# 

**NASA ISS FIT Administration Tool Architecture and**

**Deployment Guide**

|  |  |  |
| --- | --- | --- |
| **Author** | **Revision Number** | **Date** |
| pvmagacho | 1.0 | 04/17/2015 |
| pvmagacho | 1.1 | 04/29/2015 |
| pvmagacho | 1.2 | 06/03/2015 |
| pvmagacho | 1.3 | 08/07/2015 |
| pvmagacho | 1.4 | 08/13/2015 |
| pvmagacho | 1.5 | 11/19/2015 |
| pvmagacho | 1.6 | 06/06/2016 |
| pvmagacho | 1.7 | 12/09/2016 |

[Architecture 3](#_Toc452973425)

[1. Overview 3](#_Toc452973426)

[1.1 Application Requirements 3](#_Toc452973427)

[1.2 Application Overview 4](#_Toc452973428)

[1.3 Application Structure 4](#_Toc452973429)

[1.4 Application Configuration 4](#_Toc452973430)

[Deployment Guide 5](#_Toc452973431)

[2. Application Deployment 5](#_Toc452973432)

[3. Application Verification 9](#_Toc452973433)

# Architecture

# Overview

The customer of this project is the NASA Nutritional Biochemistry Laboratory working on tracking dietary intake of astronauts on the International Space Station (ISS). The NASA Nutritional Biochemistry Laboratory collects the data on type and quantity of food eaten by astronauts on the ISS, and uses these for medical reporting as well as for scientific research.

NASA has been using a special Food Frequency Questionnaire (FFQ) for tracking weekly consumption of food by astronauts. It is a long list of food products grouped by several categories according to their nutrient content. Each astronaut filled in that questionnaire with the quantity of food packages that he/she consumed during the week. The FFQ approach allowed astronauts to collect dietary intake data for a week.

The Laboratory performs offline analysis of that data found in the database, determines nutrient components consumed by each astronaut, and searches for various relationships between food diet data and physical conditions/performance of space crew. A prompt feedback on the calculated results can be given to the Flight Surgeon and the crew member (this feature is out of scope for the application, but is the ultimate goal of the data coming from the application).

The main goal of this project is to deliver a solution for efficient collection of food consumption information by astronauts on the ISS. It should provide precise, fast, and easy to use tracking ability for every day usage in micro-gravity conditions on the ISS. Convenient features for user identification, preparing food lists, various ways to input information and data synchronization with the Earth laboratory are also important parts of this project.

This module implements the administration tool, which will control the information found in the database and generate reports.

## Application Requirements

* Debian 6 (Squeeze) Linux
* Node.js
* Loopback
* Jade
* JQuery
* JQuery Datatables

## Application Overview

The IFIT Admin Tool is built using NodeJs and Loopback to handle incoming HTTP calls. Loopback handle both the incoming calls for the Admin Tool and serve also as a REST service to the iPad application.

Loopback is a model based NodeJs application. Data is retrieved from a postgresql database installed in the same computer running the admin tool.

Jade templates are used to render the retrieved information from the database as HTML pages.

Reports are generated from the tool and can have information from one or many users. The output file is an excel file with the food consumption information.

To generate the reports python scripts are used to connect to the Postgresql database and retrieve all needed information.

## Application Structure

* **Backend Services/issfit-api/** - folder with the main loopback application and scripts to load data and generate report
* **Backend Services/issfit-api/server** - folder with the server files, to handle both Admin Tool and REST server
* **Backend Services/issfit-api/database/** - folder with the database schema
* **Backend Services/issfit-api/client/public/** - folder with images, css and javascript files used by web pages
* **Backend Services/issfit-api/client/views/** - folder with Jade templates
* **Backend Services/issfit-api/reports/** - output folder for reports generated from the tool
* **Backend Services/issfit-api/csv bulk upload templates/** - folder with CSV template files to be used with bulk upload tool feature

