Data services for scientific analysis on OMEGA and OMEGA EP

Matthew Galan

Fairport High School

Advisor: Richard Kidder

Laboratory for Laser Energetics

University of Rochester

Rochester, New York

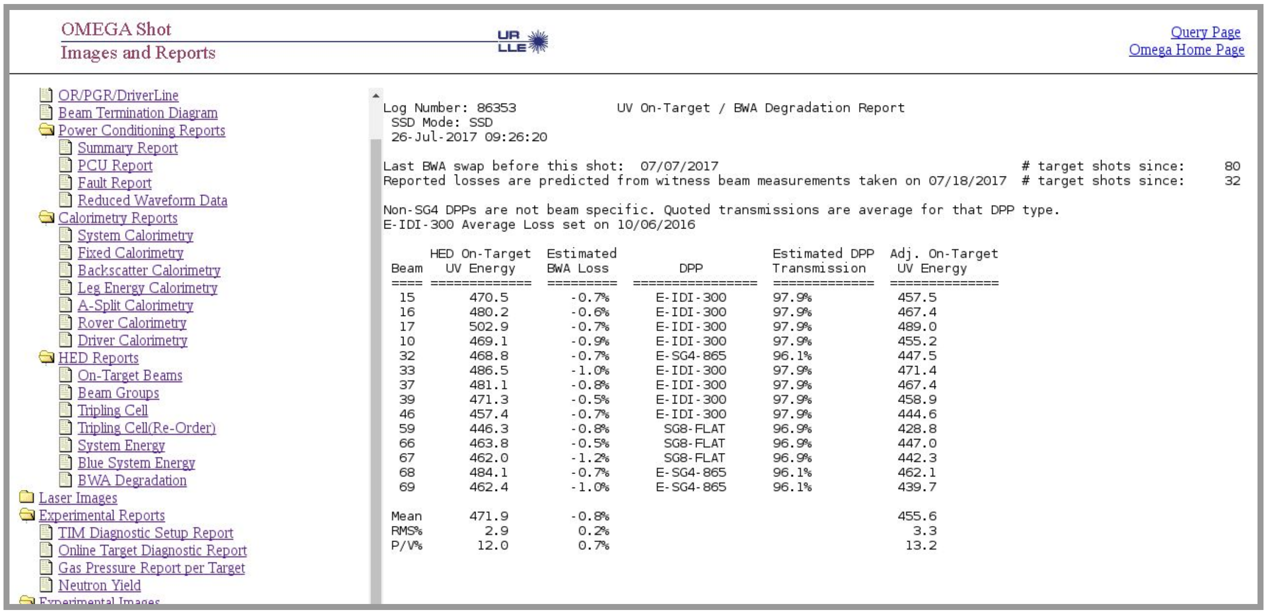
January 2018

1. Abstract

A data service was developed that allows scientists to easily incorporate data from experiments on the OMEGA and OMEGA EP lasers into their data analysis programs. This data service is designed to replace the previous method of manually copying and pasting from a static web-based report page, which was prone to error, and to augment database querying with the programming language SQL, which is complex and not accessible outside the laboratory's perimeter firewall. The data service allows scientists to programatically incorporate large amounts of data into their data analysis programs, including MatLab, Python, and Jupyter, regardless of their location. This simplifies data access for scientists and provides a one-step authentication scheme that is directly integrated into the data service, providing robust protection from unauthorized or malicious users.

2. Introduction

During experimental shot operations the OMEGA and OMEGA EP laser systems produce large quantities of highly valuable data on an hourly basis. While LLE prides itself on the efficiency of these lasers' operation, methods in which Principal Investigators (PIs) can currently access the data from their experiments are nothing short of obtuse, outdated, and prone to error. PIs had two options: use the OMEGA Shot Images and Reports page *(see Fig. 1)*, or try their hand at writing custom SQL code to manually query the database. *Fig. 2* illustrates the relationship between the database, Shot Images and Reports page, and PIs. With the OMEGA Shot Images and Reports Page, PIs had to sift through pages of plaintext data without a standard format in order to find the data they needed. Once found, PIs had to manually copy, paste, and format the data to fit it to their needs. If PIs were to use SQL, they would have to first understand the complex Omega database schemas and data relationships, then write upwards of fifty lines in order to connect to the database and pick out the data they needed *(See Fig. 3)*. Scientists external to the database networks can not access the database directly and cannot use SQL to query LLE data; they would either have to get it directly from a colleague in LLE, or by copying what is presented through web-based plaintext reports. Scientists that are within the facility that can access the database network were given the option of using a generic user account. However, a universal login does not protect data from those not authorized to see it, nor does it hold users connecting directly to the database accountable. In order to efficiently view, manipulate, and share experimental data, PIs need a solution that is secure, performant, and extendable.



