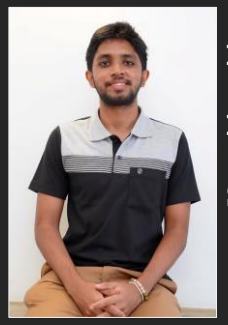
Agriprenuers

2021-090

Team

Mr.Dhammika De Silva

Ms.Janani Thamalaseelan



IT18110562

R.S.W.M.R.L Rangalla

SE

IT17023610

A.G.J.L.P

Rajapakse

SE

IT18111170

Silva S.H.I

SE

IT18027334

P.D.M Thilekarathna

CSN

Figure1: https://www.pewresearch.org/social-trends/2020/09/24/economic-fallout-from-covid-19-continues-to-hit-lower-income-families-the-hardest/

Background

- High impact on families with middle to low income.
- Extra source income is always welcomed.
- Entrepreneurship is a high-risk domain.

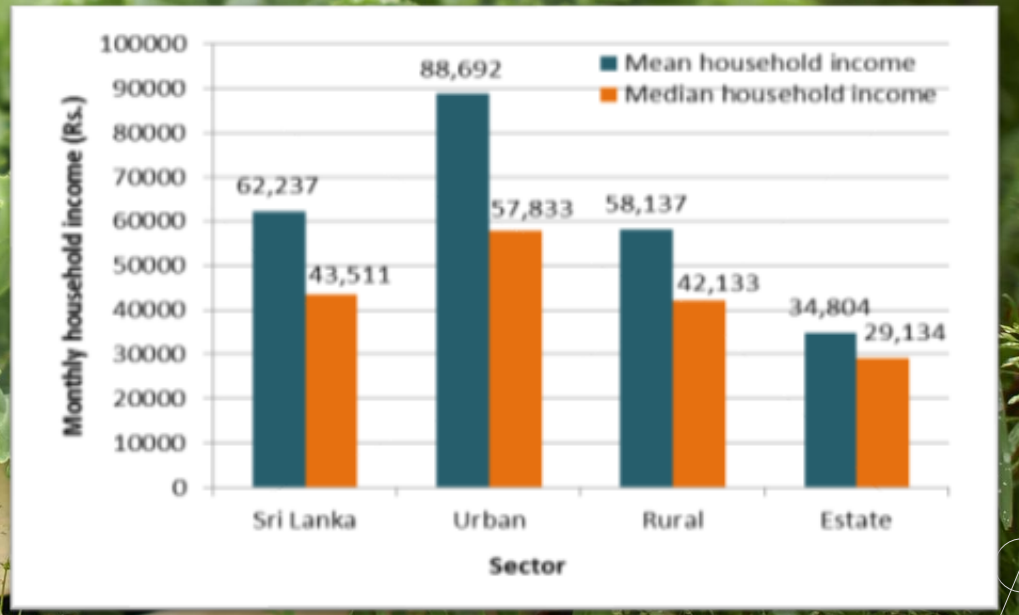


Figure2: Department of Census and Statistics Srilanka, 2016

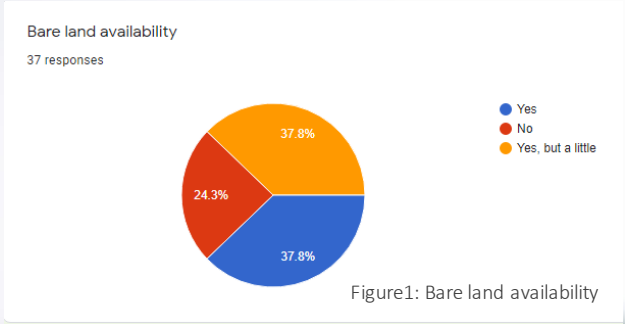


Figure1: Bare land availability

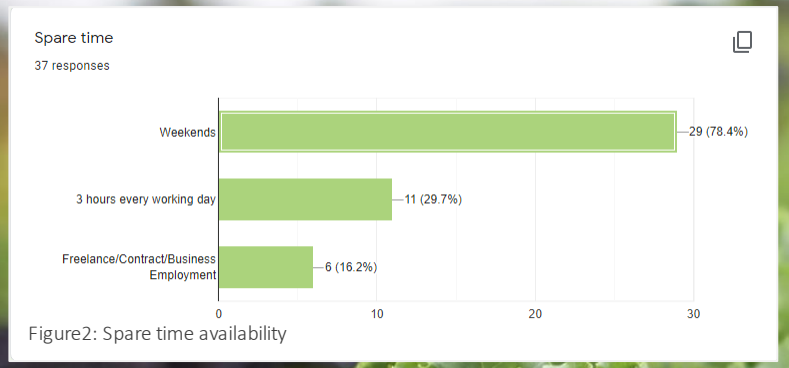


Figure2: Spare time availability

- High risk in Entrepreneurship domain.
- Less motivational strategies to promote entrepreneurship.
- No proper tool to promote entrepreneurship.
- Unavailability of stable values of a success rate figure to motivate a person to be involved in a business.
- Not utilizing bare land in many households, due to lack of time.

Research Problem

- Under Utilizing the scope of the Agriculture Domain in Urban areas
- Less motivation and promotion for entrepreneurship.

Figure3: Monthly expenditure

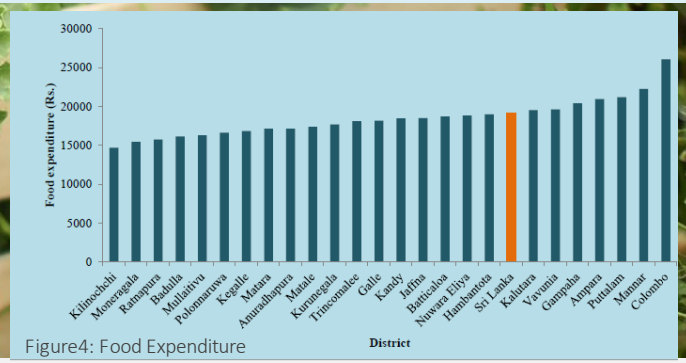


Figure4: Food Expenditure

Objectives

Main

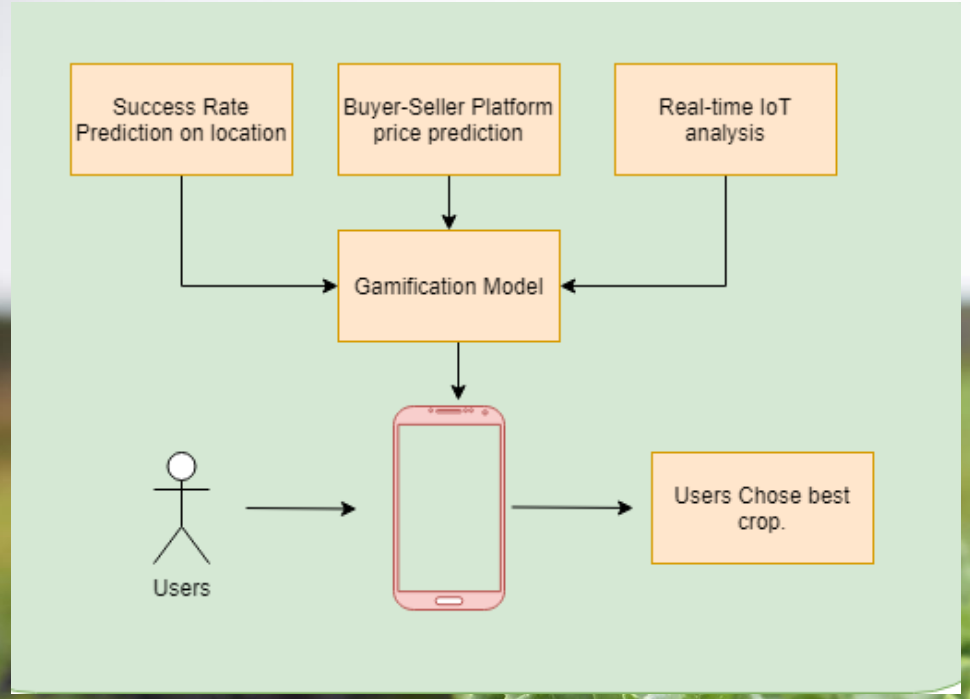
- Motivational Model for End Users to become self made entrepreneurs.

Sub-objectives

- Gamification model.
- Realtime data analysis IoT model.
- Robust Image processing model and prediction.
- Market Rate prediction model with time.

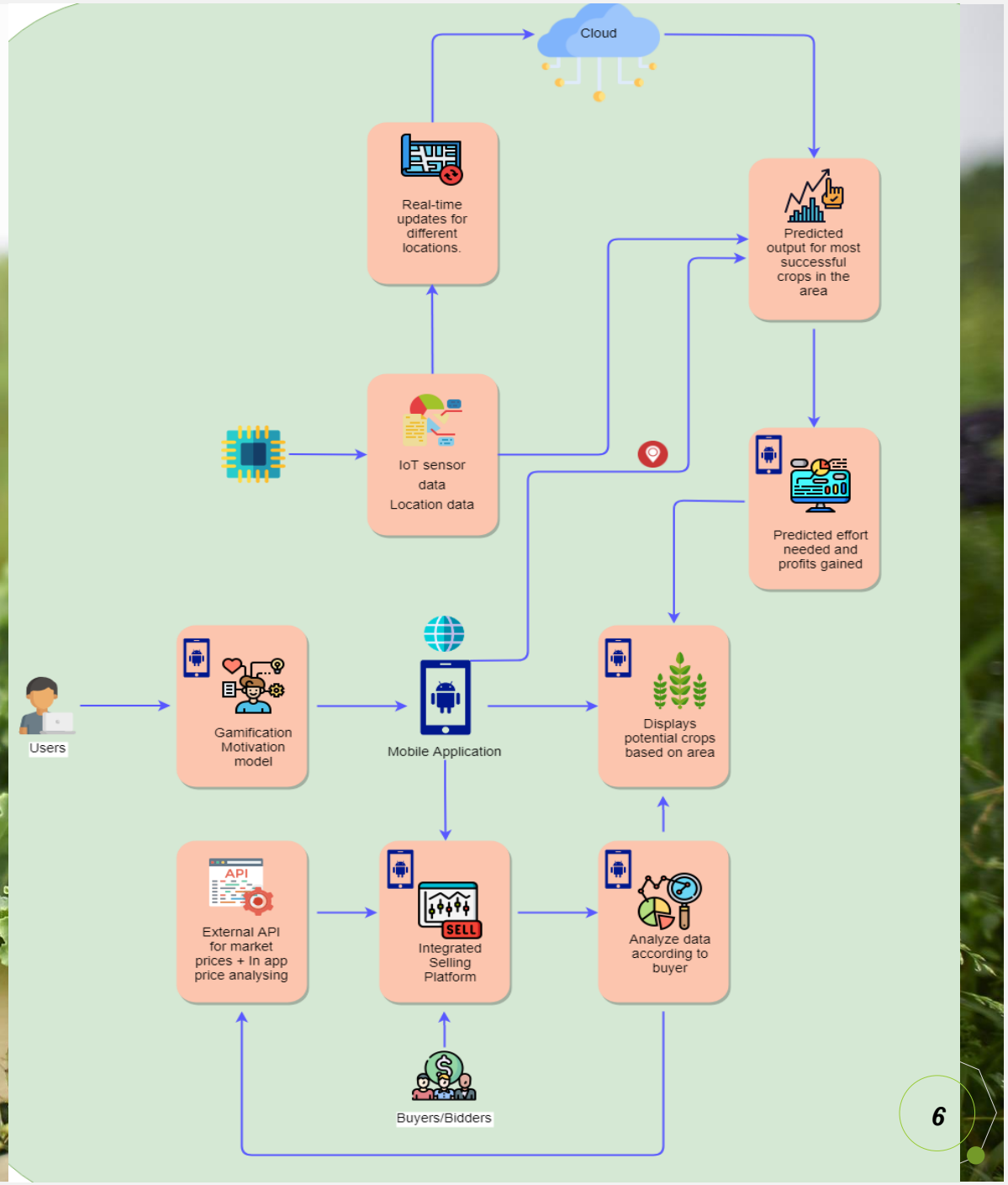
Other Objectives

- Social acceptance.
- Regional leaderboard implementation.
- Reward Scheme.
- Auction Platform.