## Application Configuration

The configuration is done inside the **Backend Services/issfit-api/server/config.js** in the variable **config**. The following default parameters are used

* config.useApache = false; (true to use Apache as a proxy)
* config.web.port = 4343; (9090 if useApache is true)
* config.web.key = '/etc/ssl/private/ssl-cert-snakeoil.key'; (SSL key – replace with your company Key)
* config.web.cert = '/etc/ssl/certs/ssl-cert-snakeoil.pem'; (SSL certificate – replace with your company Certificate)
* config.db.database = 'pl\_fit';
* config.db.username = 'pl\_fit\_db';
* config.db.password = 'CHANGEME';
* config.db.host = '127.0.0.1';
* config.db.port = 56283;

The python and shell scripts use the same configuration.

# Deployment Guide

# Application Deployment

The deployment should be made in a Debian 6 (Squeeze) linux distribution.

The following steps should be followed to setup the Admin Tool.

1. Install Debian 6 (Squeeze) linux – **skip if already installed**. The link to download the distro is <https://www.debian.org/releases/squeeze/debian-installer>
2. Login as **root** user
3. Run the following commands in prompt

> apt-get -y install wget build-essential git-core curl openssl libssl-dev sudo

1. Install postgresql 9.2 – **skip if already installed**

> sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt/ squeeze-pgdg main" > /etc/apt/sources.list.d/pgdg.list'

> wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add -

> apt-get update

> apt-get upgrade

> apt-get -y install postgresql-9.2 libpq5 libpq-dev postgresql-contrib-9.2

1. Install nodejs

> git clone https://github.com/joyent/node.git

> cd node

> git checkout v0.10.29

> ./configure --openssl-libpath=/usr/lib/ssl

> make && make install

1. Install python dev

> apt-get -y install python-dev python-psycopg2

1. Configure postgresql database
   1. Open file /etc/postgresql/9.2/main/pg\_hba.conf and add the following line at the end (change **192.168.60.0/24** to match local network mask)

**hostssl pl\_fit pl\_fit\_db 192.168.60.0/24 md5**

* 1. Open file /etc/postgresql/9.2/main/postgresql.conf and modify the following line: (change **192.168.60.53** to the server LAN IP)

**listen\_addresses = '192.168.60.53' # what IP address(es) to listen on;**

**port = 56283 # (change requires restart)**

**ssl = on # (change requires restart)**

**ssl\_cert\_file = '/etc/ssl/certs/ssl-cert-snakeoil.pem' # (change requires restart)**

**ssl\_key\_file = '/etc/ssl/private/ssl-cert-snakeoil.key' # (change requires restart)**

* 1. Restart server

> service postgresql restart

1. Create NASA database

> su -l postgres

> createdb pl\_fit

> psql -d pl\_fit -p 56283

pl\_fit=# ALTER USER postgres WITH PASSWORD '12345';

pl\_fit=# CREATE USER pl\_fit\_db WITH NOSUPERUSER NOCREATEDB NOCREATEROLE ENCRYPTED PASSWORD 'CHANGEME';

pl\_fit=# \q

> exit

1. Configure Admin tool (**online machine**)

> npm -g install node-gyp

> git clone https://github.com/nasa/NTL-ISS-Food-Intake-Tracker.git

> cd NTL-ISS-Food-Intake-Tracker/Backend\ Services/issfit-api

> npm install

> cd database

> psql -h localhost -d pl\_fit -W -U postgres -p 56283 -f database\_schema.sql

> psql -h localhost -d pl\_fit -W -U postgres -p 56283 -f static-data.sql

(when prompt enter password: 12345)

> cd ../

1. Configure login credentials (9.0)

> su -l postgres

> cd /usr/share/postgresql/9.0/contrib

> psql -d pl\_fit -p 56283

pl\_fit=# \i pgcrypto.sql;

pl\_fit=# INSERT INTO users(username, email, pwdhash) values('pl\_fit\_db', 'pl\_fit\_db@email.com', crypt('CHANGEME', gen\_salt('bf')));

pl\_fit=# \q

> exit

1. Configure login credentials (9.2)

> su -l postgres

> psql -d pl\_fit -p 56283

pl\_fit=# CREATE EXTENSION pgcrypto;

pl\_fit=# INSERT INTO users(username, email, pwdhash) values('pl\_fit\_db', 'pl\_fit\_db@email.com', crypt('CHANGEME', gen\_salt('bf')));

pl\_fit=# \q

> exit

1. Start application

> node .