OMEGA Shot
Images and Reports

Query Page
Omega Home Page

- OR/PGR/DriverLine
- Beam Termination Diagram
- Power Conditioning Reports
  - Summary Report
  - PCU Report
  - Fault Report
  - Reduced Waveform Data
- Calorimetry Reports
  - System Calorimetry
  - Fixed Calorimetry
  - Backscatter Calorimetry
  - Leg Energy Calorimetry
  - A-Split Calorimetry
  - Rover Calorimetry
  - Driver Calorimetry
- HED Reports
  - On-Target Beams
  - Beam Groups
  - Tripling Cell
  - Tripling Cell(Re-Order)
  - System Energy
  - Blue System Energy
  - BWA Degradation
- Laser Images
- Experimental Reports
  - TIM Diagnostic Setup Report
  - Online Target Diagnostic Report
  - Gas Pressure Report per Target
  - Neutron Yield
- Experimental Images

```
Log Number: 86353           UV On-Target / BWA Degradation Report
 SSD Mode: SSD
 26-Jul-2017 09:26:20

Last BWA swap before this shot:  07/07/2017                                    # target shots since:     80
Reported losses are predicted from witness beam measurements taken on 07/18/2017  # target shots since:     32

Non-SG4 DPPs are not beam specific. Quoted transmissions are average for that DPP type.
E-IDI-300 Average Loss set on 10/06/2016
```

Beam	HED On-Target UV Energy	Estimated BWA Loss	DPP	Estimated DPP Transmission	Adj. On-Target UV Energy
15	470.5	-0.7%	E-IDI-300	97.9%	457.5
16	480.2	-0.6%	E-IDI-300	97.9%	467.4
17	502.9	-0.7%	E-IDI-300	97.9%	489.0
10	469.1	-0.9%	E-IDI-300	97.9%	455.2
32	468.8	-0.7%	E-SG4-865	96.1%	447.5
33	486.5	-1.0%	E-IDI-300	97.9%	471.4
37	481.1	-0.8%	E-IDI-300	97.9%	467.4
39	471.3	-0.5%	E-IDI-300	97.9%	458.9
46	457.4	-0.7%	E-IDI-300	97.9%	444.6
59	446.3	-0.8%	SG8-FLAT	96.9%	428.8
66	463.8	-0.5%	SG8-FLAT	96.9%	447.0
67	462.0	-1.2%	SG8-FLAT	96.9%	442.3
68	484.1	-0.7%	E-SG4-865	96.1%	462.1
69	462.4	-1.0%	E-SG4-865	96.1%	439.7
Mean	471.9	-0.8%			455.6
RMS%	2.9	0.2%			3.3
P/V%	12.0	0.7%			13.2

