Intellect Solutions Technical Approach

Submitted in Response to:

Agile Service Delivery

4QTFHS150004

For:

**The General Services Administration (GSA)
Federal Acquisition Service, Integrated Technology Service
National IT Commodity Program
401 West Peachtree Street NW, Suite 820
Atlanta, GA 30308**

By:

**Intellect Solutions LLC
GSA Contract # GS-35F-144AA
5602 Pickwick Road
Centreville, VA 20120
Dr. Sunny Singh, COO, Phone: (703) 898-1498**

Table of Contents

List of Figures

1.0 Prototype Name

Adverse Drug Effects Research System (ADERS)

2.0 Use Cases addressed by ADERS

After interviewing few sample users, Intellect Solutions team came up with two use cases that would be very useful to the end users:

- I was told to take certain drug, but I am not sure if I should take that drug. I would like to find out adverse effects of the drugs suggested to me and confirm that I have all the correct label and warning information for the drugs I am using.
- I would like to stress the importance on maintaining privacy during the search.

3.0 Introduction

Our site is designed to provide users with ability to research on adverse side effects of the brand name and generic drugs and medical devices. While building the prototype, Adverse Drug Effects Research System (ADERS), Intellect followed all the 13 plays from the US Digital Services Playbook to ensure we deliver successful Agile Service prototype to GSA 18F.

Below are the details of each play and how we followed them and customized to 18F's requirements.

4.0 Requirements Development Phase

This section discusses our approach to Play 1:

- *Play 1: Understand what people need*

Our approach began with studying prototype requirements and performing research on the APIs provided by open.fda.gov. After reviewing the APIs, we identified some sample users who might use that information and conducted interviews and meetings. GSA was also identified as a stakeholder based on the requirements in the RFQ. After these meetings, our team developed user stories in Taiga.Io that addressed following information:

- Functionality
- Usability and 508 compliance
- Information displayed
- Security
- Technical designs and considerations

5.0 Prototype Design Phase

This section discusses our approach to the following Plays:

- *Play 2: Address the whole experience, from start to finish*
- *Play 3: Make it simple and intuitive*

Based on the requirements gathered from our sample users, Intellect developed high level user stories that describe end-to-end prototype behavior. We engaged an experienced team member to serve in the capacity of Usability expert and front-end developer. We developed mock-ups, created user flows, and addressed 508 compliant elements in our design. We researched different websites that are top-ranked in terms of usability such as http://www.rxlist.com/script/main/hp.asp; http://www.drugs.com/; and https://www.myprime.com/ to understand latest design elements. Additionally, we also reviewed concepts used by 18F on their consulting projects and developed the user experience.

Our prototype includes other elements such as gather user feedback using Google forms and help for users to easily perform search.

6.0 Technical Design Phase

This section discusses our approach to the following plays:

- *Play 5: Structure budgets and contracts to support delivery*
- *Play 8: Choose a modern technology stack*
- *Play 9: Deploy in a flexible hosting environment*
- *Play 13: Default to open*

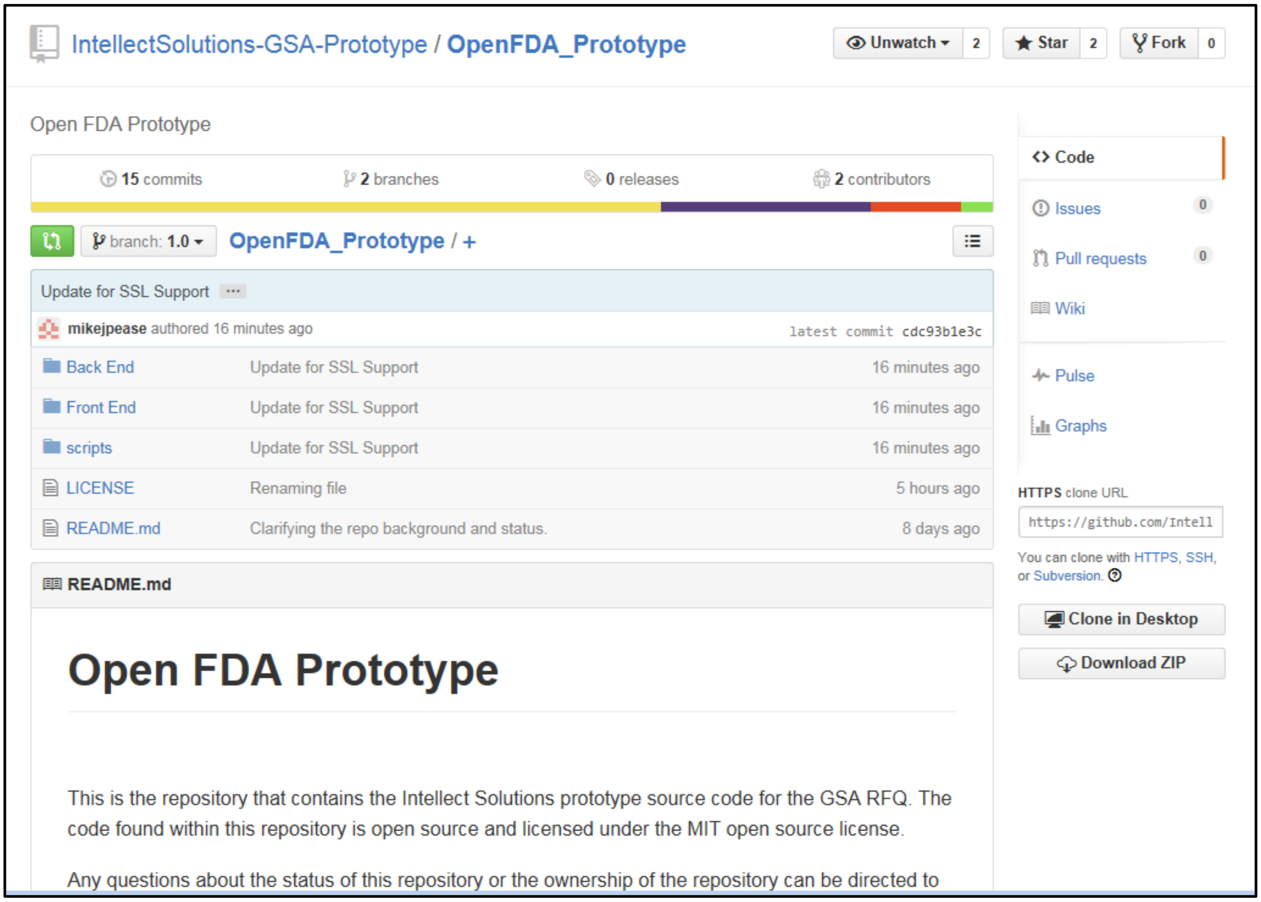
Intellect Solutions team chose all the sub-components that make up our prototype to align with US Digital Services Playbook, industry standard tools, industry leading practices, and 18F's approach to software development, to meet following criteria:

- Use latest and reliable tools used in the industry
- Increase reusability
- Open source and low startup cost
- Fast turn-around time

Intellect Solutions team used the following open-source and modern technologies:

- SaaS hosting provider selected to host the prototype is Amazon Web Services
- Source control repository is Github (Figure 1)
- Project management tool is open source and free Taiga.Io (Figure 2)
- CentOS virtual server is hosted on Amazon Web Services (Figures 3 and 4)
- Docker Containers are used to package an application into a standardized unit
- Kali Linux platform is used for vulnerability testing

- Open VAS is used for vulnerability testing
- Front-end development framework is angular.js
- Back-end development framework is node.js
- JSON API specifications



Update for SSL Support …

mikejpease authored 16 minutes ago latest commit cdc93b1e3c

Back End	Update for SSL Support	16 minutes ago
Front End	Update for SSL Support	16 minutes ago
scripts	Update for SSL Support	16 minutes ago
LICENSE	Renaming file	5 hours ago
README.md	Clarifying the repo background and status.	8 days ago

You can clone with HTTPS, SSH, or Subversion.