1. Configure IFIT App with Apache
   1. Stop node application
   2. Change config.useApache to true in config.js file
   3. Run the commands below as root

> sudo a2enmod proxy proxy\_http

> sudo a2enmod ssl

* 1. Create site file /etc/apache2/sites-available/fit.ssl with contents below:

<VirtualHost \*:443>

SSLEngine on

SSLProxyEngine on

SSLCertificateFile /etc/ssl/certs/ssl-cert-snakeoil.pem

SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key

ErrorLog /var/log/apache2/fit.error.log

TransferLog /var/log/apache2/fit.access.log

ProxyRequests off

ProxyPreserveHost on

ProxyPass / <http://localhost:3000/>

ProxyPassReverse / <http://localhost:3000/>

<Proxy \*>

Order deny,allow

Allow from all

AuthType Basic

AuthName "Private"

AuthBasicProvider file

AuthUserFile /etc/apache2/htpasswd

Require valid-user

</Proxy>

</VirtualHost>

* 1. Run the commands below as root

> a2ensite fit-ssl

> htpasswd -cb /etc/apache2/htpasswd pl\_fit\_db CHANGEME

> chmod a+r /etc/apache2/htpasswd

> service apache2 restart

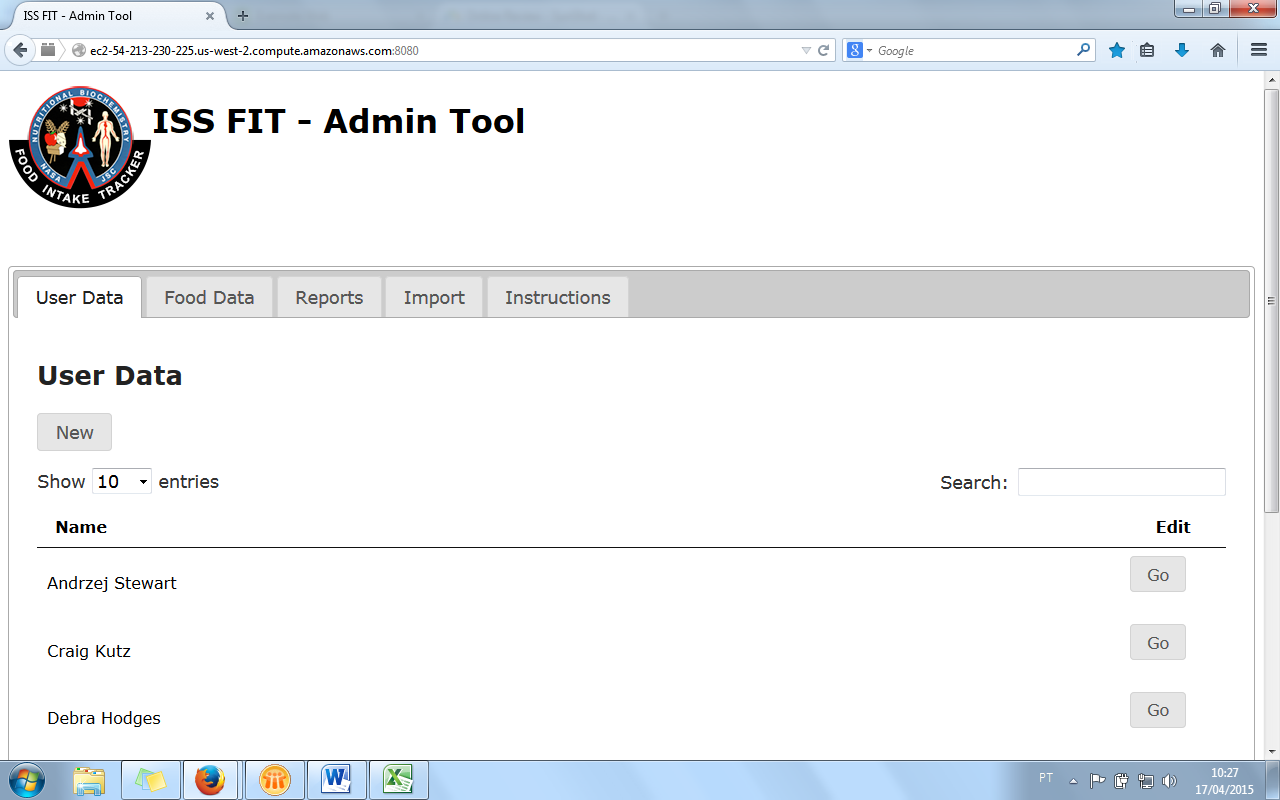
* 1. Start application

> node .