Fig 1: The LLE OMEGA Short Images and Reports page

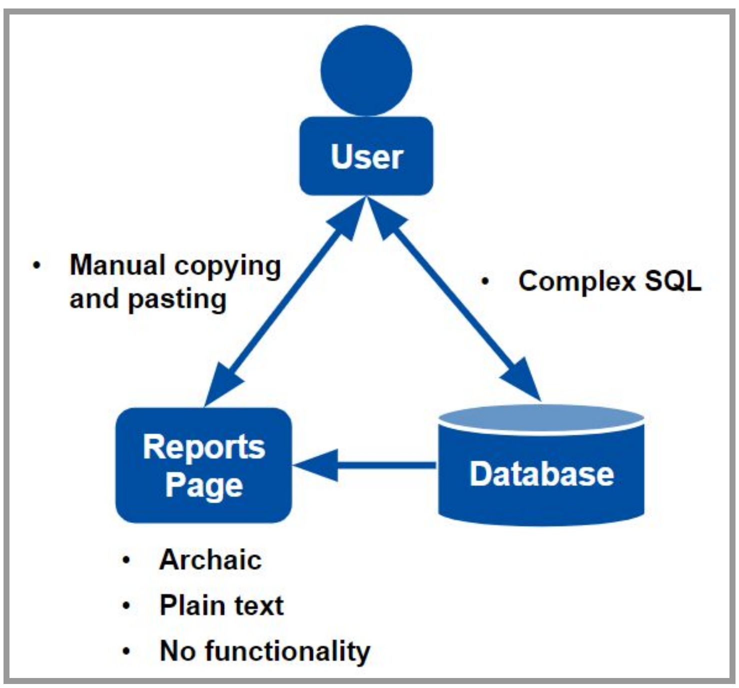


Fig 2: A triangular relationship between the user (PIs), database, and reports page.

```
function dbConn=getLLEdatabase(<name>, genericFlag)
% Use generic/generiuc if the user requests it.
if nargin < 2 || isempty(genericFlag) || ~genericFlag
    uname = '';
    pass = '';
else
    uname = '<user>';
    pass = '<password>';
end
if nargin < 1 || isempty(<name>)
    % A function argument can override the environment variable
    <name> = getenv('DATABASE');
    if isempty(<name>)
        % The replication database is the default
        <name> = 'rep';
    end
end

% On windows, this is all we need to do.
if strfind(computer, 'PCWIN')
    dbConn = database(<name>, pass);
else

% If we get here, it's not windows. JDBC drivers should be
% available. Use them.
javaaddpath(fullfile(getenv('ORACLE_HOME'), 'lib', 'ojdbc6.jar'))
switch <name>
    case 'exp'
        <host> = 'hostname:port';

    case {'ep_exp', 'epexp'}
        <host> = 'hostname:port';
        <name> = 'epexp';

    case {'dev', 'ep_dev'}
          %walnut?
        <host> = 'hostname:port';

    otherwise
        error('ckin:UnknownDatabase',...
            'Database ID "%s" is unknown.', <name>);
end

dbConn = database(<name>, uname, pass,...
    'oracle.jdbc.driver.OracleDriver',...
        ['jdbc:oracle:thin:@' <host> ':' <name>]);
end

% Make sure it's a working connection, or error out.
assert(isconnection(dbConn) == 1,...
    'ckin:LLEdatabase:ConnectionFailed',...
    'Connection to database "%s" failed: %s', dbName, dbConn.Message)
```

Fig 3: An example of the MatLab code needed to query the LLE database

3. Development

A data service was developed that would eliminate the need for PIs to retrieve the data from the OMEGA Shots and Reports page or write their own SQL to query the database. The data service calls prewritten SQL procedures inside the database in order to retrieve needed data *(see Fig. 4).*

Fig. 4: An example database procedure that would be called by the data service.

The data service then formats the data into JavaScript Object Notation (JSON), a human readable format that is compatible with any programming language or most data analysis programs (including MatLab 2016 and up - earlier versions require a publicly available toolkit for JSON) . See *Fig. 5* for an example of the JSON the data service returns.

```
[
  {
    "log": 84251,
    "hs_date": "2017-01-31T18:14:36",
    "beam": 10,
    "ir": 169.48577839206084,
    "green": 42.178540427902796,
    "uv": 510.83841640208857,
    "hed_total": 722.5027352220521,
    "cell_in": 921.6612565725013,
    "conv": 0.5794044704649481,
    "uv_on_target": 446.8364526458662,
    "cal_date": "2017-01-05T14:37:26",
    "trans_date": "2017-01-30T08:39:06",
    "dpr_serial": "352-22",
    "dpr_in_out": "IN ",
    "ocl_set": 513,
    "ssd_mode": "SSD"
  },
  {
    "log": 84251,
    "hs_date": "2017-01-31T18:14:36",
    "beam": 11,
    "ir": 171.45534470198933,
    "green": 41.44056133164611,
    "uv": 511.16208851801423,
    "hed_total": 724.0579945516497,
    "cell_in": 924.1653772613896,
    "conv": 0.5786240491859618,
    "uv_on_target": 446.89128855531703,
    "cal_date": "2017-01-05T14:37:26",
    "trans_date": "2017-01-30T08:39:06",
    "dpr_serial": "307-08",
    "dpr_in_out": "IN ",
    "ocl_set": 513,
    "ssd_mode": "SSD"
  },
```


Fig. 5: A snippet of JSON that the data service may return.