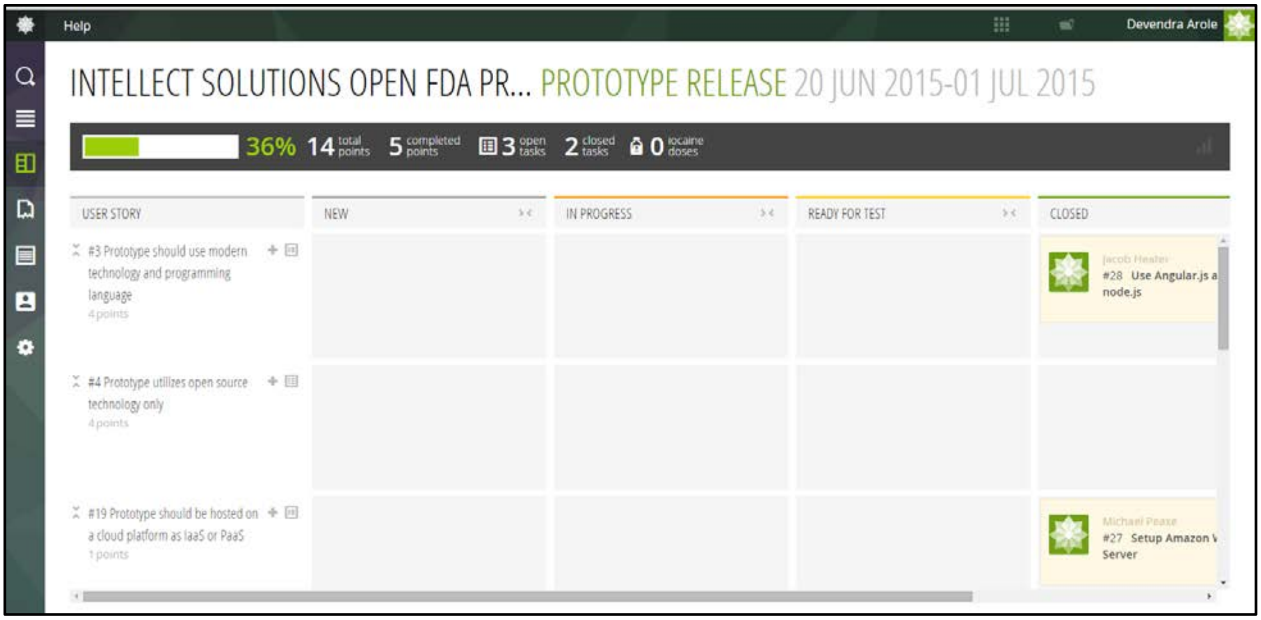
README.md

Open FDA Prototype

This is the repository that contains the Intellect Solutions prototype source code for the GSA RFQ. The code found within this repository is open source and licensed under the MIT open source license.

Any questions about the status of this repository or the ownership of the repository can be directed to

Figure 1: GitHub Repository Source Control Repository



Help

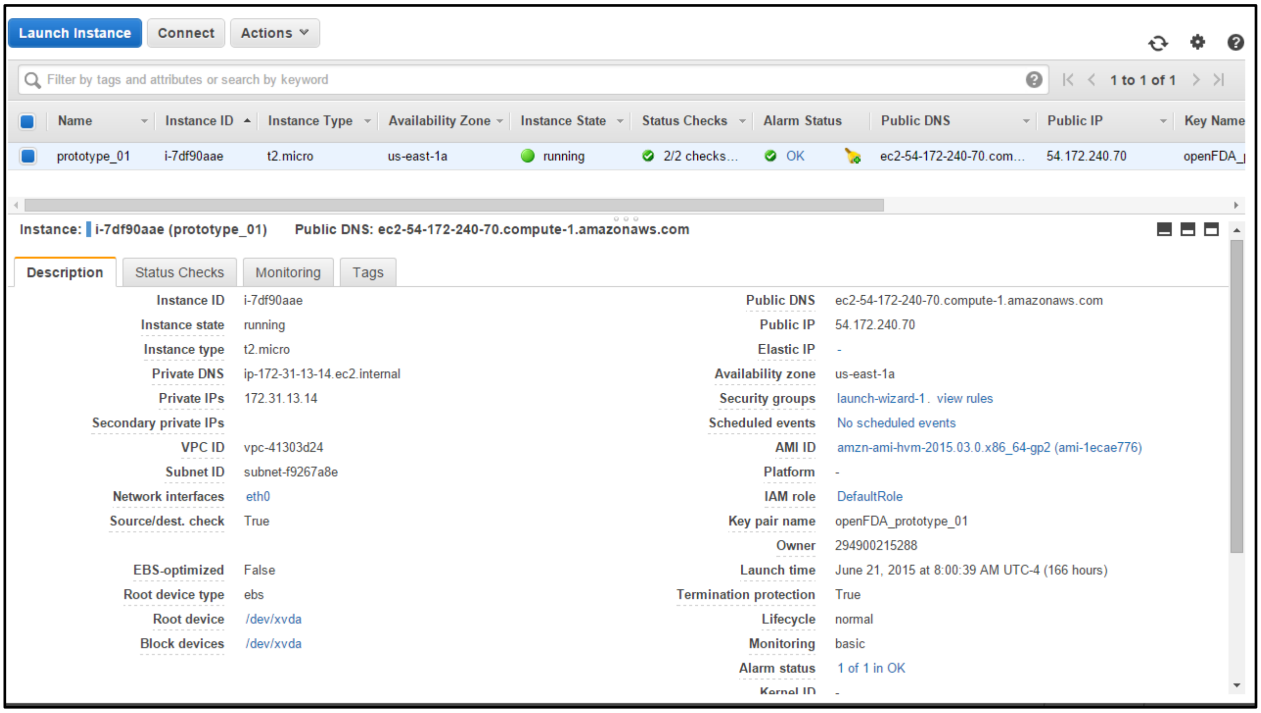
Devendra Arole

INTELLECT SOLUTIONS OPEN FDA PR... PROTOTYPE RELEASE 20 JUN 2015-01 JUL 2015

36% 14 total points 5 completed points 3 open tasks 2 closed tasks 0 iocaine doses

USER STORY	NEW	IN PROGRESS	READY FOR TEST	CLOSED
#3 Prototype should use modern technology and programming language 4 points				Jacob Heater #28 Use Angular.js a node.js
#4 Prototype utilizes open source technology only 4 points				
#19 Prototype should be hosted on a cloud platform as IaaS or PaaS 1 points				Michael Peake #27 Setup Amazon V Server