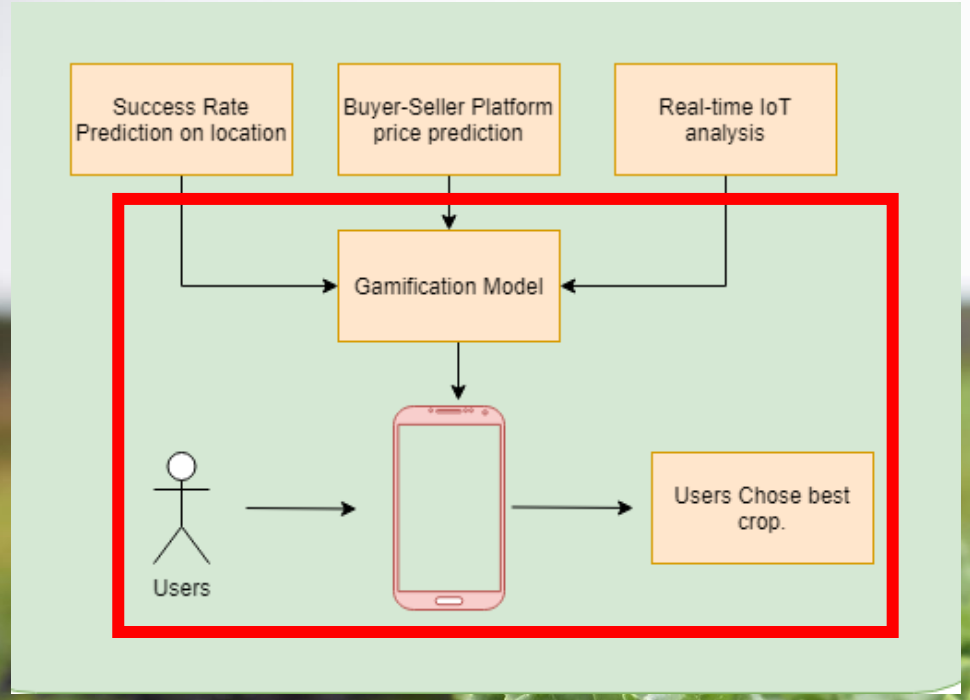
System Overview Diagram

- 2 Prediction Models with the help of Machine Learning
- 1 IoT Analysis Model
- Gamification Model



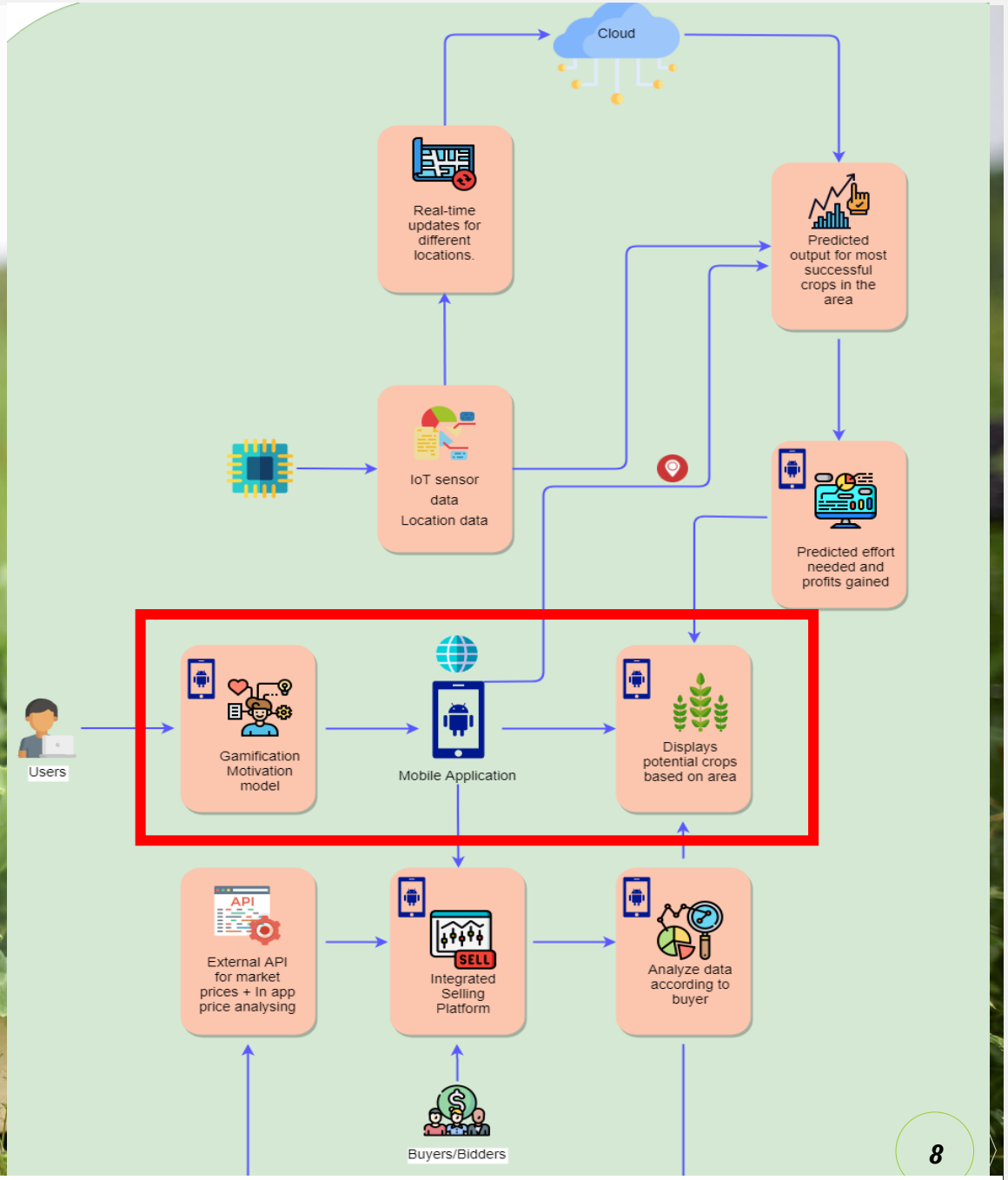
IT18111170 | Silva S.H.I

Specialization: Software Engineering



Gamification Motivation Model

- Leaderboard & Reward Schema
- Motivational Psychological model
- Gamification model



Introduction

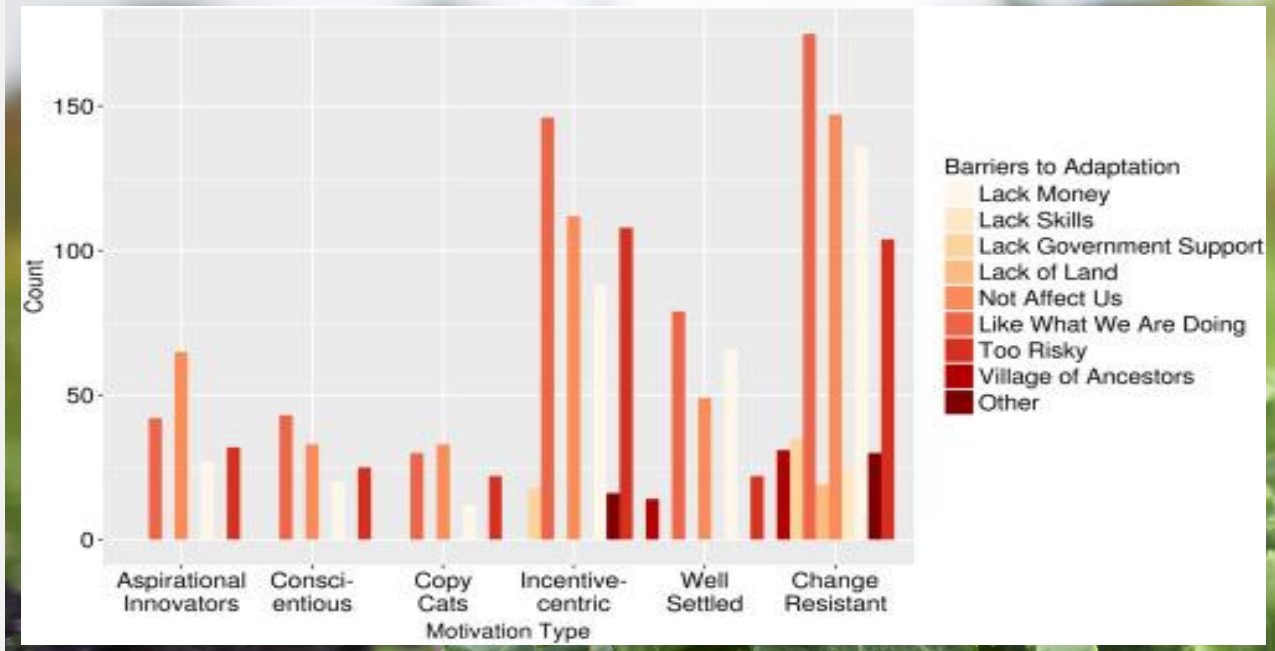


Figure: Motivation type against Barriers to adaption

Background and Research gap

- People are lazy to motivate and become entrepreneurs in agriculture even they are eligible
- Some motivational model in agriculture can be found but Mobile App using with psychology aspect are very less.
- Implement Mobile Application with considering the psychology motivational aspects including Gamification reward model.

Main Objective

- Motivate users to become self-made entrepreneurs and sustainable in economy.
- Basically creating the Mobile Application

Sub Objective

- Create regional Leaderboard which shows ranks of users based on district and money
- Create reward model proving discounts on taxes using points for user to addict and motivate.
- Gamification model that present the earned prices for crops

Research Problem

- There is lack of motivation on Agriculture for individuals to become entrepreneurs. Even if they are eligible