In order to maximize compatibility and usability, the data service is a RESTful web service. The service returns data based only on parameters in the URL that the user provides. Since all modern programming languages are capable of retrieving content provided by a webpage, the service is both operating system and language agnostic, meaning PIs can access the data easily no matter what their preferences may be. Given its web-based nature, the service is also location agnostic, given that users are authenticated. Furthermore, PIs working with large quantities of data may do so easily by simply iterating their URL parameter of choice. This allows for blocks of data to be retrieved in a convenient, organized manner.

In order to use the dataservice, PIs need to obtain a token from a web portal and then insert a url provided by the dataservice. Example implementations for Python 2 and MatLab 2016 are demonstrated in *Fig. 6* and *Fig. 7*, respectively.

```python
# Import packages
import urllib
import json
# Pick a procedure and shot number
procedure = "asbol"
shot_num = 52818
token = "d798ee4c30bc9c0808e6debb797a4e7cb19b8f4bdd5f58f9b7fcc794538c752c"
# Get a dictionary from the JSON using json.loads()
data = json.loads(urllib.urlopen("http://lle-stage-informatics2:3140/api/" + procedure + "/" + str(shot_num)).read() + "/" + token)
# Example use of dictionary - get the sae_offset of the first entry
print data[0]["sae_offset"]
```

Fig 6: The only Python 2 code needed to use the data service.

```matlab
% Pick a procedure and shot number
procedure = 'asbol';
shot_num = 52818;
token = 'd798ee4c30bc9c0808e6debb797a4e7cb19b8f4bdd5f58f9b7fcc794538c752c';
% Get a struct from the JSON using webread()
data = webread(strcat('http://lle-stage-informatics2:3140/api/', procedure, '/', num2str(shot_num), '/', token));
% Example use of struct - get the sae_offset of the first entry
val = data(1).sae_offset;
```

Fig 7: The only MatLab 2016+ code needed to use the data service.

Several improvements were made to the code driving the data service. The original data service code had a separate function written for each database procedure. The code inside of these functions was identical, except for the names of the database procedure, which were hard coded in. This meant for every database procedure that was written, of which there would be dozens in the future, the same code block in the data service code had to be duplicated and the name for the procedure modified. *Fig. 8* Illustrates the hard coded nature of the functions.

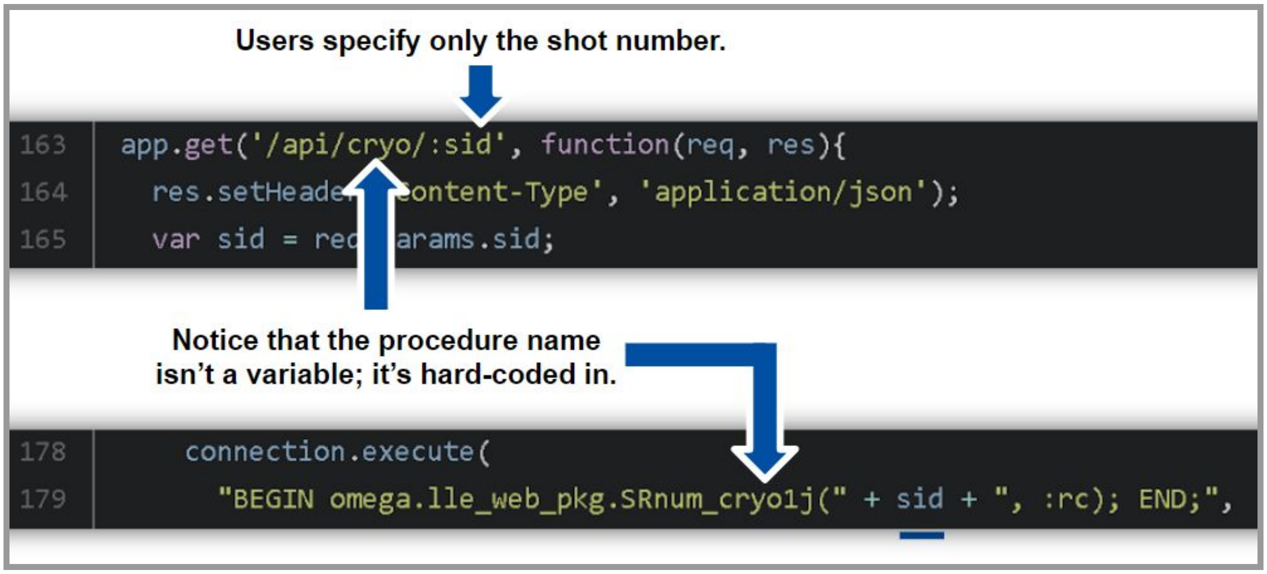


Fig. 8: Snippets of a data service function

The data service functions in question were reduced down to a single template version that takes the procedure name as an argument provided in the URL in addition to the shot number. Additionally, user authentication was being developed at the same time. Users would provide their unique, temporary ID string at the end of the URL. The data service would then verify that the user was authorized to view the data they were trying to access. *Fig. 9* reveals the changes made to the data service functions.