Figure 2: Tiaga.io Collaboration Site

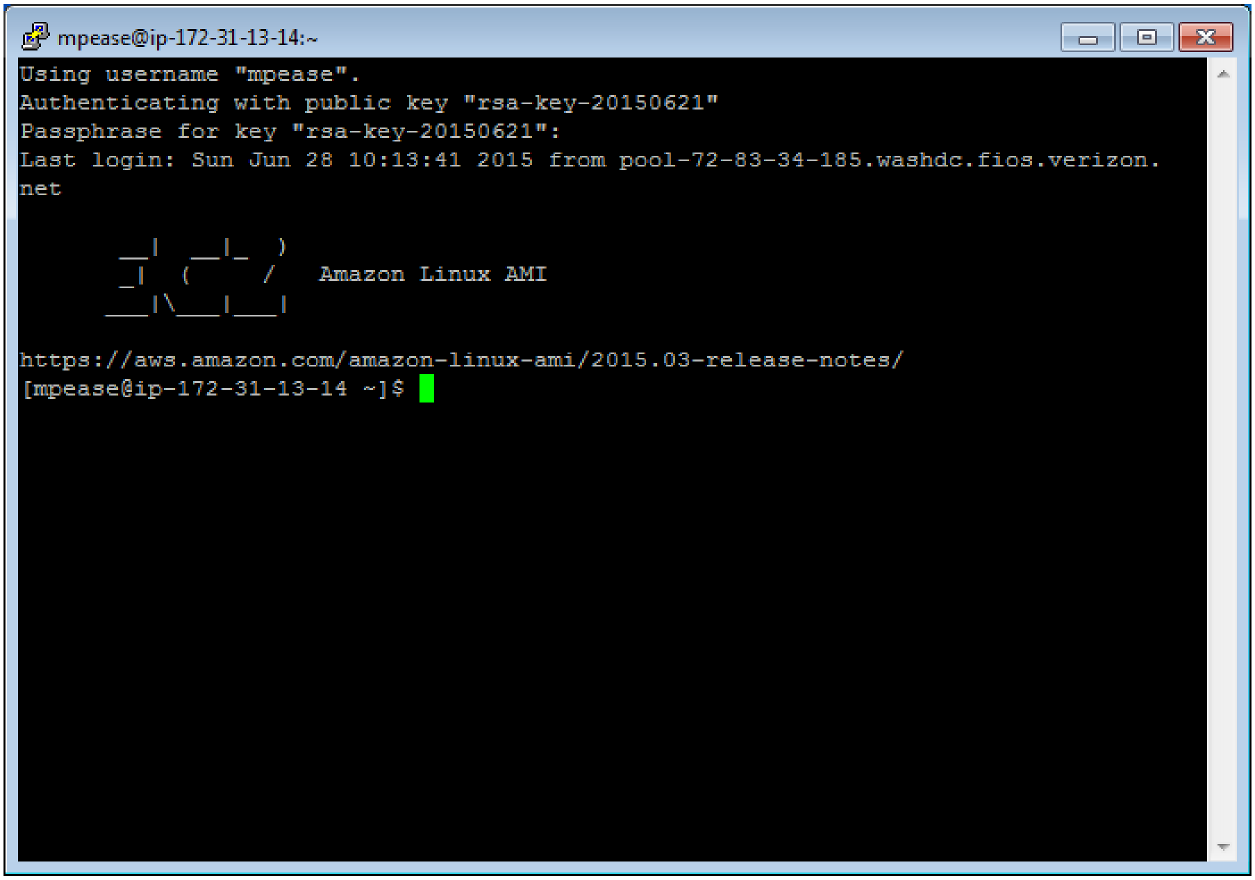


Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS	Public IP	Key Name
prototype_01	i-7df90aae	t2.micro	us-east-1a	running	2/2 checks...	OK	ec2-54-172-240-70.com...	54.172.240.70	openFDA_

Instance: i-7df90aae (prototype_01) Public DNS: ec2-54-172-240-70.compute-1.amazonaws.com

Instance ID	i-7df90aae	Public DNS	ec2-54-172-240-70.compute-1.amazonaws.com
Instance state	running	Public IP	54.172.240.70
Instance type	t2.micro	Elastic IP	-
Private DNS	ip-172-31-13-14.ec2.internal	Availability zone	us-east-1a
Private IPs	172.31.13.14	Security groups	launch-wizard-1. view rules
Secondary private IPs		Scheduled events	No scheduled events
VPC ID	vpc-41303d24	AMI ID	amzn-ami-hvm-2015.03.0.x86_64-gp2 (ami-1ecae776)
Subnet ID	subnet-f9267a8e	Platform	-
Network interfaces	eth0	IAM role	DefaultRole
Source/dest. check	True	Key pair name	openFDA_prototype_01
		Owner	294900215288
EBS-optimized	False	Launch time	June 21, 2015 at 8:00:39 AM UTC-4 (166 hours)
Root device type	ebs	Termination protection	True
Root device	/dev/xvda	Lifecycle	normal
Block devices	/dev/xvda	Monitoring	basic
		Alarm status	1 of 1 in OK
		Kernel ID	-

Figure 3: Amazon Web Service Prototype Instance Configuration Summary



```
Using username "mpease".
Authenticating with public key "rsa-key-20150621"
Passphrase for key "rsa-key-20150621":
Last login: Sun Jun 28 10:13:41 2015 from pool-72-83-34-185.washdc.fios.verizon.
net

       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-ami/2015.03-release-notes/
[mpease@ip-172-31-13-14 ~]$ 
```

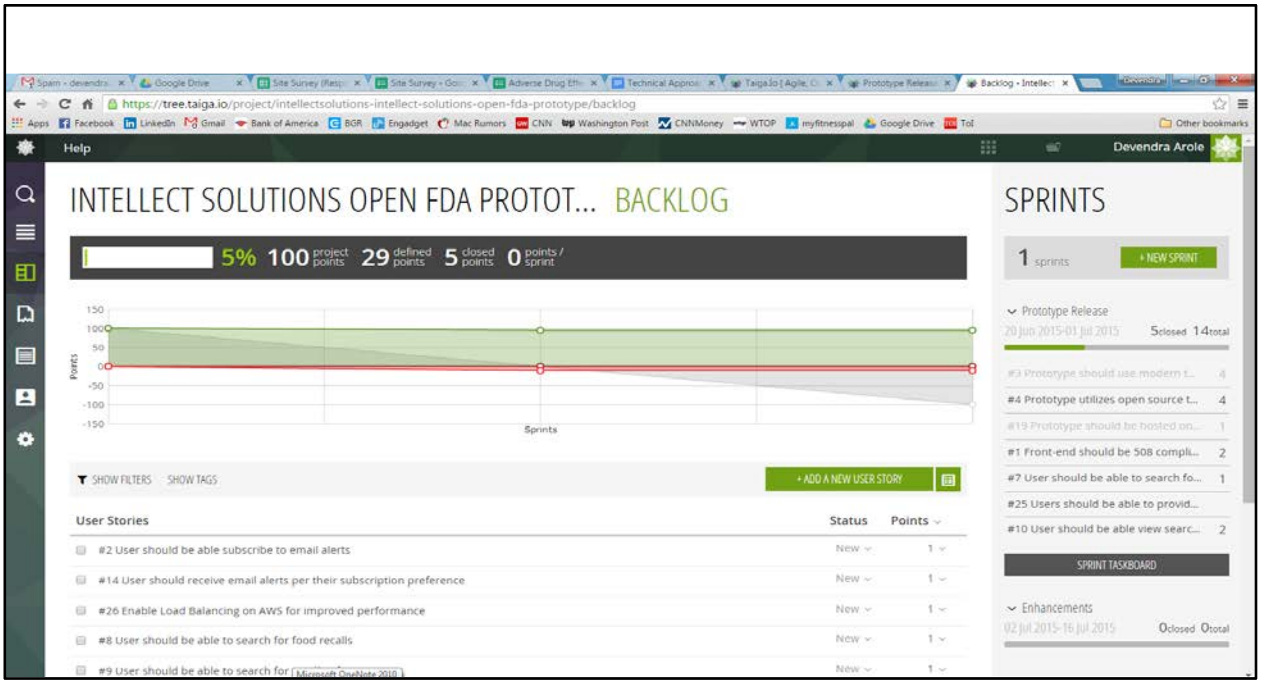
Figure 4: Login Screen to AWS Instance with AMI Release Information Reference

7.0 Prototype Development Phase

This section discusses our approach to the following plays:

- *Play 4: Build the service using agile and iterative practices*
- *Play 6: Assign one leader and hold that person accountable*
- *Play 7: Bring in experienced teams*
- *Play 10: Automate testing and deployments*

Intellect Solutions team is adept at Agile development methodology, having done this for other government clients. After reviewing the requirements, gathering user stories by interacting with sample users via interviews and meetings, Intellect Solutions team quickly identified core team members required to develop the prototype, began daily Agile prints, and developed user stories (Figures 5 and 6).