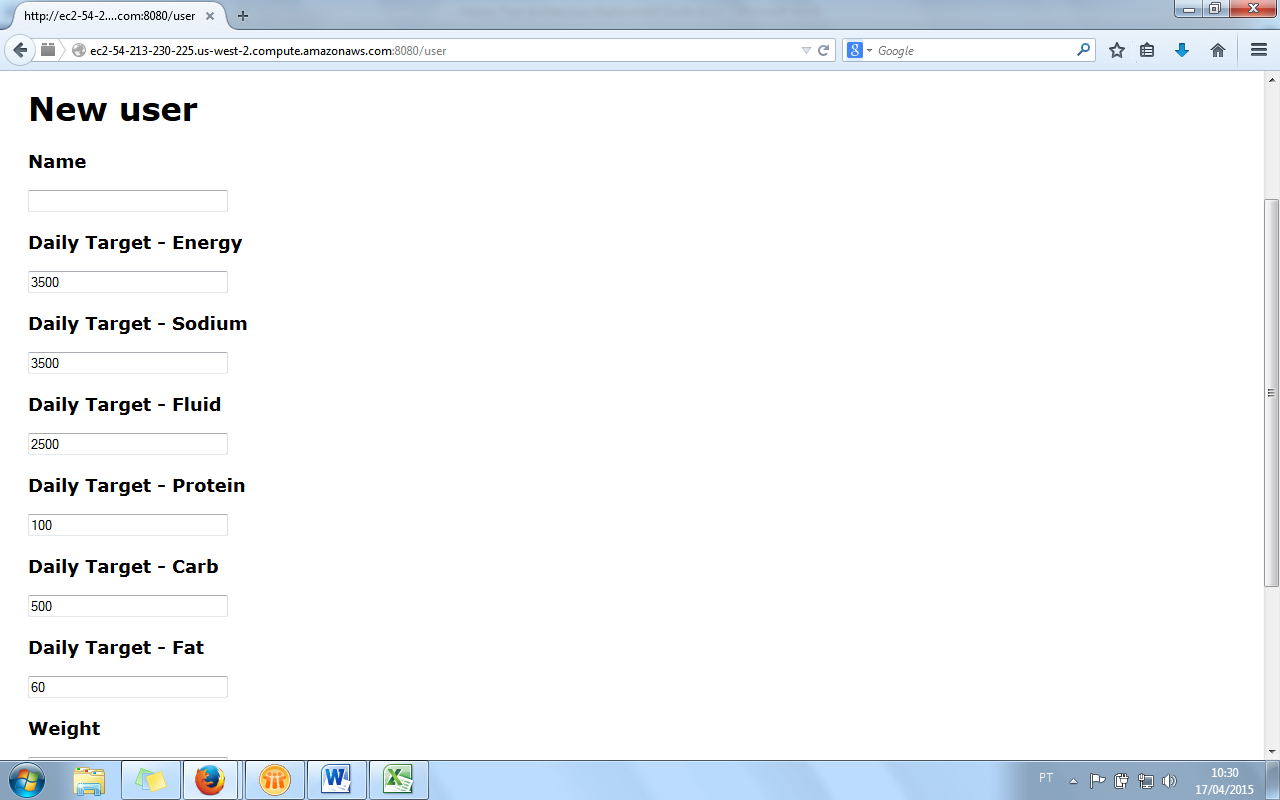
# Application Verification

To verify the application open the browser with the IP of the server and port 3000 or port 43 if using Apache as proxy.

