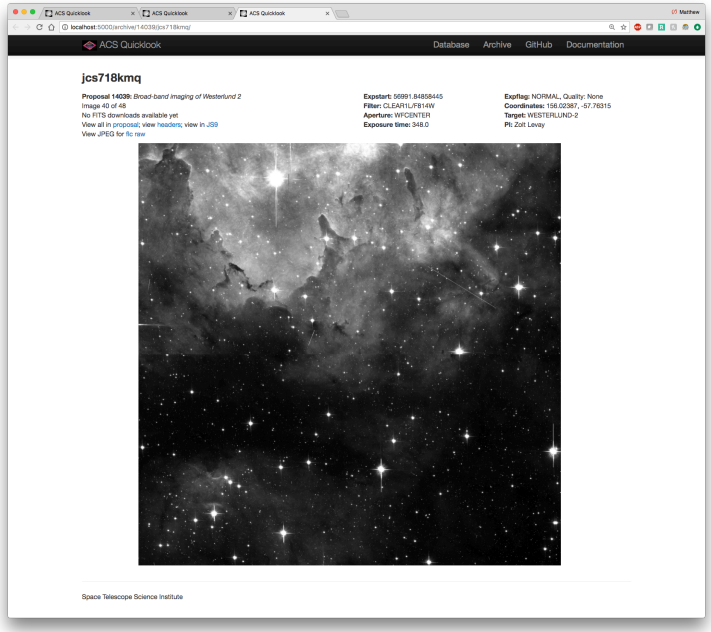
Filesystem Deliverable



Software Package Deliverable



API Documentation Deliverable



Web Application Deliverable

Acknowledgements

- The ACS Instrument Team: Roberto Avila, Andreas Bellini, Ralph Bohlin, David Borncamp, Marco Chi(something) Dan Coe, Tyler Dehjardins, Norman Grogin, Samantha Hoffman, Pey-Lian Lim, Ray Lucas, Aparna Maybhate, Matt McMaster, Vera Platais, Jenna Ryon
- The WFC3/Quicklook Team: Varun Bajaj, Ariel Bowers, Michael Dulude, Meredith Durbin, Jules Fowler, Catherine Gosmeyer, Heather Gunning, Harish Khandrika, Catherine Martlin, Jennifer Medina, Abhi Rajan, Clare Shanahan, Ben Sunnquist, Alex Viana
- Other Employees of STScI: Elena Sabbi, Francesca Boffi, Alex Yermolaev, Fausat Ogunsanya, Vera Gibbs
- Paper Reviewers: Sara Ogaz and Jules Fowler