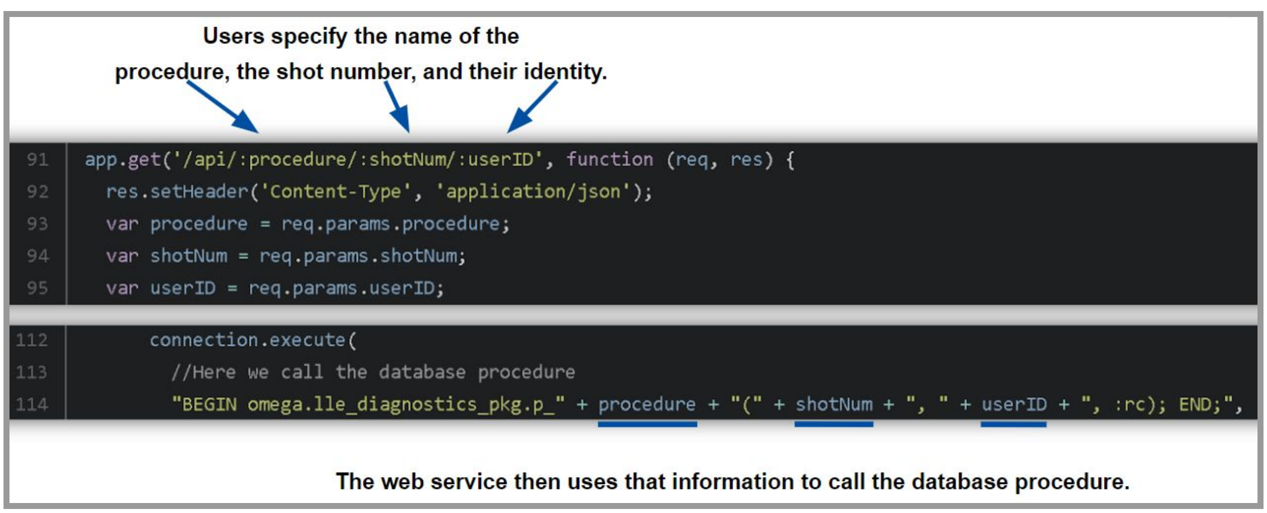


Fig 9: The new, template version of the data service functions.

These revisions were successful in reducing both the size of the code behind the data service and the work required to maintain its functionality. *Fig. 10* provides a visual representation of the resultant code reduction.

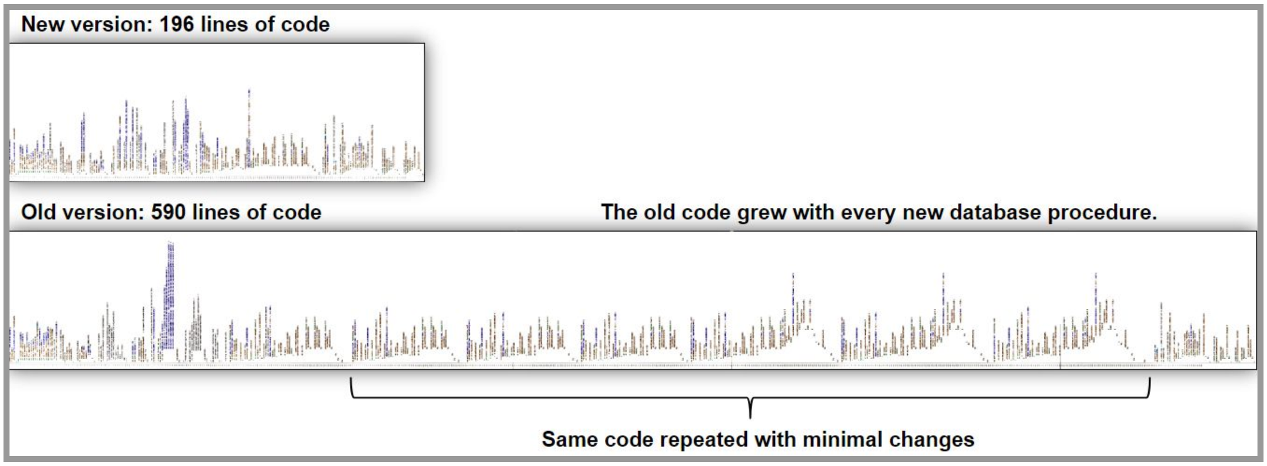


Fig 10: A side by side comparison of actual pictures of the old and new code behind the data service. The code displayed is rotated 90°.