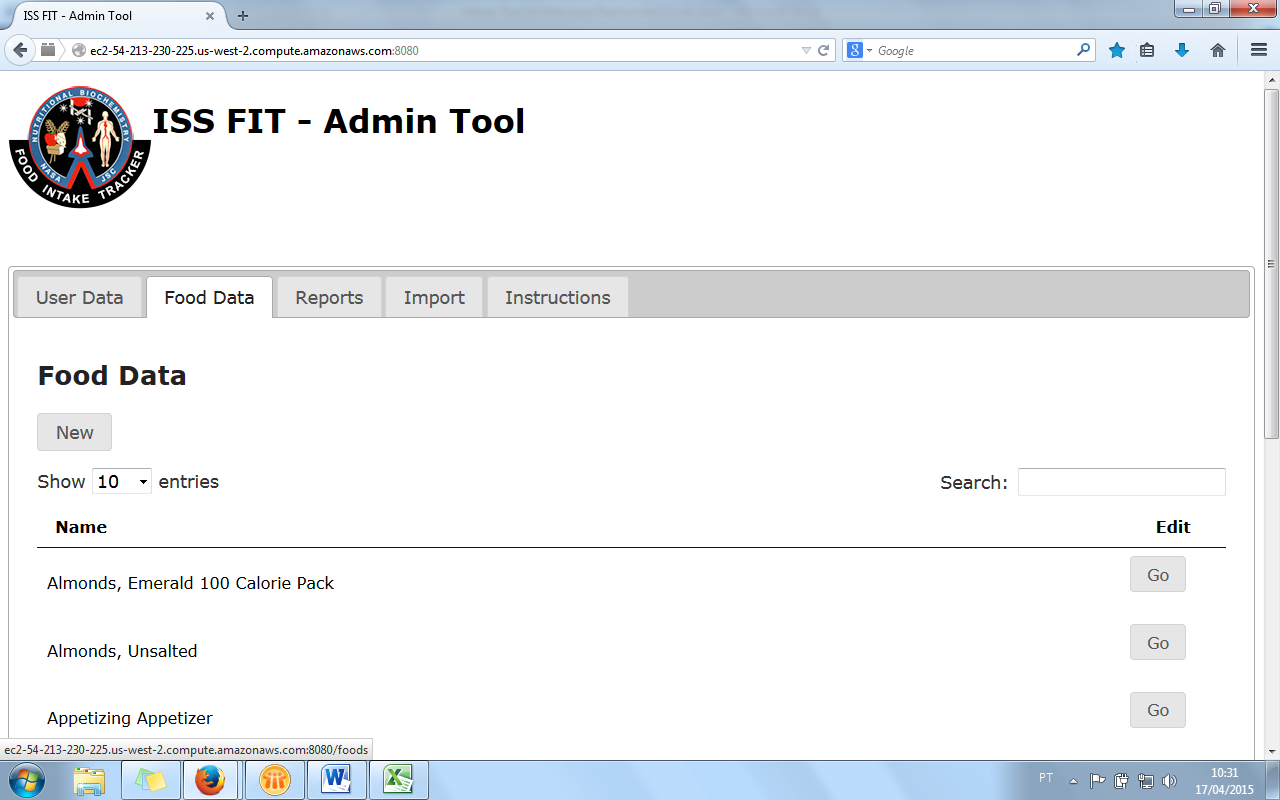
* http://{SERVER\_IP}:3000

