Documentation: Team Orange

“A fertile soil alone does not carry agriculture to perfection.”– E. H. Derby

High Level Overview

Goal: Our application allows for seamless end to end tracking of a seed from the seed grower to the farmer.

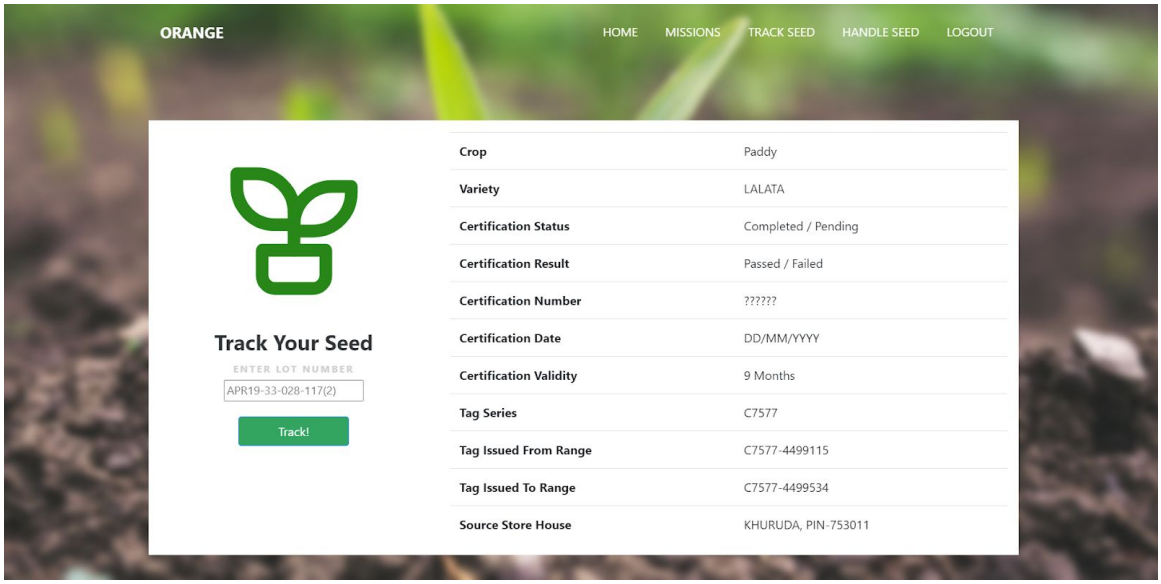
- Our architecture consists of the following
    - **Frontend**: Login, register pages for all organizations involved. Data entry pages for SPA, STL and SCA.
    - **Backend**: Communicates between the frontend, SQL databases and blockchain for data storage.
    - **Databases**: SQL and NoSQL databases for storing login information and any extra seed information which is not required to be stored on the blockchain.
    - **Blockchain**: Stores and tracks all seed information along with the history.
    - **Security and transparency**: The blockchain cannot be tampered with and the entire history of changes is available. Only the required information is shown to the organizations using blockchain channels.
    - **Analytics & Prediction:** Predict features like fertility status using nutrient index.
    - **Feedback & Trust Scoring**: Collect feedback from the farmer. Based on the feedback of many farmers, we calculate a score.

User Interface

A single web app is available for all types of users. Based on the type of user signed in the web app displays the permitted options. The types of users are:

1. Farmer / Seed Grower
2. SPA (Seed Processing Agency)
3. SPP (Seed Processing Plant)
4. SCA (Seed Certification Agency)
5. STL (Seed Testing Lab)
6. WareHouse / Distributor