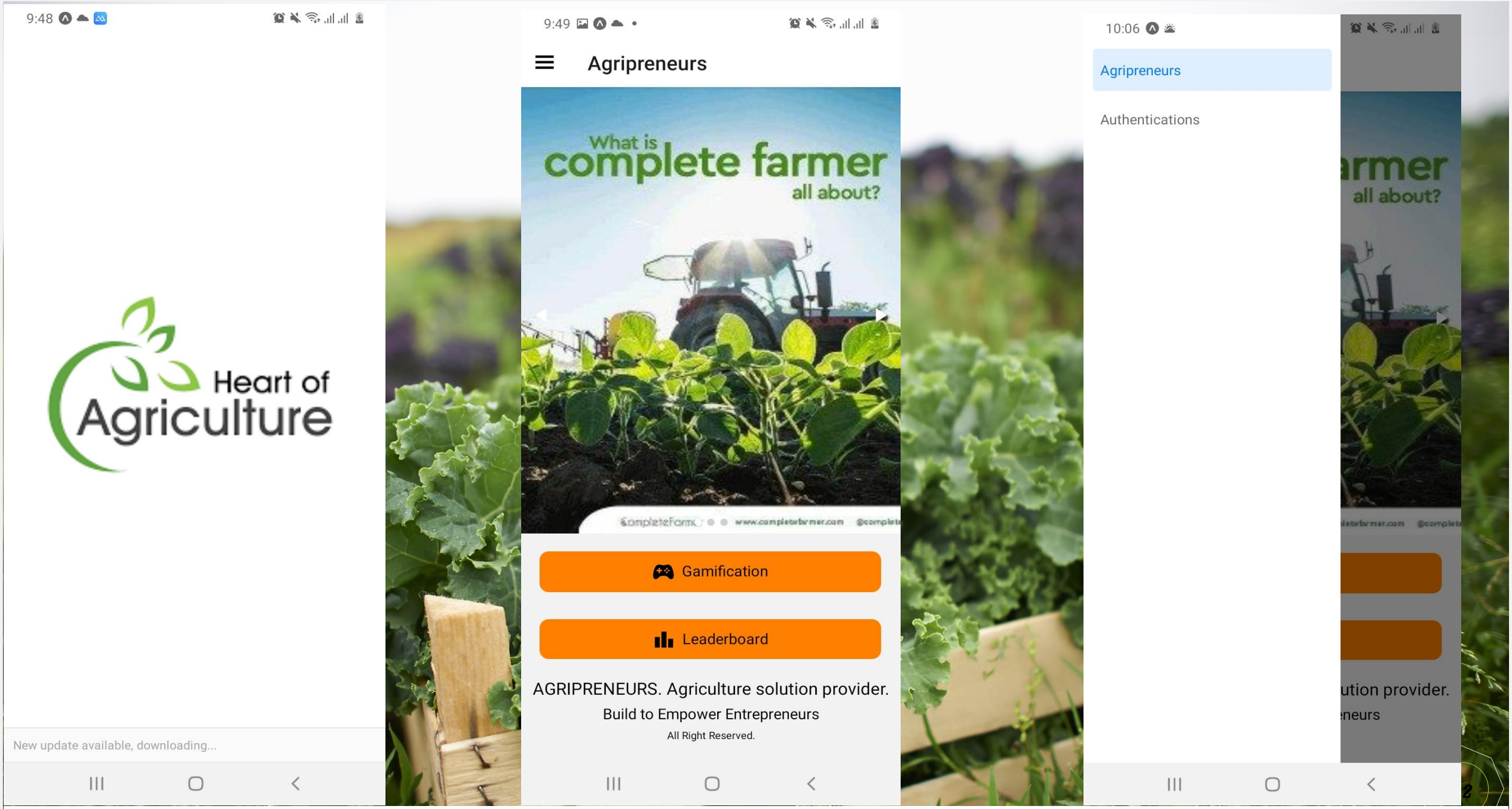
Research Methodology

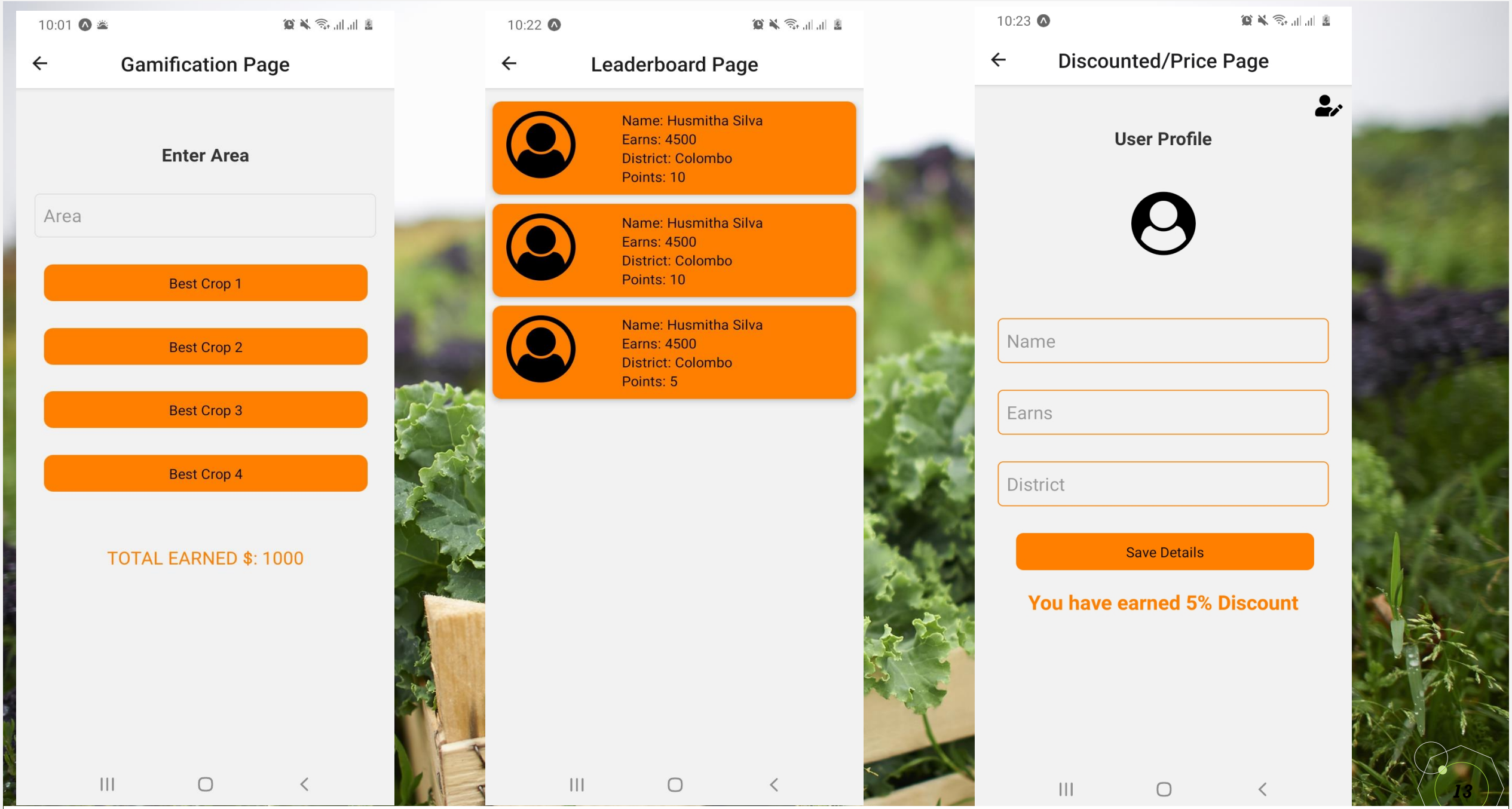
Technologies, Tools

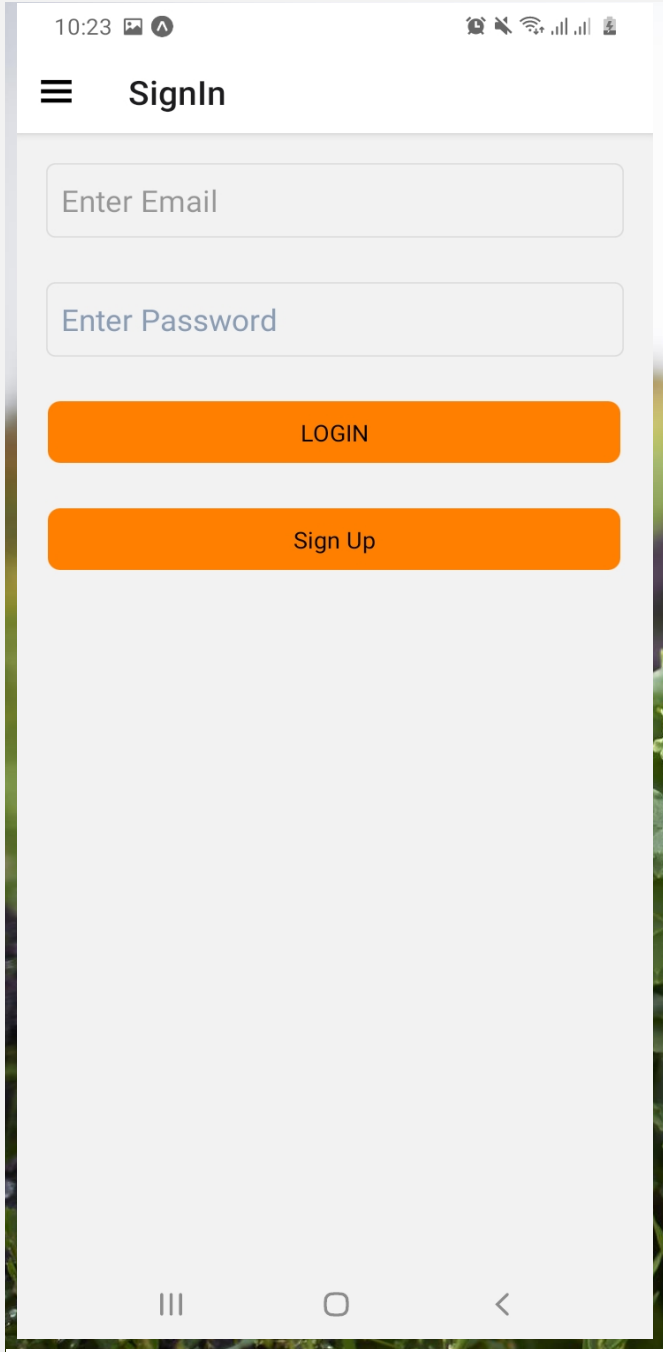
- Mobile App which has React native frontend with Python Django framework for Backend
- Firebase-Database
- Tools- VS code, Emulators, Expo, Postman