INTELLECT SOLUTIONS OPEN FDA PROTOT... BACKLOG

5% 100 project points 29 defined points 5 closed points 0 points / sprint

User Stories	Status	Points
□ #2 User should be able subscribe to email alerts	New	1
□ #14 User should receive email alerts per their subscription preference	New	1
□ #26 Enable Load Balancing on AWS for improved performance	New	1
□ #8 User should be able to search for food recalls	New	1
□ #9 User should be able to search for	New	1

SPRINTS

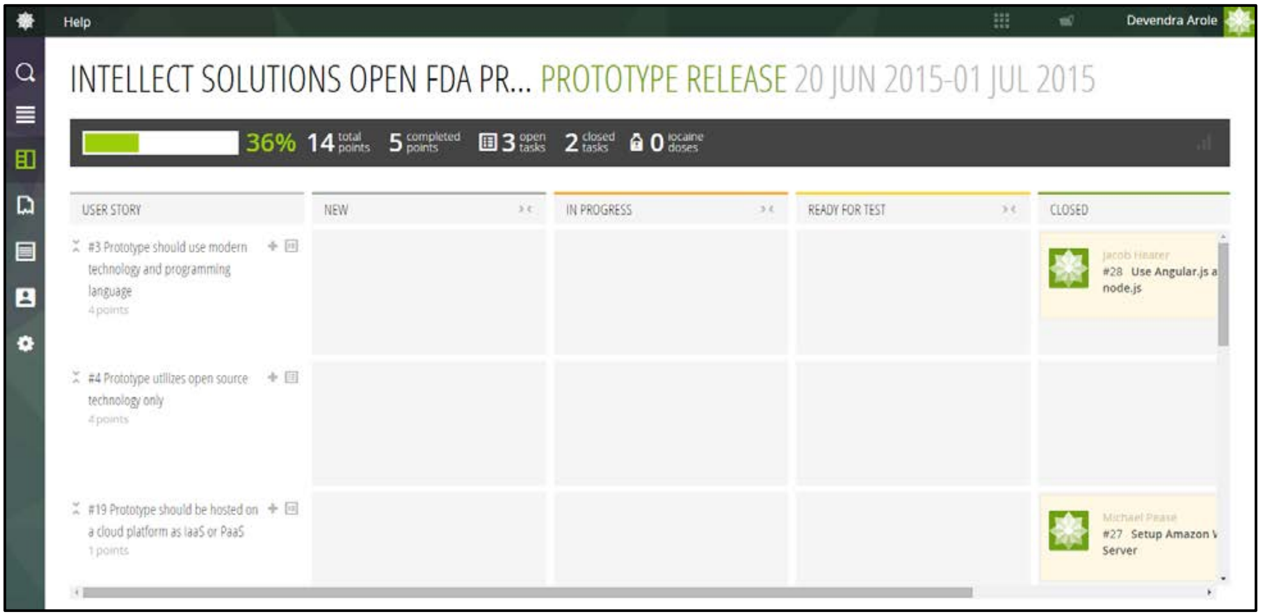
1 sprints

Prototype Release
20 Jun 2015-01 Jul 2015 5closed 14total

#3 Prototype should use modern t...	4
#4 Prototype utilizes open source t...	4
#19 Prototype should be hosted on...	1
#1 Front-end should be 508 compli...	2
#7 User should be able to search fo...	1
#25 Users should be able to provid...	
#10 User should be able view searc...	2

Enhancements
02 Jul 2015-16 Jul 2015 0closed 0total

Figure 5: Agile Sprints



INTELLECT SOLUTIONS OPEN FDA PR... PROTOTYPE RELEASE 20 JUN 2015-01 JUL 2015

36% 14 total points 5 completed points 3 open tasks 2 closed tasks 0 iocaine doses

USER STORY	NEW	IN PROGRESS	READY FOR TEST	CLOSED
#3 Prototype should use modern technology and programming language 4 points				Jacob Hearer #28 Use Angular.js a node.js
#4 Prototype utilizes open source technology only 4 points				
#19 Prototype should be hosted on a cloud platform as IaaS or PaaS 1 points				Michael Pease #27 Setup Amazon V Server

Figure 6: User Stories Developed in Taiga.Io

The work was based around software releases at the end of each sprint. Initial release focused on developing an alpha prototype in the short time frame requested by the government. We also identified subsequent releases to address feedback from unit testing and end user testing. We also identified some nice to have features that included functionality enhancements and UI improvements.

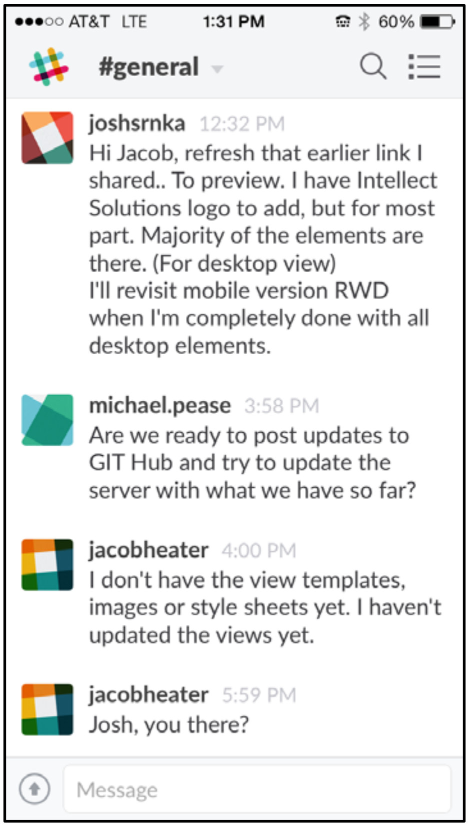
Intellect Solutions team identified a diverse team with varied experience to develop the prototype. Team members that supported the development of prototype represented the following labor categories under the ***development pool 2***:

- Technical Architect
- Front End Developer
- Back End Developer
- DevOps Engineer

The Technical Architect played the role of the leader who was responsible for setting the tone of the engagement, execution, and overall quality. This team came together, shared their experience, reviewed the RFQ requirements, and quickly developed a prototype (Figures 7 through 9).

Figure 7: Design Sessions

Figure 8: Design Sessions



joshsrnka 12:32 PM
Hi Jacob, refresh that earlier link I shared.. To preview. I have Intellect Solutions logo to add, but for most part. Majority of the elements are there. (For desktop view)
I'll revisit mobile version RWD when I'm completely done with all desktop elements.

michael.pease 3:58 PM
Are we ready to post updates to GIT Hub and try to update the server with what we have so far?

jacobheater 4:00 PM
I don't have the view templates, images or style sheets yet. I haven't updated the views yet.

jacobheater 5:59 PM
Josh, you there?

Figure 9: Team Collaboration Using Slack Tool

8.0 Testing

We performed the following type of testing as part of our methodology:

1. Functionality – We used JSFilddle.Net to create test AJAX scripts for submitting and validating Node.JS functionality to query and return data from OpenFDA. Additionally, we demonstrated the working prototype to the Intellect team members and invited immediate critique (Figures 10 and 11).

Figure 10: Demo to the Internal Team

Figure 11: Feedback from Internal Team Members

2. Automation – We wrote our own script using curl to visit each page on site and download content. The resulting output files are checked for any 404 errors.

3. Performance – We used review logs to monitor query and page response times then adjusted code and configurations based on log analysis results.