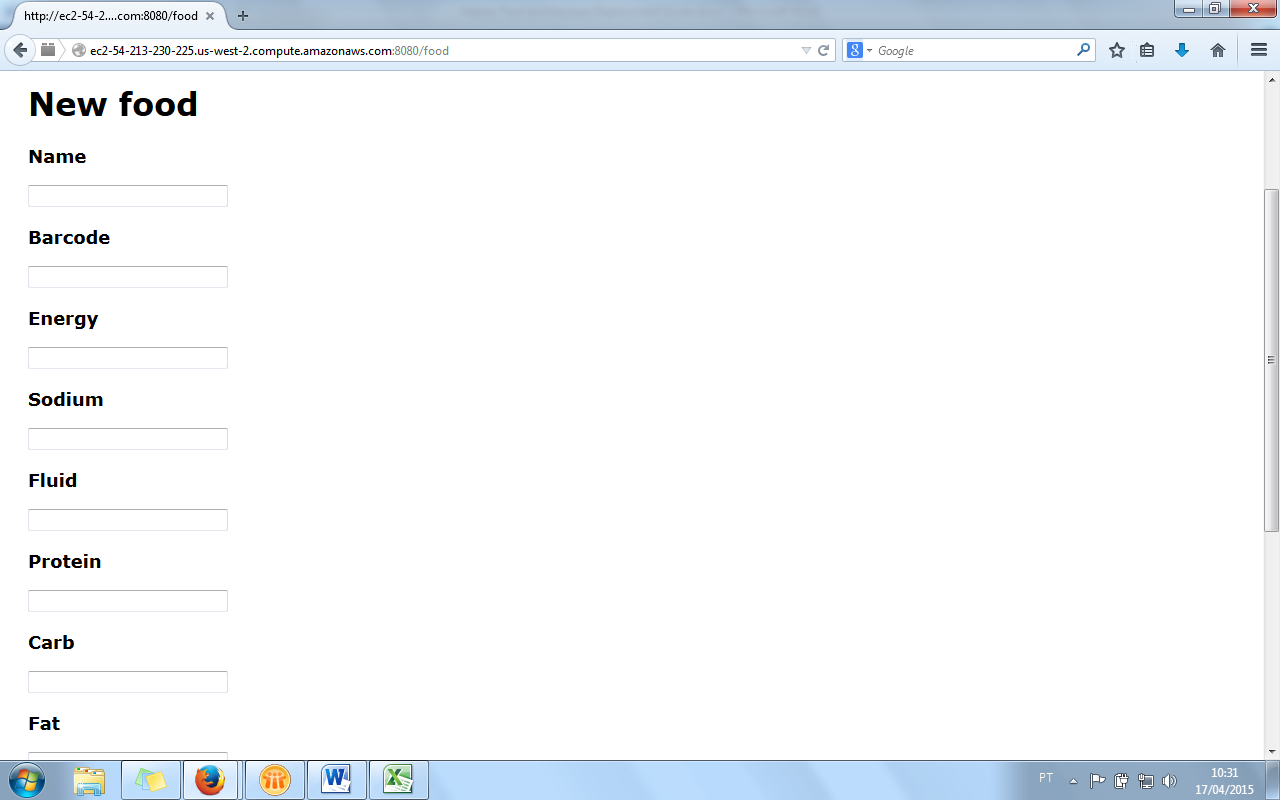


* Click New to add new user.