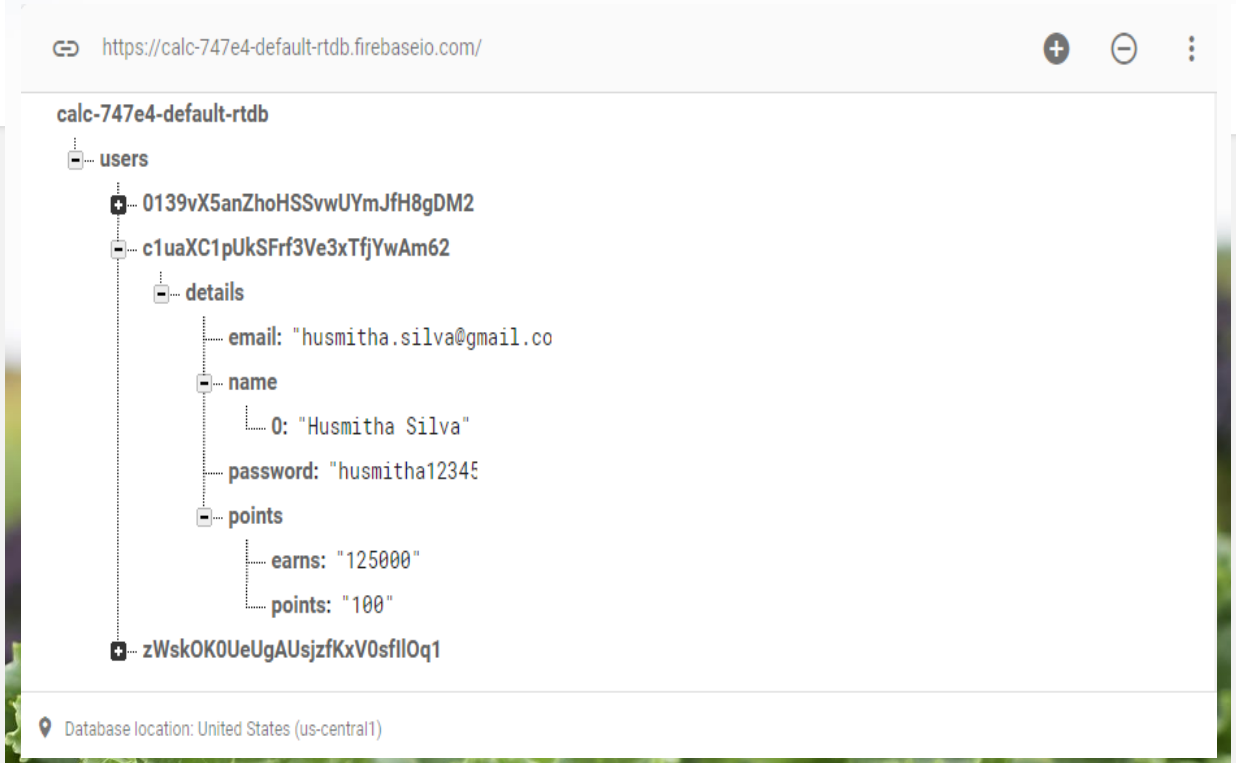
Methodology

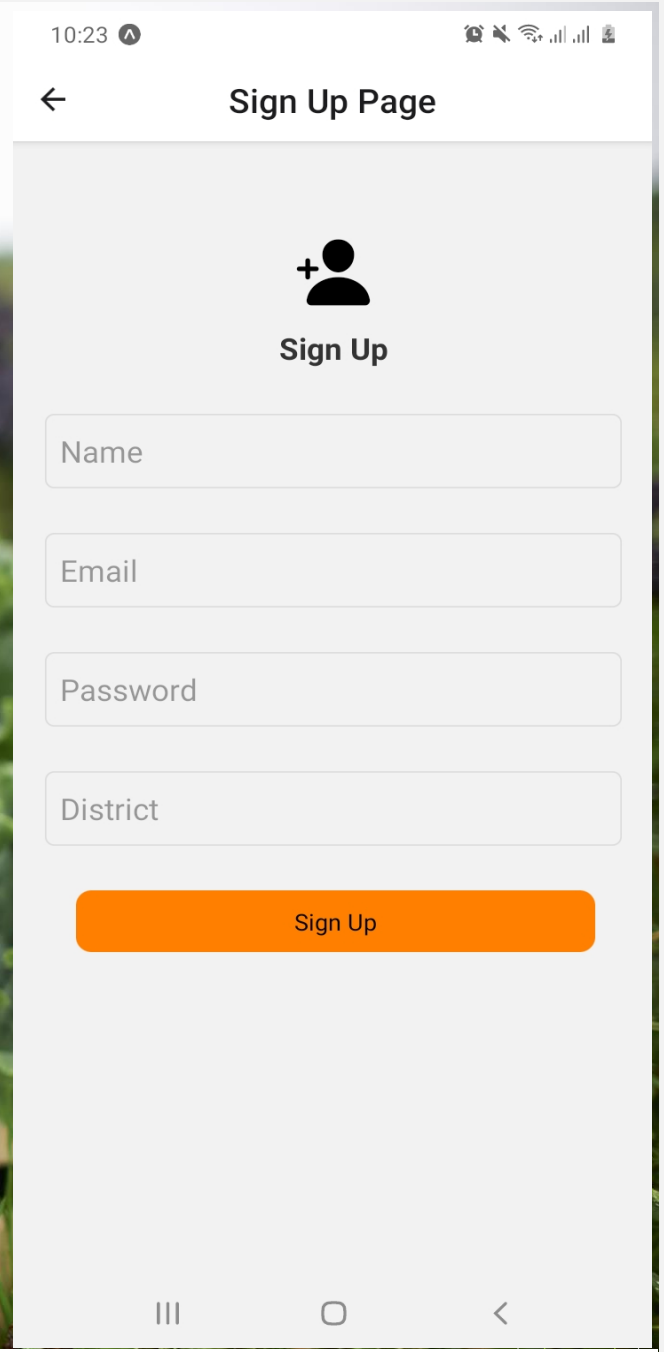
- Do a research, Watched videos and read motivational, psychological google articles for build UI
- Read articles of how colors effect humans and UX approaches
- Do a research of animation and design for Mobile App.
- Research about Django and firebase Create CRUD operation for user
- 50% build basic UI and partially backend user CRUD operations


9:48
Heart of Agriculture
New update available, downloading...
9:49
Agripreneurs
What is complete farmer all about?
Gamification
Leaderboard
AGRIPRENEURS. Agriculture solution provider.
Build to Empower Entrepreneurs
All Right Reserved.
10:06
Agripreneurs
Authentications

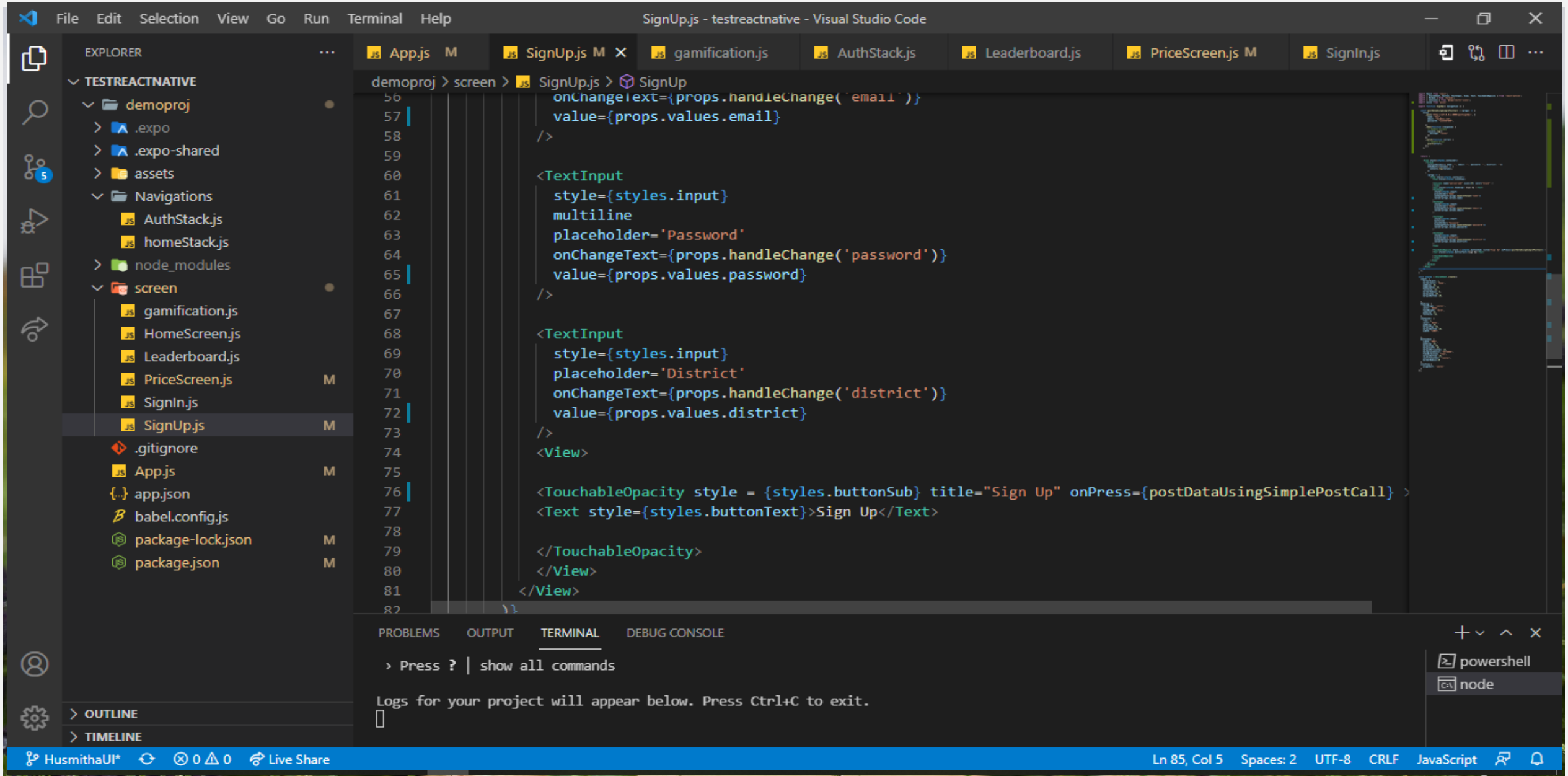

10:01
Gamification Page
Enter Area
Area
Best Crop 1
Best Crop 2
Best Crop 3
Best Crop 4
TOTAL EARNED $: 1000
10:22
Leaderboard Page
Name: Husmitha Silva
Earns: 4500
District: Colombo
Points: 10
Name: Husmitha Silva
Earns: 4500
District: Colombo
Points: 10
Name: Husmitha Silva
Earns: 4500
District: Colombo
Points: 5
10:23
Discounted/Price Page
User Profile
Name
Earns
District
Save Details
You have earned 5% Discount





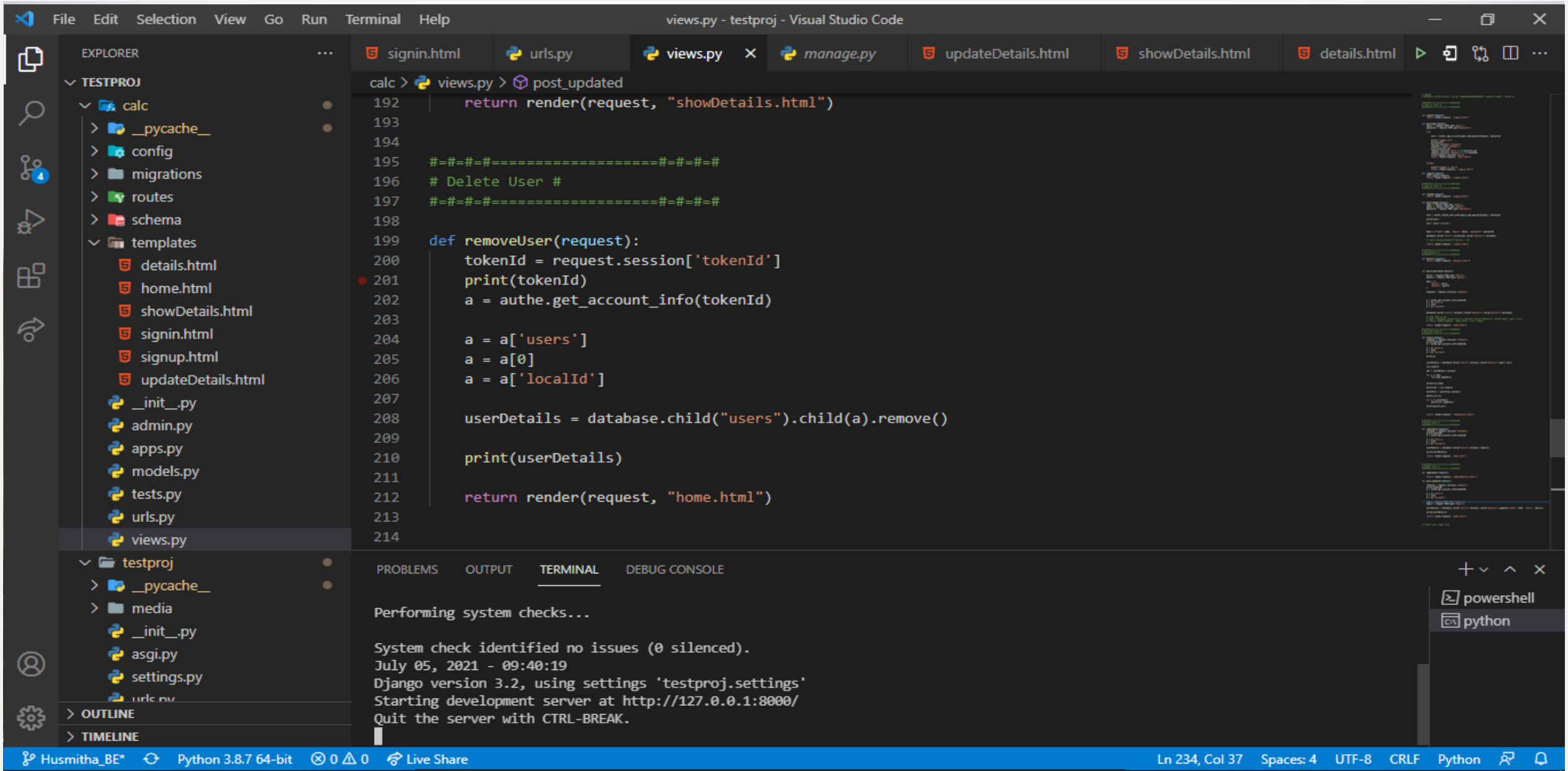


- Basic UI for the Mobile Application
- Firebase-Database for CRUD operations for User
- Hope to add user guides, app tour