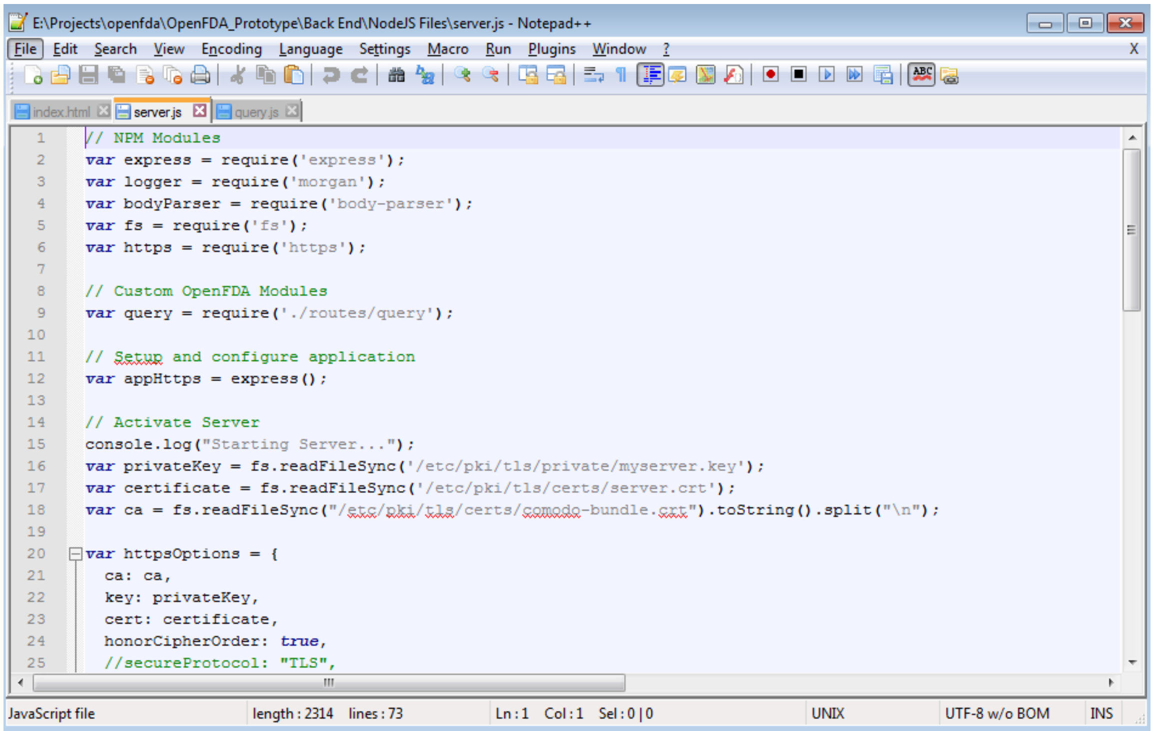
4. Security (through vulnerability management and penetration testing) – We used Kali Linux (Figure 12) and Open VAS platform tools to test exposed services for vulnerabilities; SSL Labs and Comodo for validating SSL configurations.

Kali Linux [Running] - Oracle VM VirtualBox
Machine View Devices Help
Applications Places
Sat Jun 27, 10:47 AM
root
Computer
KALI LINUX™
"the quieter you become, the more you are able to hear"
Right Ctrl

Figure 12: Kali Linux Penetration Testing Tool

5. 508 Testing – We used Firebug and FireEye per recommendation by GSA http://www.gsa.gov/portal/content/103534

6. Unit Testing – We used text editors (Figure 13) to edit code files; preliminary functional testing was performed on local web services running on development workstations; resulting updates were submitted to the Github Repository.

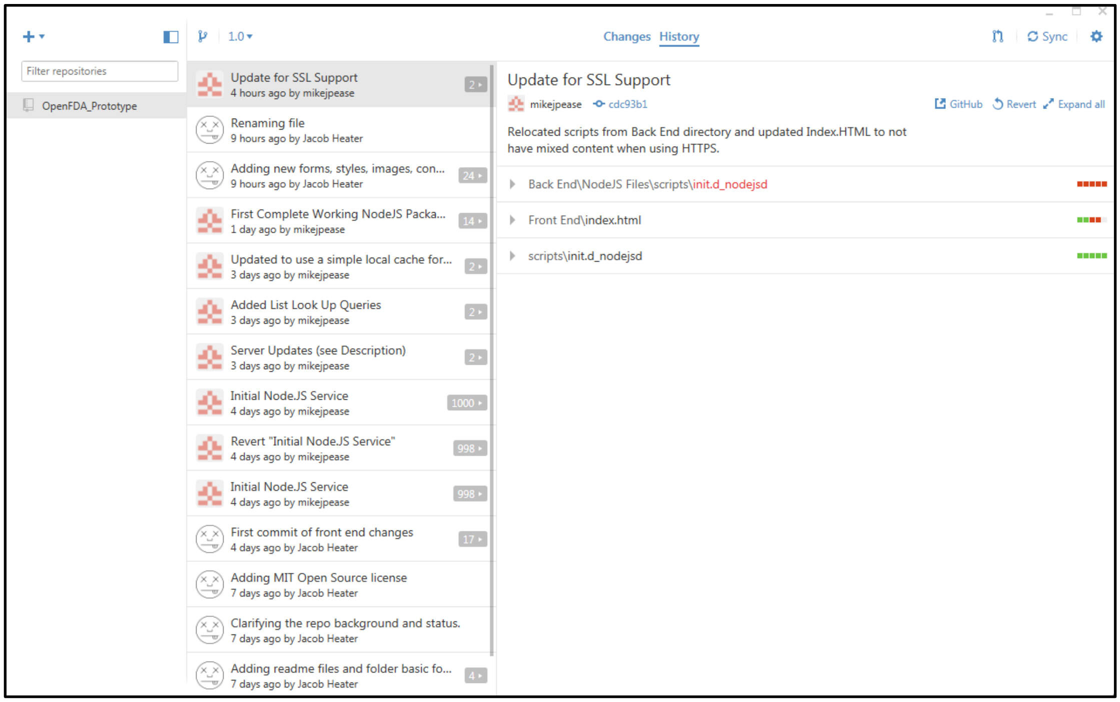

E:\Projects\openfda\OpenFDA_Prototype\Back End\NodeJS Files\server.js - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
index.html
server.js
query.js
JavaScript file
length : 2314 lines : 73
Ln : 1 Col : 1 Sel : 0 | 0
UNIX
UTF-8 w/o BOM
INS

```
// NPM Modules
var express = require('express');
var logger = require('morgan');
var bodyParser = require('body-parser');
var fs = require('fs');
var https = require('https');

// Custom OpenFDA Modules
var query = require('./routes/query');

// Setup and configure application
var appHttps = express();

// Activate Server
console.log("Starting Server...");
var privateKey = fs.readFileSync('/etc/pki/tls/private/myserver.key');
var certificate = fs.readFileSync('/etc/pki/tls/certs/server.crt');
var ca = fs.readFileSync("/etc/pki/tls/certs/comodo-bundle.crt").toString().split("\n");

var httpsOptions = {
  ca: ca,
  key: privateKey,
  cert: certificate,
  honorCipherOrder: true,
  //secureProtocol: "TLS",
```

Figure 13: Using Notepad++ to Edit Files

9.0 Configuration Management and Source Code Deployment

- Taiga.Io integrates with Github to check-in, check-out source code files, and for collaboration (Figure 14).
- Automated scripts are manually executed to update the server with the latest versions of the front and back end files.
- Wrote custom scripts to automate the Docker build and running commands.