5. Future Plans

Built-in data service support for the LLE diagnostic analysis page is still under development. This feature would allow PIs to view and graph selected data instantly without having to pull the data from the data service using a programming language (such as Python 2) or a data analysis program (such as MatLab 2016). The data service integration is also the mechanism that provides PIs with an easy way to get the URL needed to get a specific data set from the data service. See *Fig. 11* for a mockup of what the LLE diagnostic analysis page may look like with these features.

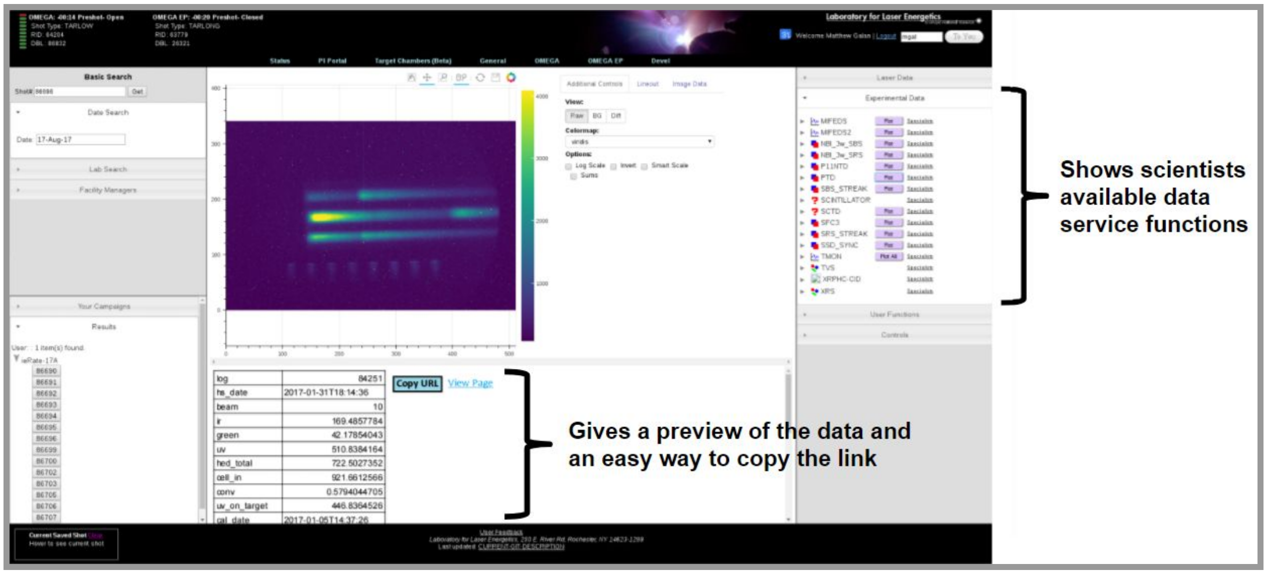


Fig 11: A mockup of the in-development diagnostic analysis page

6. Conclusion

In order to streamline the data retrieval process for the OMEGA and OMEGA EP laser systems, a data service was developed that allows PIs to seamlessly incorporate data from the database into their favorite data analysis solutions using only a few lines of prewritten code. Older methods involved manually copying, pasting, and formatting large blocks of data from the OMEGA Shot Images and Reports page, or directly connecting to the database with over fifty lines of SQL. Integration into the LLE Diagnostic Analysis page, as well as user authentication, is currently in development.

7. Acknowledgements

My sincerest gratitude to Mr. Richard Kidder, Dr. Stephen Craxton, and Ms. Jean Steve for making this project, as well as the LLE Summer Research Program as a whole, possible. This was not only an incredible opportunity, but a tremendous learning experience. Of course, I would be nowhere without the unending support from my family and friends. Thank you.