React Native Front end

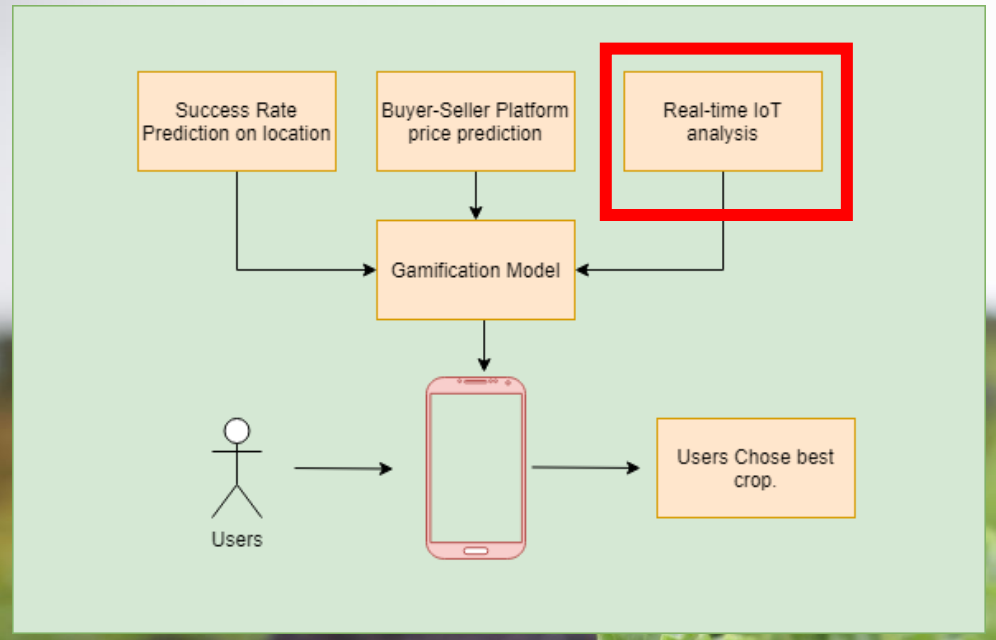
IT18111170 | Silva S.H.I | 2021-090



Django Back end

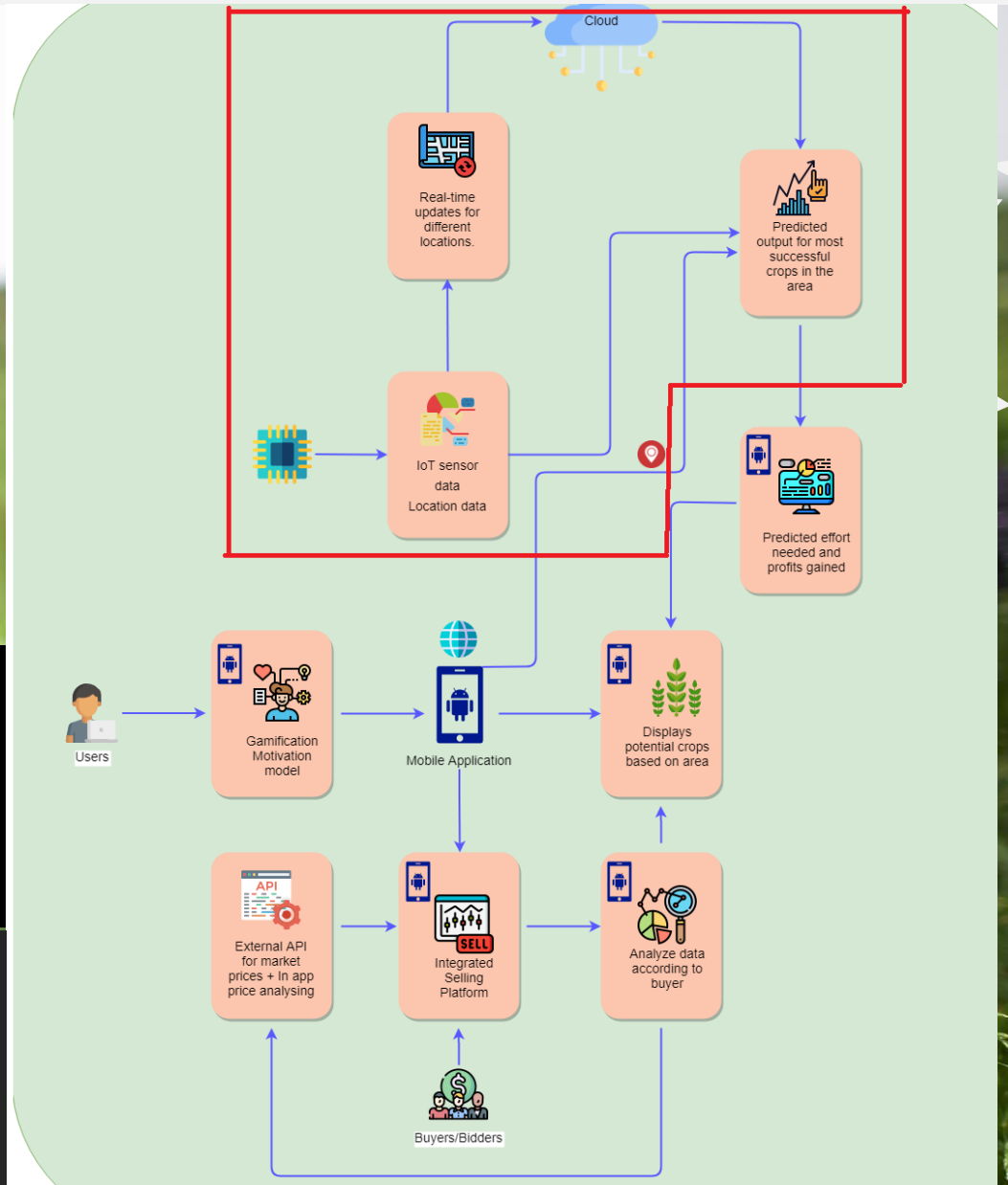
IT18027334 | Thilekarathna P.D.M

Specialization: Computer Systems & Network Engineering



IoT based data analysis model

- ✓ Check the soil condition and Environment
- ✓ Data analysis model (ML model) of the suitable crops to be grown
- ✓ Real time data analysis



RESEARCH GAP

Already existing systems are utilizing IoT methodologies;

- ✓ New type of devices
- ✓ Enables all smart farms to be connected

BUT LACK OF,

IoT methodology for motivate non- farmers.....

	Mobius using &Cube	Smart Agriculture app	Green Farm- DM	FieldCheck App	Agripreneurs
Using IoT sensor data to best crop prediction	NO	NO	NO	NO	YES
Real time update	YES	YES	YES	YES	YES
Motivational model	NO	NO	NO	NO	YES

RESEARCH QUESTION

- How does IoT effect to Motivate People to growing crops?
- How to motivate non-farmers using IoT methodologies?
- How to find most accurate ML model?
- Does it mobile device?

MAIN OBJECTIVE

•Building a IoT based data analysis model to predict the most suitable crop in the specific bare land

SUB OBJECTIVE

- Design IoT based sensor integrated hardware device to identify suitable areas of the land.
- Data analysis model to predict the suitable Crops.
- Real time update.

RESERCH METHODOLOGY

Technologies, Techniques

- NodeMCU development board as the Microcontroller (ESP8266 Microcontroller)
- Programming languages- Arduino & Python
- Tools- Arduino IDE / Jupyter notebook
- Database – Firebase
- Algorithms - Logistic Regression / Decision Tree / Random Forest / SVM / XGBoost / Naïve Bayes

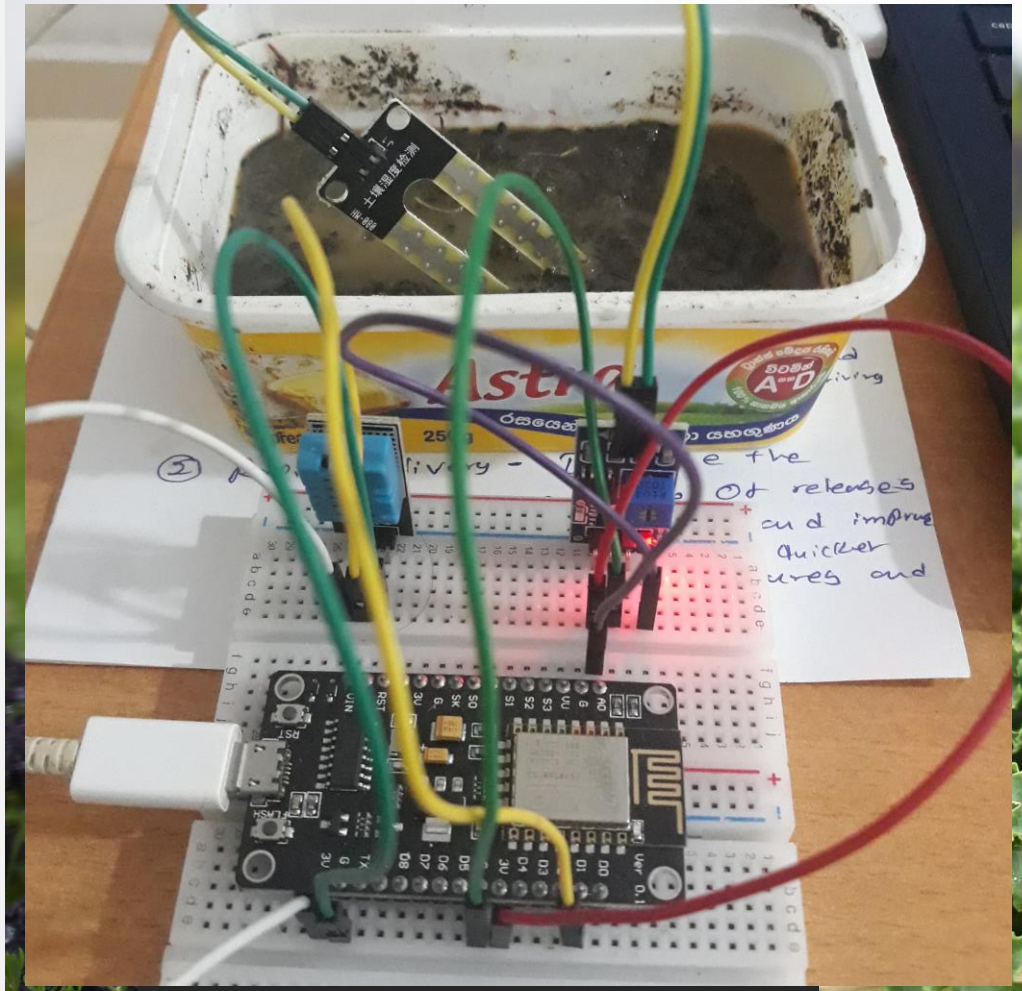
System, Software Requirements

- Network gateway need to be act as an intermediate.
- Humidity and temperature sensors (DHT11).
- Moisture detection sensor (YL-69)
- Mobile Application.
- MQTT (standard publish/subscribe protocol)

Data Sources

- Open data portal Sri Lanka
- NAICC- Govikam Magazines
- Department of Agriculture Srilanka Website
- Kaggle / YouTube tutorials / Stackoverflow

Evidences for the completion



Hardware Device

jupyter CropPrediction Last Checkpoint: 06/27/2021 (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run Code

In [4]:
```
PATH = 'F:/My Courses/SensorSample.csv'
df = pd.read_csv(PATH)
```

In [5]:
```
df.head()
```

Out[5]:

	Humidity	Temperature	Crop	Unnamed: 3
0	55	21	Chilli	NaN
1	55	22	Chilli	NaN
2	55	23	Chilli	NaN
3	55	24	Chilli	NaN
4	55	25	Chilli	NaN

In [6]:
```
df.tail()
```

Out[6]:

	Humidity	Temperature	Crop	Unnamed: 3
272	59	27	Bitter Gourd	NaN
273	60	24	Bitter Gourd	NaN
274	60	25	Bitter Gourd	NaN
275	60	26	Bitter Gourd	NaN
276	60	27	Bitter Gourd	NaN