Update for SSL Support
4 hours ago by mikejpease 2

Renaming file
9 hours ago by Jacob Heater

Adding new forms, styles, images, con...
9 hours ago by Jacob Heater 24

First Complete Working NodeJS Packa...
1 day ago by mikejpease 14

Updated to use a simple local cache for...
3 days ago by mikejpease 2

Added List Look Up Queries
3 days ago by mikejpease 2

Server Updates (see Description)
3 days ago by mikejpease 2

Initial Node.JS Service
4 days ago by mikejpease 1000

Revert "Initial Node.JS Service"
4 days ago by mikejpease 998

Initial Node.JS Service
4 days ago by mikejpease 998

First commit of front end changes
4 days ago by Jacob Heater 17

Adding MIT Open Source license
7 days ago by Jacob Heater

Clarifying the repo background and status.
7 days ago by Jacob Heater

Adding readme files and folder basic fo...
7 days ago by Jacob Heater 4

Update for SSL Support
mikejpease cdc93b1

Relocated scripts from Back End directory and updated Index.HTML to not have mixed content when using HTTPS.

Back End\NodeJS Files\scripts\init.d_nodejsd
Front End\index.html
scripts\init.d_nodejsd

Figure 14: Synchronizing Changes to GitHub Repository

10.0 System Security and Ongoing Maintenance

This section discusses approach to the following plays:

- *Play 11: Manage security and privacy through reusable processes*
- *Play 12: Use data to drive decisions*

Security and privacy played an important role in our approach. Intellect designed its prototype by placing huge emphasis on security up front unlike traditional software development projects where security is an afterthought or nice to have.

Our prototype is deployed on Amazon Web Services hosting provider which is FedRAMP certified - http://aws.amazon.com/compliance/fedramp-faqs/ (Figure 15). All the connections made to our website used HTTPS TLS v1.0, v1.1, and v1.2 with Cipher modes that place preferences on TLSv1.2 and only used the TLSv1.0 and 1.1 for backward compatibility. Additionally, we used OpenVAS to perform weekly vulnerability scans and addressed the critical and high vulnerabilities exploitable from the internet within 15 days.

During the initial architecture evaluation and design security review, we noted that while the OpenFDA site allowed HTTPS for secure data transmission, the service only supported GET requests. As a result, a risk was identified that internet service providers (ISP's) and other man-in-the-middle situations could obtain information regarding users of the service by allowing them to capture the query parameters and identify drugs, devices, or other medical information that users researched. Our architecture attempts to mitigate this vulnerability used Node.JS to provide a server side query capability. Encrypted connections to the ADERS server platform were created for the requestor to send the query request to OpenFDA and then return the results to the requestor. While this did not eliminate the vulnerability, it anonymized the connection by preventing the attribution of the query parameters to the actual individual. To further protect the user, the web services were not configured to capture query parameters with requesting IP addresses to ensure the anonymity of the user query requests to OpenFDA.

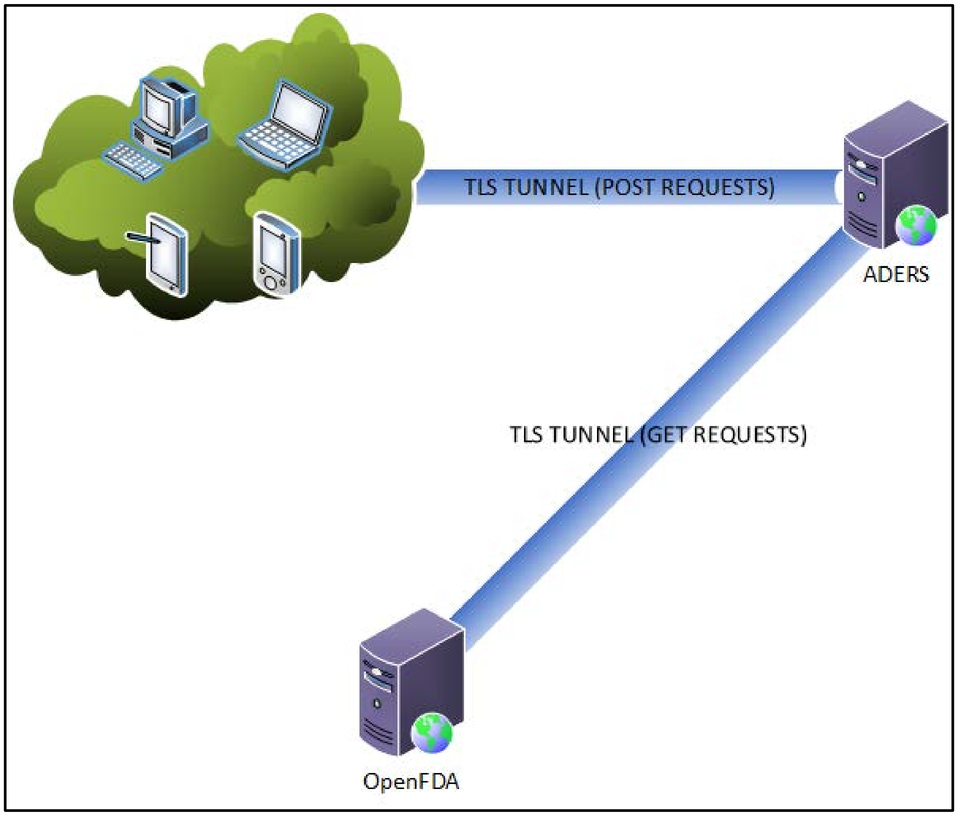
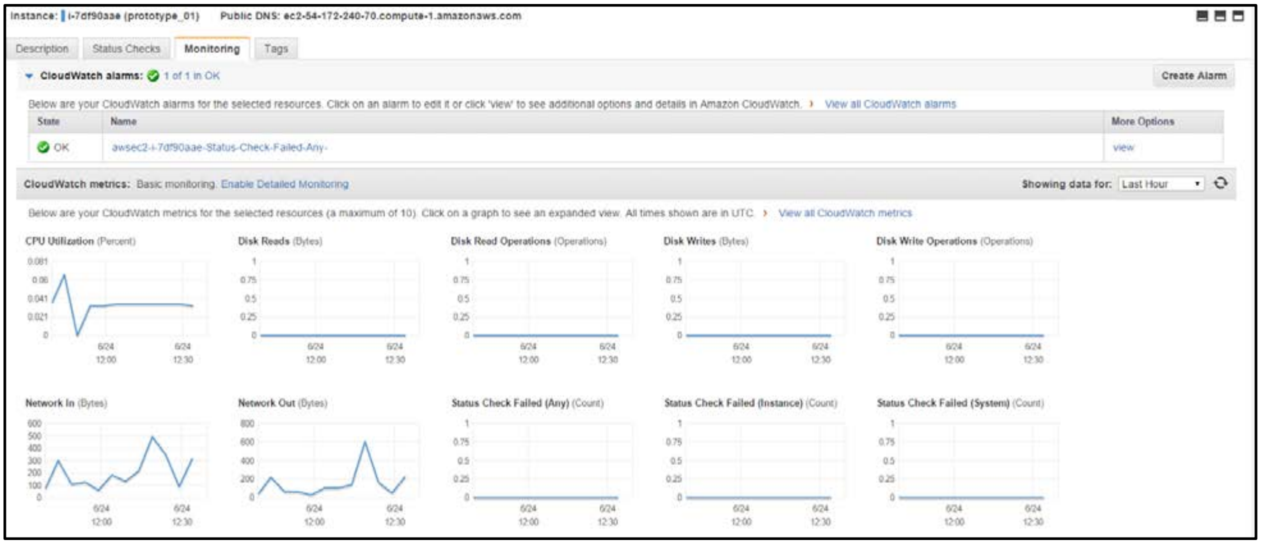


Figure 15: High-Level Architecture Diagram

As part of our commitment to maintaining privacy of user information, we only captured information necessary to meet functionality of the system. Apache captured IP addresses as part of normal website logging for statistical, performance, and security monitoring.