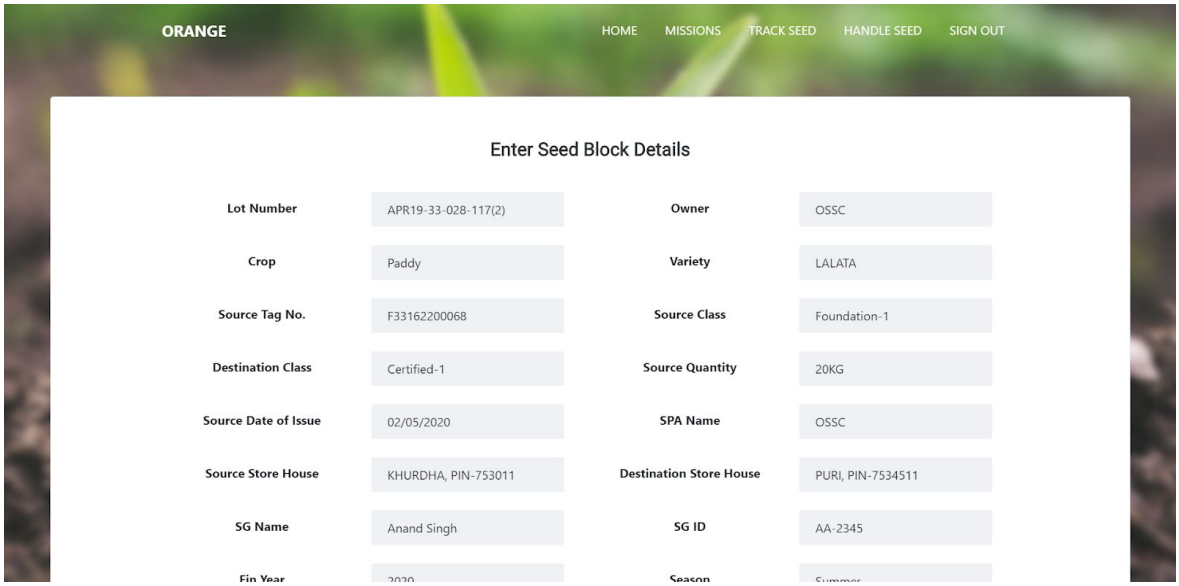
A farmer and a seed grower have similar UI as both have to to track the seed lots related to them. They have to register as a farmer and login before accessing the “TRACK SEED” tool.



The “TRACK SEED” tool allows any user to track the see the details of the seed lot using its “Tag Number”.

All agencies need to register as an Agency of their respective type so that they can access their respective forms and tools. All agencies are given an authorization code so that they can register on the website. Registration is not completed if a wrong authorization is given.

Based on the type of Agency signed in, the web app displays the respective form so that it can create/update a seed block.
Only an SPA is allowed to create a seed block. All other agencies are required to update the seed block using its Lot Number which is unique to each lot.



Cloud database & Backend

Setting Up

Since our Web App is running with a Flask(Python) Backend we needed an interface to communicate with the oracle cloud service,
NoSQL Database Python SDK was used for this purpose, the Steps
taken to setup the communication between the server and cloud were done as given in this Documentation -
https://nosql-python-sdk.readthedocs.io/en/latest/installation.html#connecting-an-application

It requires a python package called ‘borneo’ to be installed and a .py file to be created which will provide us with a ‘handle’ which will be used to communicate with the server. We also need to install a package called ‘oci’ which is used to handle all the authentication. We can either keep all the Cloud Account Credentials in the Python file we created earlier or we can keep it in ~/.oci directory and configure the python file accordingly.
RSA keys also need to be generated and the public key needs to be given in the API Section of the Cloud interface.
The process of handling the Credentials, Private,Public Keys and FIngerprint is given here -
https://docs.cloud.oracle.com/en-us/iaas/Content/API/Concepts/apisigningkey.htm

After the setup is complete, we can import the handle created into our Flask Code and call the required Functions to interact with the Database.
The Commands to be used to interact with Tables is given here - https://nosql-python-sdk.readthedocs.io/en/latest/tables.html

Running the backend Flask server

- Clone the repository on the VM (this step is not required in the currently deployed VM)
- `cd Flask-Data-From-HTML-Form/app` to move to the flask app's directory
- Set the flask app name using `export FLASK_APP=app`
- (optional) create a new virtualenv if not already present
- Install all the requirements using: `pip3 install -r ../requirements.txt`
- Create a .env file in the same folder with the following contents
  AUTH=<insert oracle blockchain API basic auth here (base64 encoded)>
  SPA_URL=<URL of "/bcsgw/rest/v1/transaction/invocation" for the main organisation>
  - For example:
    AUTH="Basic YXl1c2g6R3BFaTZybXdjMiNOM1IlbnQofXU="
    SPA_URL="https://orange-atpapex-bom.blockchain.ocp.oraclecloud.com:7443/restproxy/bcsgw/rest/v1/transaction/invocation"
- Run the flask app using: `sudo flask run  --host=0.0.0.0 --port=5000`
- Type the VM's external IP address in the browser with port 5000. For example: http://152.67.0.239:5000/

Overview

- Oracle Cloud service is being used to manage the registration data and other important/extra details about the seeds.
- We are using a NoSQL database to store all the data since it is more flexible as the data we are storing can be very varying and we may not always have all the required fields.
- NoSQL databases are also famous for being more easily scalable and much more flexible when compared to SQL databases.