In [7]:
```
df.size
```

Retrieving sample data set

Evidences for the completion

```
In [19]: from sklearn.tree import DecisionTreeClassifier

         DecisionTree = DecisionTreeClassifier(criterion="entropy",random_state=2,max_depth=5)

         DecisionTree.fit(Xtrain,Ytrain)

         predicted_values = DecisionTree.predict(Xtest)
         x = metrics.accuracy_score(Ytest, predicted_values)
         acc.append(x)
         model.append('Decision Tree')
         print("DecisionTrees's Accuracy is: ", x*100)

         print(classification_report(Ytest,predicted_values))
```

```
DecisionTrees's Accuracy is:  67.85714285714286
              precision    recall  f1-score   support

 Bitter Gourd       0.50      1.00      0.67         7
      Brinjal       1.00      0.75      0.86         8
    Capsicum        0.86      0.90      0.88        20
      Chilli        0.50      0.36      0.42        14
      Tomato        0.40      0.29      0.33         7

    accuracy                            0.68        56
   macro avg        0.65      0.66      0.63        56
weighted avg        0.69      0.68      0.67        56
```

```
In [20]: from sklearn.model_selection import cross_val_score

In [21]: score = cross_val_score(DecisionTree, features, target,cv=2)

In [22]: score

Out[22]: array([0.61870504, 0.49275362])
```

Accuracy & cross validation score – Decision Tree

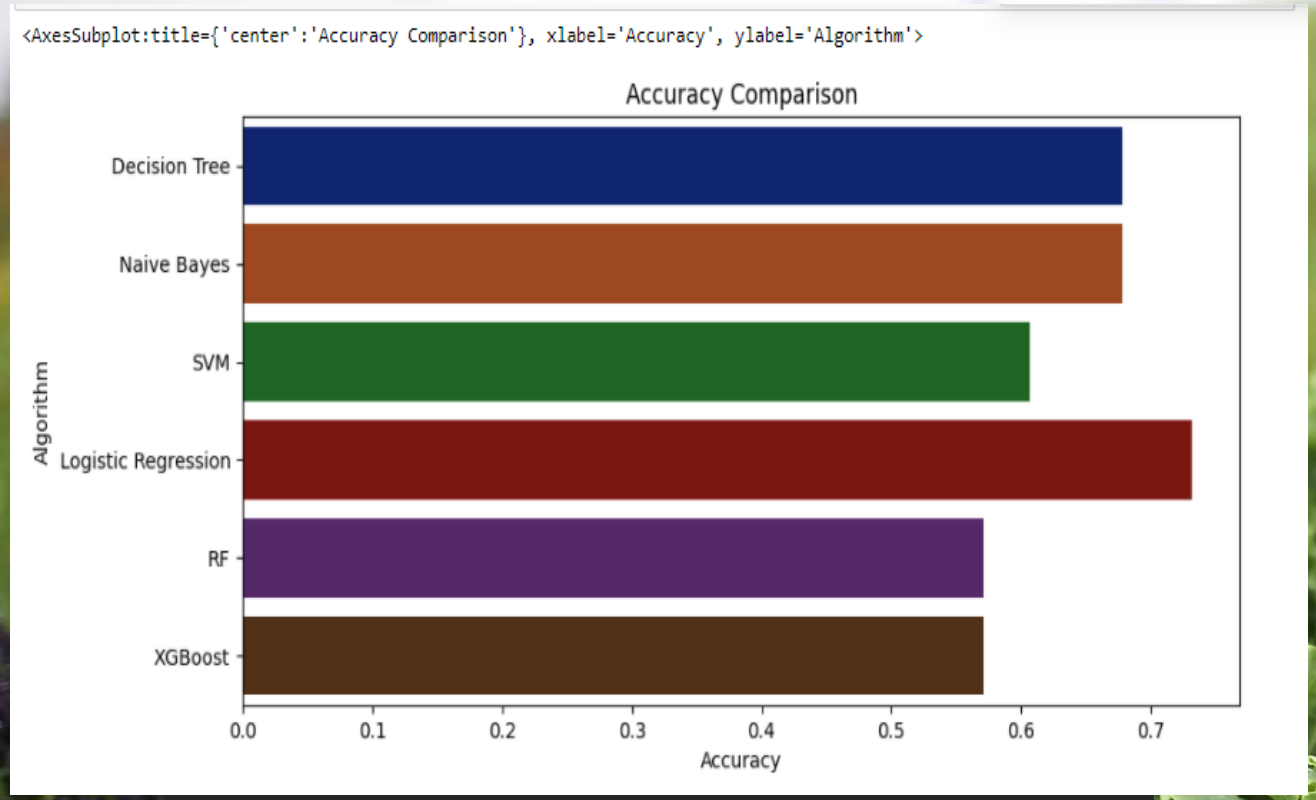
```
In [24]: from sklearn.naive_bayes import GaussianNB

         NaiveBayes = GaussianNB()

         NaiveBayes.fit(Xtrain,Ytrain)

         predicted_values = NaiveBayes.predict(Xtest)
         x = metrics.accuracy_score(Ytest, predicted_values)
         acc.append(x)
         model.append('Naive Bayes')
         print("Naive Bayes's Accuracy is: ", x)

         print(classification_report(Ytest,predicted_values))
```

```
Naive Bayes's Accuracy is:  0.6785714285714286
              precision    recall  f1-score   support

 Bitter Gourd       0.50      0.71      0.59         7
      Brinjal       1.00      0.75      0.86         8
    Capsicum        0.94      0.85      0.89        20
      Chilli        0.53      0.64      0.58        14
      Tomato        0.20      0.14      0.17         7

    accuracy                            0.68        56
   macro avg        0.63      0.62      0.62        56
weighted avg        0.70      0.68      0.68        56
```

```
In [25]: # Cross validation score (NaiveBayes)
         score = cross_val_score(NaiveBayes,features,target,cv=5)
         score

Out[25]: array([0.60714286, 0.69642857, 0.89090909, 0.76363636, 0.61818182])
```

Accuracy & cross validation score- Naïve Bayes

Evidences for the completion



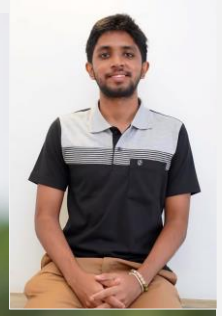
Accuracy Comparison

	Humidity	Temperature	Crop
0	55	21	Chilli
1	55	22	Chilli
2	55	23	Chilli
3	55	24	Chilli
4	55	25	Chilli

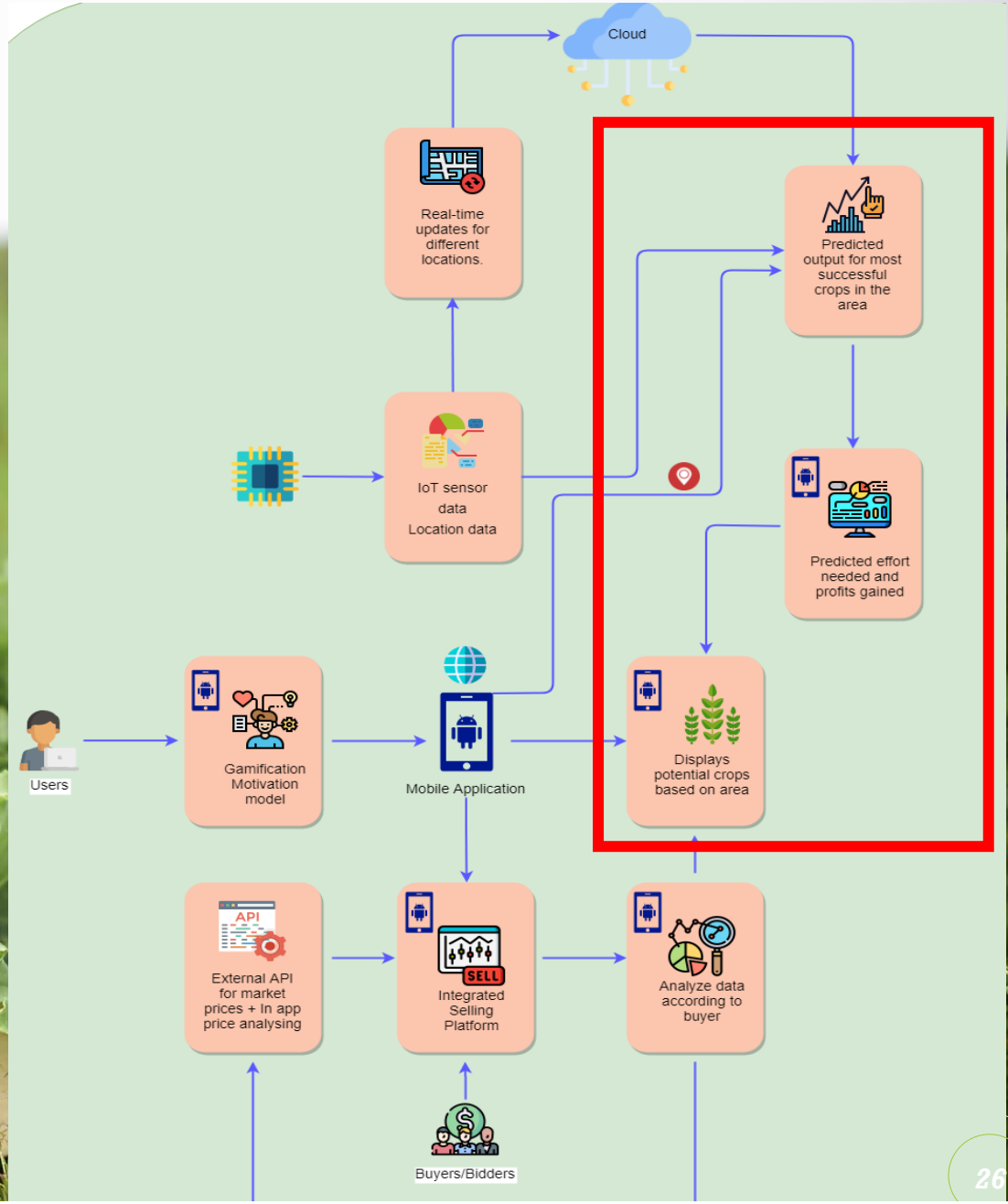
```
In [44]: data = np.array([[55,21]])
         prediction = LogReg.predict(data)
         print(prediction)

         ['Chilli']

In [45]: data = np.array([[55,21]])
         prediction = RF.predict(data)
         print(prediction)

         ['Capsicum']
```

IT18110562 | R.S.W.M.R.L. Rangalla

Specializing : Software Engineering



Robust Image processing model & Prediction model



Background

- Prediction machine learning mainly involves in thee supervised machine learning domain
- Detect abnormal crop groups due to diseases.
- Machine learning prediction algorithms have been evolving to give more predictive and accurate results.

Research Question

- To build a robust plant disease detection model.
- Success rate predictor according to user location

Research Gap

- **Robust image processing model** to detect disease and give solution by fertilizers.
- **Success rate prediction model** according to data analyzed from the application itself.

