# File: test_plan.txt
# Description: A test plan document for my implementation of the LOST Web Application

Step 1 - Create a fresh instance of the OSNAP VM base image.

Step 2 - Using the following two commands, clone the contents of my git repo into the image, then move them to the appropriate location.

```
        git clone https://github.com/zplunte/cis322.git ./gitrep
        cp -r gitrep/* ./
```

Step 3 - Initialize the database server with the next command which runs initlostdb.sh

```
        ./initlostdb.sh
```

Note: You should see some scrolling text that confirms the database server is being setup.

Step 4 - Create the LOST Web Application database with the following command

```
        createdb lost
```

Step 5 - Run preflight.sh with lost database to create tables and populate wsgi with src files. The command you should use is

```
        ./preflight.sh lost
```

Note: You should see a series of 'CREATE TABLE' messages echoed back after running this command.

Step 6 - Start the Apache web server with the following command

```
        apachectl start
```

Step 7 - Verify that the server is running by visiting localhost:8080 in a web browser.You should see a login page with the OSNAP insignia and a 'create user' button.

Step 8 - Create a user and explore the site. First create a Logistics Officer so that you can create facilities, create assets, dispose a few assets, and make a transfer request. You should also check the Asset Report page to verify that it works correctly. Then create a Facilities Officer so that you can approve transfer requests. You should then return to a Logistics Officer account and test the set load/unload times functionality.

Step 9 - Verify that the data you entered through your browser has been stored in the database by running the following commands and looking for the entries in the psql database.

```
psql lost
select * from users;
select * from facilities;
select * from assets;
select * from transfers;
```

Step 10 - Test the export functionality by navigating into the export directory and running export_data.sh with the appropriate arguments. The commands you should use are

```
cd export/
./export_data.sh lost test_export
```

Note: This should create the directory test_export/ in the directory export/ and it should now be populated with appropriate .csv files. Navigate into test_export/ and verify the contents of the .csv files with the following commands

```
cd test_export/
ls
more users.csv
more facilities.csv
more assets.csv
more transfers.csv
```

Step 11 - Navigate back to the main directory with this command

```
cd ../..
```

Step 12 - Drop this database and then recreate a clean version so that we can test our import scripts. Use the following commands to get a clean version of the database.

```
apachectl stop
dropdb lost
createdb lost
./preflight.sh lost
apachectl start
```

Note: After running ./preflight.sh lost you should see the same list of 'CREATE TABLE' statements as before.

Step 13 - Test the import functionality by navigating into the import directory and running test_import.sh with the appropriate arguments. The commands you should use are

```
cd import/
```

```
./test_import.sh lost
```

Note: You should see a scrolling series of echo messages displaying the progress of the import. These should terminate quickly with no error, returning you to a command line prompt.

Step 14 - Verify that the import step worked correctly with the following commands

```
psql lost
select * from users;
select * from facilities;
select * from assets;
select * from transfers;
```

# That's it for now! Hope everything works.