Technical Details

We are using the oracle python sdk because it would be easy to integrate with our flask server.
Follow the setup as mentioned in the official oracle docs.
There are three main tables:

1. One table holds all the registration details of all the users who register. Regardless of the type of user, the fact that we are using a nosql database allows us to append all details into this single table.
2. Second table holds complete details of all the seeds with TagNO as the primary key. We can accommodate any number of parameters in this table, as it has only 2 columns. One, being the primary key and the other being a json object containing all the details.
3. Third table contains details of all the reviews and feedback of all the farmers which also has LotNo as the primary key.

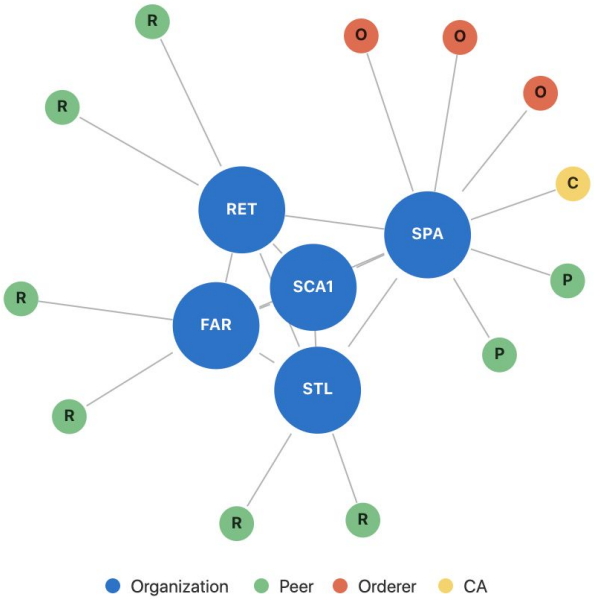
Blockchain

The blockchain is hosted on Oracle Cloud Blockchain platform which is based on the Hyperledger Fabric distributed ledger system.

The following Organizations were created in the Blockchain platform:

- SPA
- SCA

- STL
- FAR
- RET



Each of the organizations have 2 peers.

Every organization is connected by channels.
Each channel has a chaincode associated with it. The chaincode is implemented using the Hyperledger Fabric SDK written in the Go language.

The chain code is present under “Blockchain/seedBlock.go” in the repository.

Intelligence

- Soil Fertility: We calculate the quality of soil from the soil fertility index provided by the user. We also calculate a trust score for a particular user by prompting them to enter the fertility class. We compare their entered fertility class to our calculated fertility class. If they are the same, we assign a truth score of 1. If they are

not we assign a score between 0 and 0.5 depending on various conditions.

- Feedback Mechanism: Every distributor provides a farmer with a OrderID. Using this order ID, we take feedback from a farmer who is growing the seeds about the seed quality. We then score the entire batch of seeds based on a score of 1-5 with a comment. For every good/bad score, we try to fix the root issue by having a scoring system for STLs, distributors and seedGrowers based on the feedback by the farmer. Currently, it is a rule based system, but as we are able to collect more data and with time, it will transition into a robust ML system. The sentiment analysis is done using NLP techniques to get a polarity score which is then averaged over a particular STL/distributor/seedGrowers to give the necessary outputs.

Human in the Loop AI System

```
def scoring(score, comment, distributorName, certified,stlName,seedGrowerName, lotID, orderID):
    """This function computes a score for distributors, seedgrowers, sentiment of farmers and stl.
        For the sentiment, we use SOTA nlp packages.

    param score: score provided by farmer
    param orderID: orderID provided by farmer
    param comment: comment by the farmer on the seed quality

    Return type:Dict
    Returns: {
        "stl":stlScore,
        "dis":distributorScore,
        "sg":seedGrower,
        "sentiment":sent
    }
    """
```

Code with documentation

OrderID	14CAK	19KOPO	19BNGA
Avg STL Score	4	3	1
Avg Distributor Score	3.5	3	3
Average Seed Grower Score	3.8	3	5
Average Sentiment	Positive	Positive	Negative

Sample Scoring

Running the Telegram bot

- Inside the repository directory, go to **`Flask-Data-From-HTML-Form/app`**
- Add your telegram bot token to the .env file like so:
  TOKEN=<insert token here>
- Then simply the bot.py file using
  `python3 bot.py`

Running the SMS bot

- Inside the repository directory, go to **`Flask-Data-From-HTML-Form/app`**
- Due to limitations of a Twilio free account, we were not able to make a general bot that works on user queries. Currently, the script can send an SMS to a number with the seed details
- Open sms.py and replace lines 10, 11, 16, 40 and 41 with the appropriate details (descriptions are given in the script)
- Then simply the sms.py file using
  `python3 sms.py`