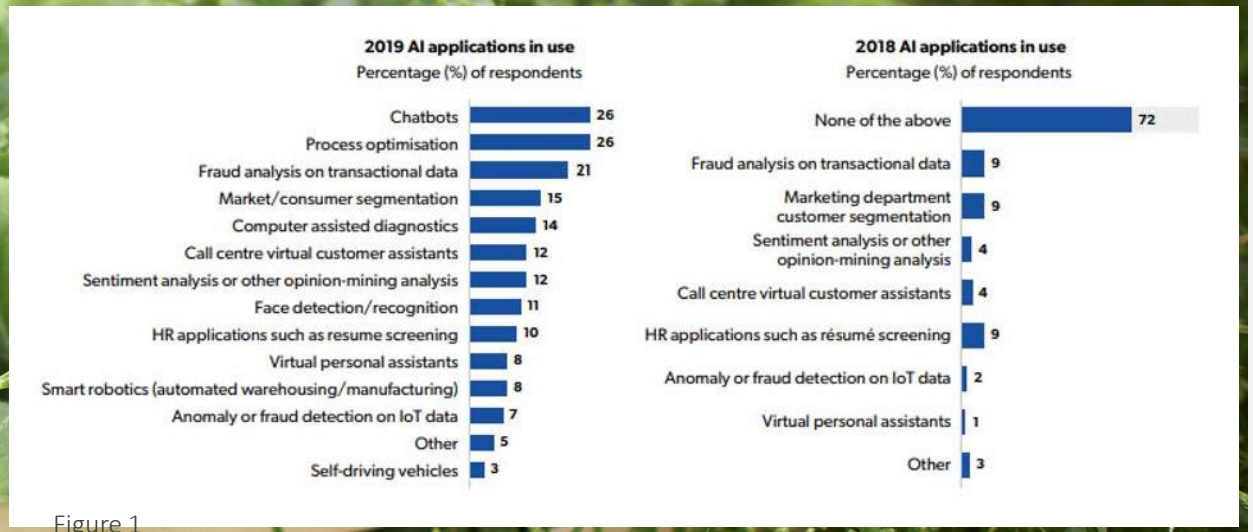


Figure 1

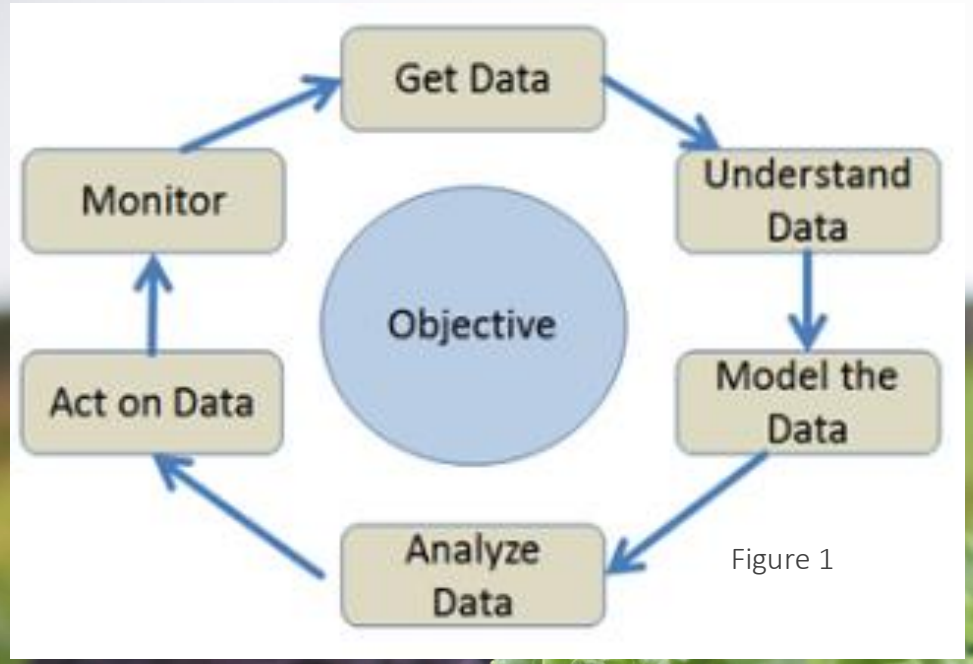


Figure 1

Objectives

- Motivate
- Robust Image processing model.
- Success rate predictor.

Sub objectives

- Discovering the most appropriate algorithm for disease detection.
- Robust image processing model.
- Detect abnormal plants by analysis the uploaded picture.
- Provide appropriate solutions/fertilizers.
- Obtaining a reliable dataset.

Methodology

- Built a data preprocessing model for feature extraction.
- Done by feature extraction methods which included Hu moments, Haralick and color histogram.
- To robustly utilize the model, only 2 main training labels are being introduced which are diseased plants and healthy plants respectively.
- Colors of the leaves are being analyzed to train the image processing model.

- The data-set is being divided into a ratio of 0.8 to 0.2 for validation and training respectively.
- 10 Fold Cross validation is done to obtain an accurate model, since a limited data set is available.

Algorithms

- All popular model training algorithms were compared with each other to other.
- Random Forest Tree had the highest accuracy in detecting a diseased plant of 98%.

Tools and Technologies

- Python 3 using Jupyter Notebook
- Tensorflow
- OpenCV
- Sklearn
- H5py to store data.
- Joblib to increase efficency.

Click to add text

Algorithms

- Support Vector Machine
- Random Forest
- Naïve Bayes
- Decision Trees
- Logistic Regression
- Linear Discriminant

Evidence for Completion

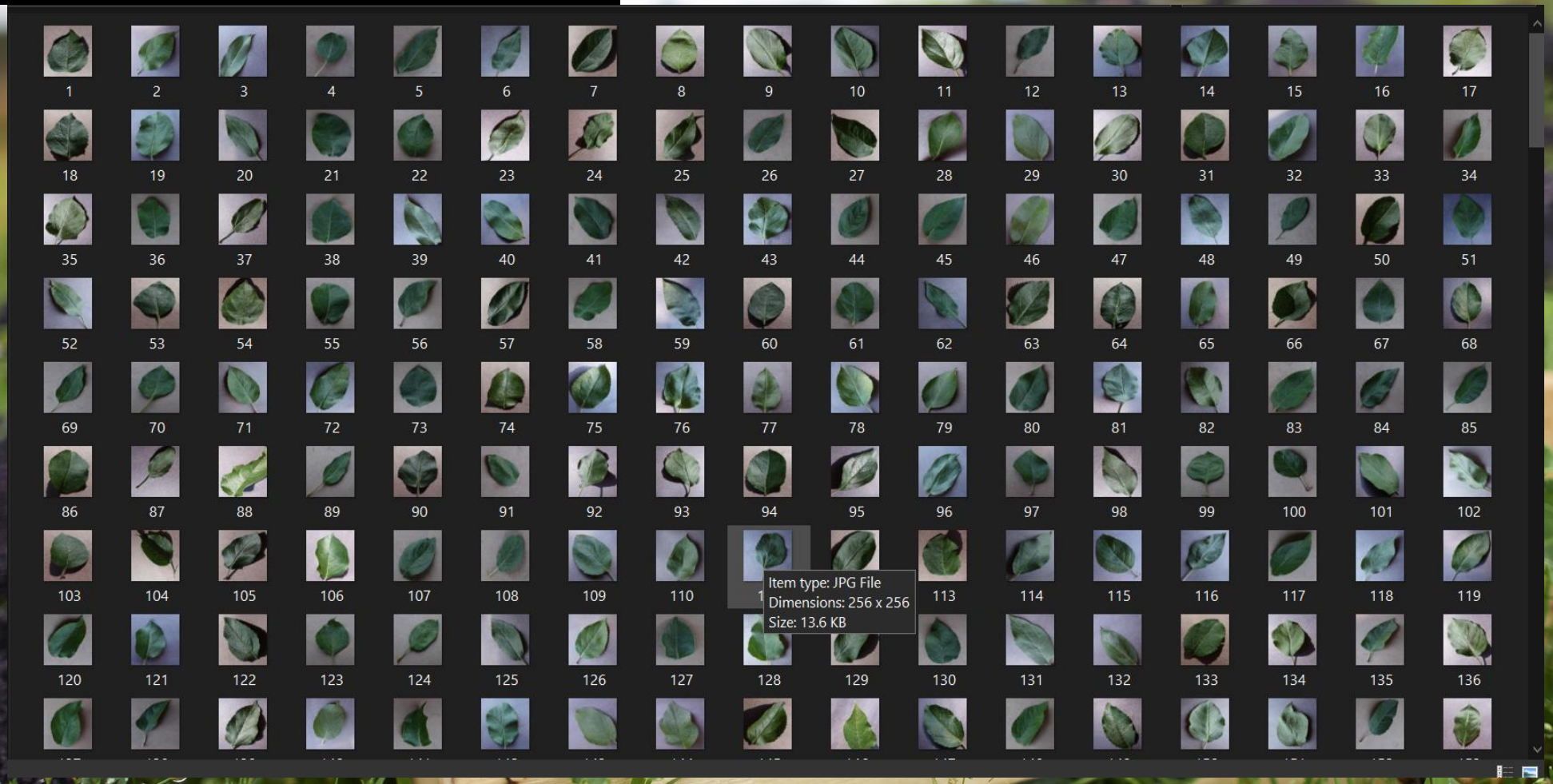


Figure B: Healthy Leaves

Evidence for Completion

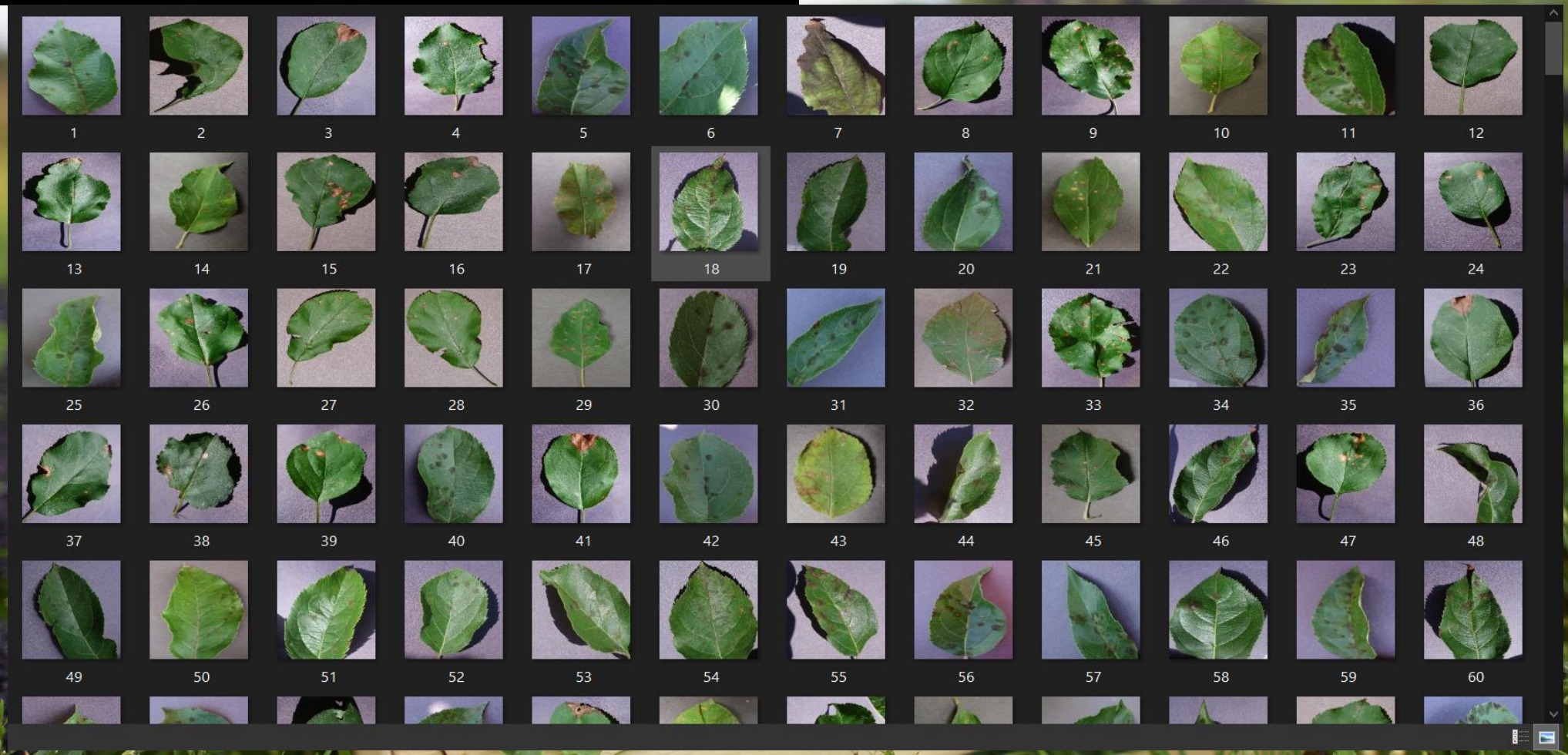


Figure B: Diseased Leaves

Evidence for Completion

```
#feature descriptor1: Hu moments (yellowish/fire)
def fd_hu_moments(image):
    image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    feature = cv2.HuMoments(cv2.moments(image)).flatten()
    return feature
```

```
#feature descriptor2: Haralick texture(Quantify image according to the texture)
def fd_haralick(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    haralick = mahotas.features.haralick(gray).mean(axis = 0)
    return haralick
```

```
#featur descriptor3: Color Histogram
def fd_histogram(image, mask = None):
    image = cv2.cvtColor(image,cv2.COLOR_BGR2HSV)
    hist = cv2.calcHist([image],[0,1,2],None,[bins,bins,bins],[0,256,0,256,0,256])
    cv2.normalize(hist,hist)
    return hist.flatten()
```