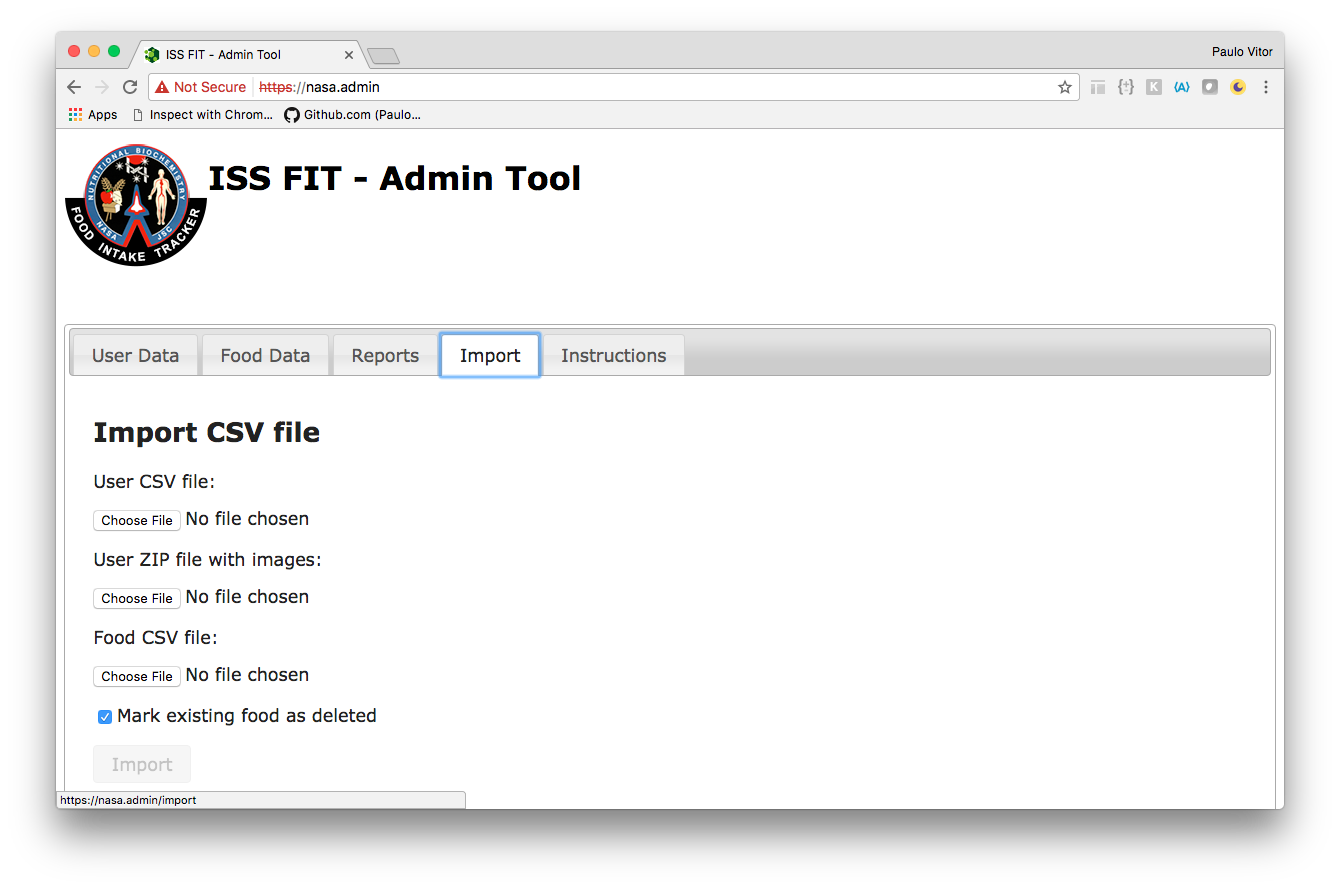


* Click “Food Data” tab. Click New to add new food.





* Click “Import” tab to load CSV files. Instructions can be found in “Instructions Tab”



* Click “Reports” tab to generate reports. Select users or all users’ option.