AWS provides system level continuous monitoring to assist in data driven decisions (Figure 16). This helps us setup alerts, monitor system performance, and make necessary adjustments to the prototype behavior in a timely manner.



Instance: i-7df90aae (prototype_01) Public DNS: ec2-54-172-240-70.compute-1.amazonaws.com

CloudWatch alarms: 1 of 1 in OK

Below are your CloudWatch alarms for the selected resources. Click on an alarm to edit it or click 'view' to see additional options and details in Amazon CloudWatch. › View all CloudWatch alarms

State	Name	More Options
OK	awsec2-i-7df90aae-Status-Check-Failed-Any-	view

CloudWatch metrics: Basic monitoring. Enable Detailed Monitoring

Showing data for: Last Hour

Below are your CloudWatch metrics for the selected resources (a maximum of 10). Click on a graph to see an expanded view. All times shown are in UTC. › View all CloudWatch metrics

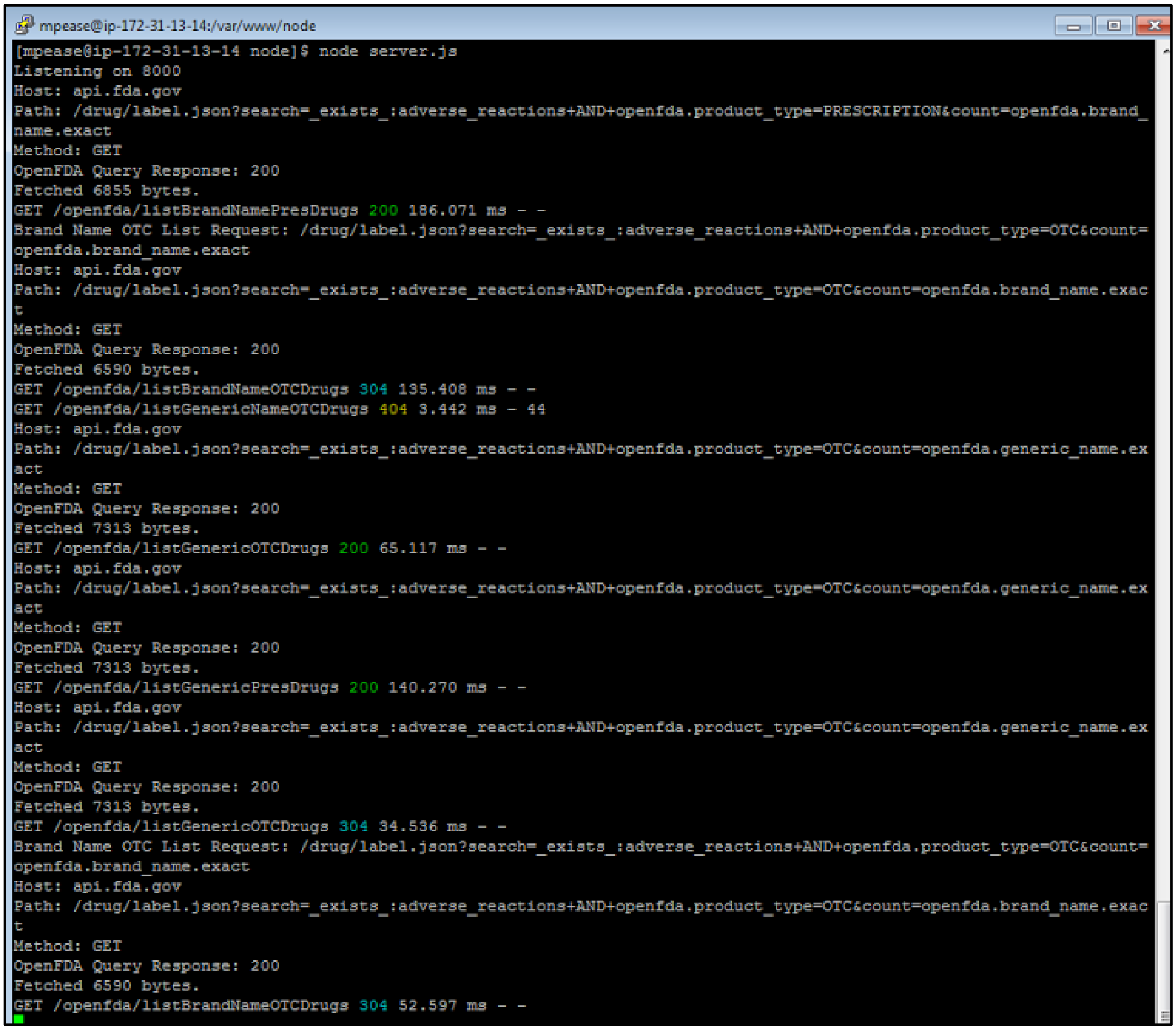
CPU Utilization (Percent)
0.081
0.06
0.041
0.021
0
Disk Reads (Bytes)
Disk Read Operations (Operations)
Disk Writes (Bytes)
Disk Write Operations (Operations)
1
0.75
0.5
0.25
0
6/24 12:00
6/24 12:30
Network In (Bytes)
600
500
400
300
200
100
0
Network Out (Bytes)
800
600
400
200
0
Status Check Failed (Any) (Count)
Status Check Failed (Instance) (Count)
Status Check Failed (System) (Count)

Figure 16: AWS Performance Monitoring Features

During development, performance monitoring for query response and page displays were analyzed as per normal functional testing and evaluation. During the test result review process and expected user interaction on the pages, the team identified that there are four queries required to display the drop down lists for users to select a drug. Those were:

- Generic Drug List of Over the Counter Medication
- Generic Drug List for Prescription Medication
- Brand Name Drug List of Over the Counter Medication
- Brand Name Drug List for Prescription Medication

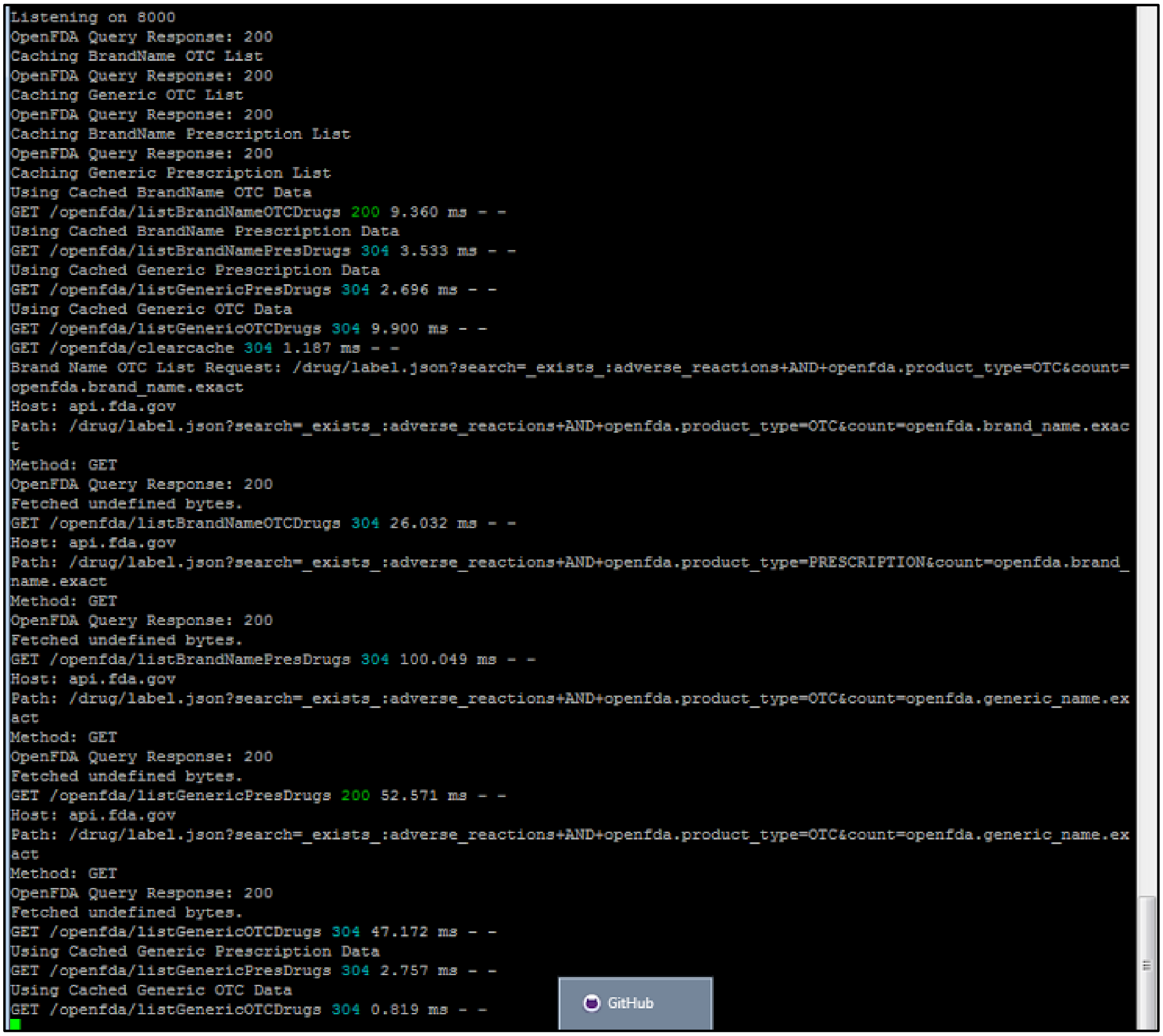
During initial testing, the OpenFDA query response times ranged from 65 ms to 186 ms. Based on this information, the team concluded that these queries for multiple users and multiple times per user could be cached on the server to reduce the number of queries submitted to OpenFDA and improve the response time for page displays (Figure 17).



```
[mpease@ip-172-31-13-14 node]$ node server.js
Listening on 8000
Host: api.fda.gov
Path: /drug/label.json?search=_exists_:adverse_reactions+AND+openfda.product_type=PRESCRIPTION&count=openfda.brand_
name.exact
Method: GET
OpenFDA Query Response: 200
Fetched 6855 bytes.
GET /openfda/listBrandNamePresDrugs 200 186.071 ms - -
Brand Name OTC List Request: /drug/label.json?search=_exists_:adverse_reactions+AND+openfda.product_type=OTC&count=
openfda.brand_name.exact
Host: api.fda.gov
Path: /drug/label.json?search=_exists_:adverse_reactions+AND+openfda.product_type=OTC&count=openfda.brand_name.exac
t
Method: GET
OpenFDA Query Response: 200
Fetched 6590 bytes.
GET /openfda/listBrandNameOTCDrugs 304 135.408 ms - -
GET /openfda/listGenericNameOTCDrugs 404 3.442 ms - 44
Host: api.fda.gov
Path: /drug/label.json?search=_exists_:adverse_reactions+AND+openfda.product_type=OTC&count=openfda.generic_name.ex
act
Method: GET
OpenFDA Query Response: 200
Fetched 7313 bytes.
GET /openfda/listGenericOTCDrugs 200 65.117 ms - -
Host: api.fda.gov
Path: /drug/label.json?search=_exists_:adverse_reactions+AND+openfda.product_type=OTC&count=openfda.generic_name.ex
act
Method: GET
OpenFDA Query Response: 200
Fetched 7313 bytes.
GET /openfda/listGenericPresDrugs 200 140.270 ms - -
Host: api.fda.gov
Path: /drug/label.json?search=_exists_:adverse_reactions+AND+openfda.product_type=OTC&count=openfda.generic_name.ex
act
Method: GET
OpenFDA Query Response: 200
Fetched 7313 bytes.
GET /openfda/listGenericOTCDrugs 304 34.536 ms - -
Brand Name OTC List Request: /drug/label.json?search=_exists_:adverse_reactions+AND+openfda.product_type=OTC&count=
openfda.brand_name.exact
Host: api.fda.gov
Path: /drug/label.json?search=_exists_:adverse_reactions+AND+openfda.product_type=OTC&count=openfda.brand_name.exac
t
Method: GET
OpenFDA Query Response: 200
Fetched 6590 bytes.
GET /openfda/listBrandNameOTCDrugs 304 52.597 ms - -
```

Figure 17: Performance before Optimization

Once the caching capability was added to the service, the response times dropped to 2-9ms (Figure 18).



```
Listening on 8000
OpenFDA Query Response: 200
Caching BrandName OTC List
OpenFDA Query Response: 200
Caching Generic OTC List
OpenFDA Query Response: 200
Caching BrandName Prescription List
OpenFDA Query Response: 200
Caching Generic Prescription List
Using Cached BrandName OTC Data
GET /openfda/listBrandNameOTCDrugs 200 9.360 ms - -
Using Cached BrandName Prescription Data
GET /openfda/listBrandNamePresDrugs 304 3.533 ms - -
Using Cached Generic Prescription Data
GET /openfda/listGenericPresDrugs 304 2.696 ms - -
Using Cached Generic OTC Data
GET /openfda/listGenericOTCDrugs 304 9.900 ms - -
GET /openfda/clearcache 304 1.187 ms - -
Brand Name OTC List Request: /drug/label.json?search=_exists_:adverse_reactions+AND+openfda.product_type=OTC&count=
openfda.brand_name.exact
Host: api.fda.gov
Path: /drug/label.json?search=_exists_:adverse_reactions+AND+openfda.product_type=OTC&count=openfda.brand_name.exac
t
Method: GET
OpenFDA Query Response: 200
Fetched undefined bytes.
GET /openfda/listBrandNameOTCDrugs 304 26.032 ms - -
Host: api.fda.gov
Path: /drug/label.json?search=_exists_:adverse_reactions+AND+openfda.product_type=PRESCRIPTION&count=openfda.brand_
name.exact
Method: GET
OpenFDA Query Response: 200
Fetched undefined bytes.
GET /openfda/listBrandNamePresDrugs 304 100.049 ms - -
Host: api.fda.gov
Path: /drug/label.json?search=_exists_:adverse_reactions+AND+openfda.product_type=OTC&count=openfda.generic_name.ex
act
Method: GET
OpenFDA Query Response: 200
Fetched undefined bytes.
GET /openfda/listGenericPresDrugs 200 52.571 ms - -
Host: api.fda.gov
Path: /drug/label.json?search=_exists_:adverse_reactions+AND+openfda.product_type=OTC&count=openfda.generic_name.ex
act
Method: GET
OpenFDA Query Response: 200
Fetched undefined bytes.
GET /openfda/listGenericOTCDrugs 304 47.172 ms - -
Using Cached Generic Prescription Data
GET /openfda/listGenericPresDrugs 304 2.757 ms - -
Using Cached Generic OTC Data
GET /openfda/listGenericOTCDrugs 304 0.819 ms - -
```

Figure 18: Performance after optimization

11.0 Future Enhancements

One of the future enhancements planned is to analyze the location of the traffic to determine load balancing requirements. AWS provides load balancing services at its servers across the Continental US and internal locations in Europe, Asia, etc. Once set up appropriately, the load balancer automatically re-distributes connections across multiple locations and instances. We plan to leverage the IP addresses geocoding feature to identify locations where additional AWS EC2 instances and load balancing would provide the most significant performance increase based on actual user activity.