Figure C: Defining the feature extraction algorithms

Evidence for Completion

jupyter classification_model_final Last Checkpoint: 6 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Run Code

In [10]:

```
# for loop to loop over the traininf data in the subfolder

for training_name in train_labels:

    #joining training data path and each species training folder
    dir = os.path.join(train_path, training_name)

    #getting the current training label
    current_label = training_name

    #looping over the images in each subfolder
    for x in range(1,images_per_class+1):

        #joining strings to get the image file name
        file = dir + "/" +str(x) + ".jpg"

        image =  cv2.imread(file)
        image = cv2.resize(image, fixed_size)

        #Running the function bit by bit

        RGB_BGR = rgb_bgr(image)
        BGR_HSV = bgr_hsv(RGB_BGR)
        IMG_SEG = img_segmentation(RGB_BGR, BGR_HSV)

        #Calling global feature descriptors

        fv_hu_moments = fd_hu_moments(IMG_SEG)
        fv_haralick = fd_haralick(IMG_SEG)
        fv_histogram = fd_histogram(IMG_SEG)

        #Concatenate
```

Figure D: Applying the feature extraction functions to the dataset, Looping over all the images on .jpg format

jupyter classification_model_final Last Checkpoint: 6 hours ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Connected Not Trusted Python 3

Run Code

```
print("[Status] preprocessed folder:{}".format(current_label))

print("[Status] completed global feature extraction")
```

```
[Status] preprocessed folder:diseased
[Status] completed global feature extraction
[Status] preprocessed folder:diseased
[Status] completed global feature extraction
[Status] preprocessed folder:diseased
[Status] completed global feature extraction
[Status] preprocessed folder:diseased
[Status] completed global feature extraction
[Status] preprocessed folder:diseased
[Status] completed global feature extraction
[Status] preprocessed folder:diseased
[Status] completed global feature extraction
[Status] preprocessed folder:diseased
[Status] completed global feature extraction
[Status] preprocessed folder:diseased
[Status] completed global feature extraction
[Status] preprocessed folder:diseased
[Status] completed global feature extraction
[Status] preprocessed folder:diseased
[Status] completed global feature extraction
[Status] preprocessed folder:diseased
[Status] completed global feature extraction
[Status] preprocessed folder:diseased
[Status] completed global feature extraction
```

In [11]:

```
print("[Status] feature vector size{}",format(np.array(labels).shape))
```

```
[Status] feature vector size{} (1600,)
```

Figure E : Shows the result upon successful execution

Evidence for Completion

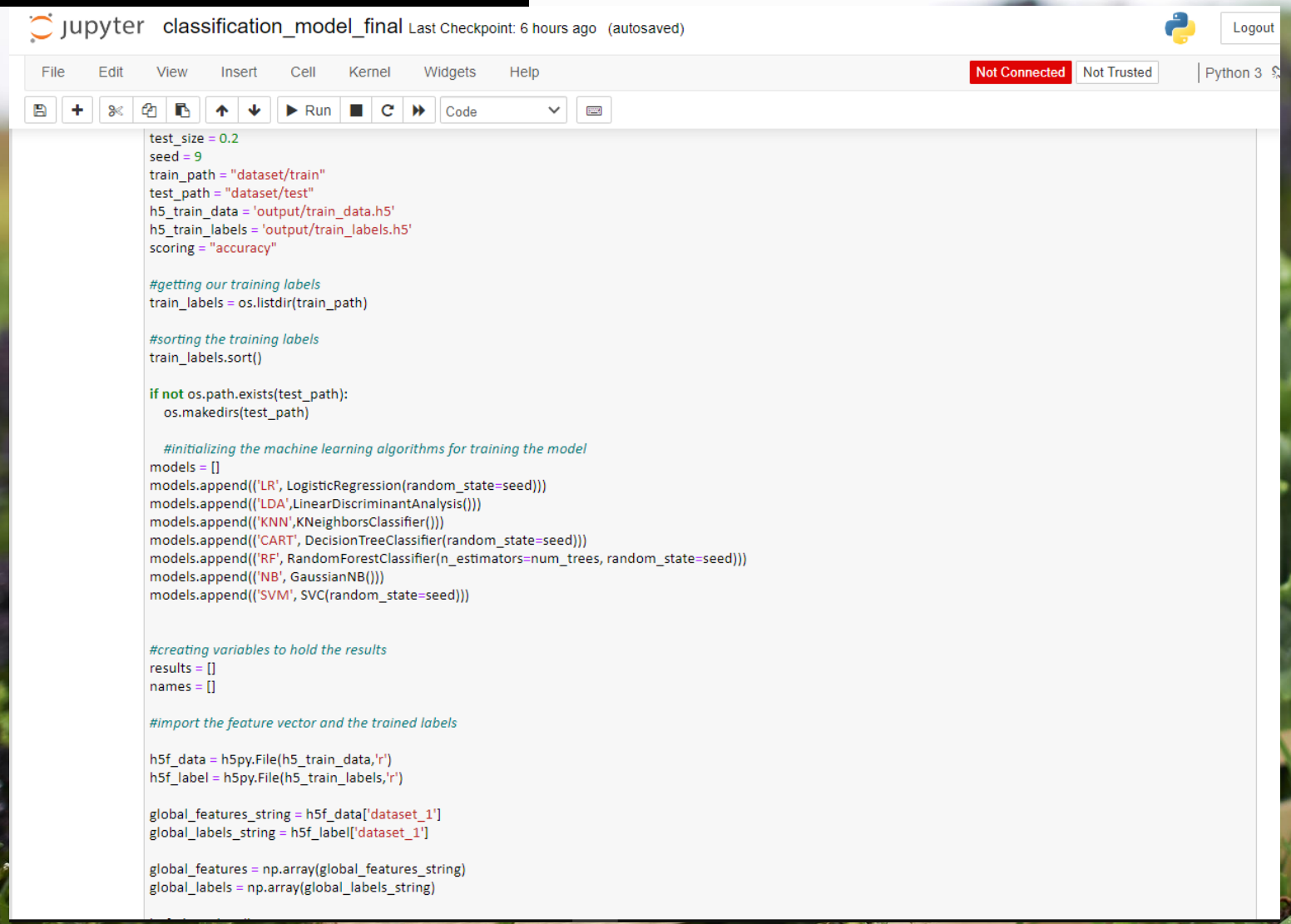


Figure F: Training the model

Evidence for Completion

jupyter classification_model_final Last Checkpoint: 6 hours ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Connected Not Trusted Python 3

Run Code

In [33]:
```
#10-fold cross validation to get the accuracy of each of the algorithms being used for training the model

for name, model in models:
    kfold = KFold(n_splits=10, random_state=seed,shuffle = True)
    cv_results = cross_val_score(model, trainDataGlobal, trainLabelsGlobal, cv=kfold, scoring=scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)

# boxplot algorithm comparison
fig = pyplot.figure()
fig.suptitle('Machine Learning algorithm comparison')
ax = fig.add_subplot(111)
pyplot.boxplot(results)
ax.set_xticklabels(names)
pyplot.show()
```

```
LR: 0.916406 (0.021833)
LDA: 0.907813 (0.019702)
KNN: 0.920312 (0.012500)
CART: 0.906250 (0.027951)
RF: 0.957031 (0.013189)
NB: 0.857031 (0.010511)
SVM: 0.916406 (0.020086)
```

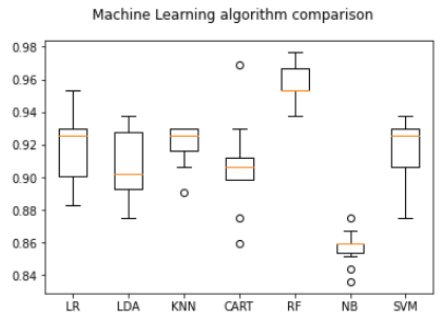


Figure G: Output of the model

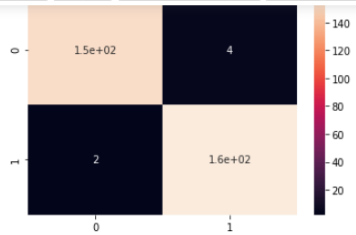
jupyter classification_model_final Last Checkpoint: 7 hours ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Connected Not Trusted Python 3

Run Code



In [47]:
```
print(classification_report(testLabelsGlobal, y_predict))
```

```
              precision    recall  f1-score   support

           0       0.99      0.97      0.98       158
           1       0.98      0.99      0.98       162

    accuracy                           0.98       320
   macro avg       0.98      0.98      0.98       320
weighted avg       0.98      0.98      0.98       320
```

In [49]:
```
from sklearn.metrics import accuracy_score
accuracy_score(testLabelsGlobal, y_predict)
```

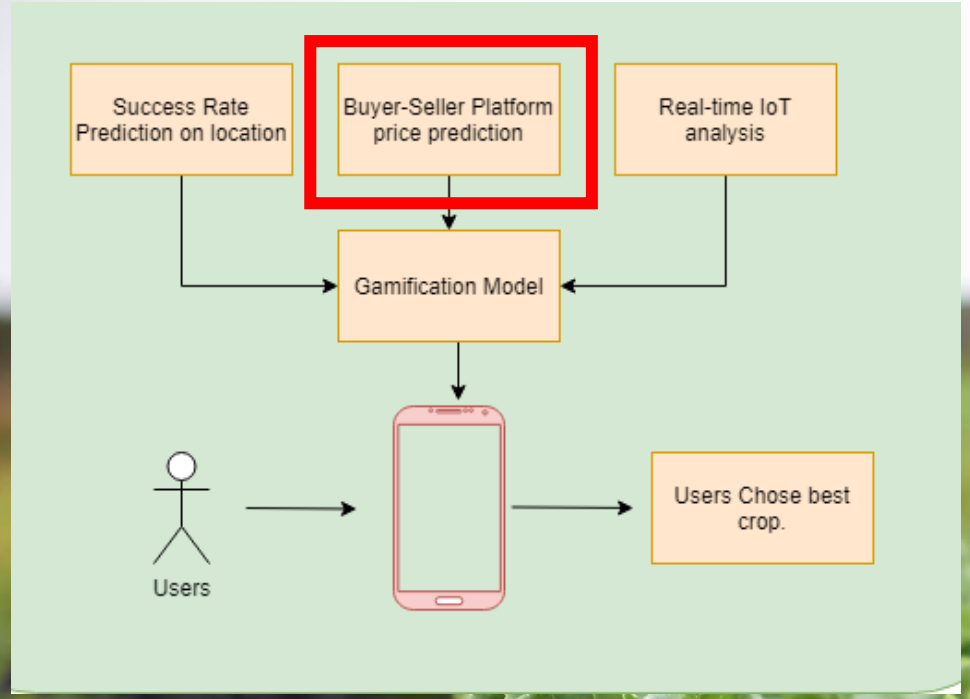
Out[49]: 0.98125

In [ ]:

Figure H: Random Forest Tree gives 98% accuracy

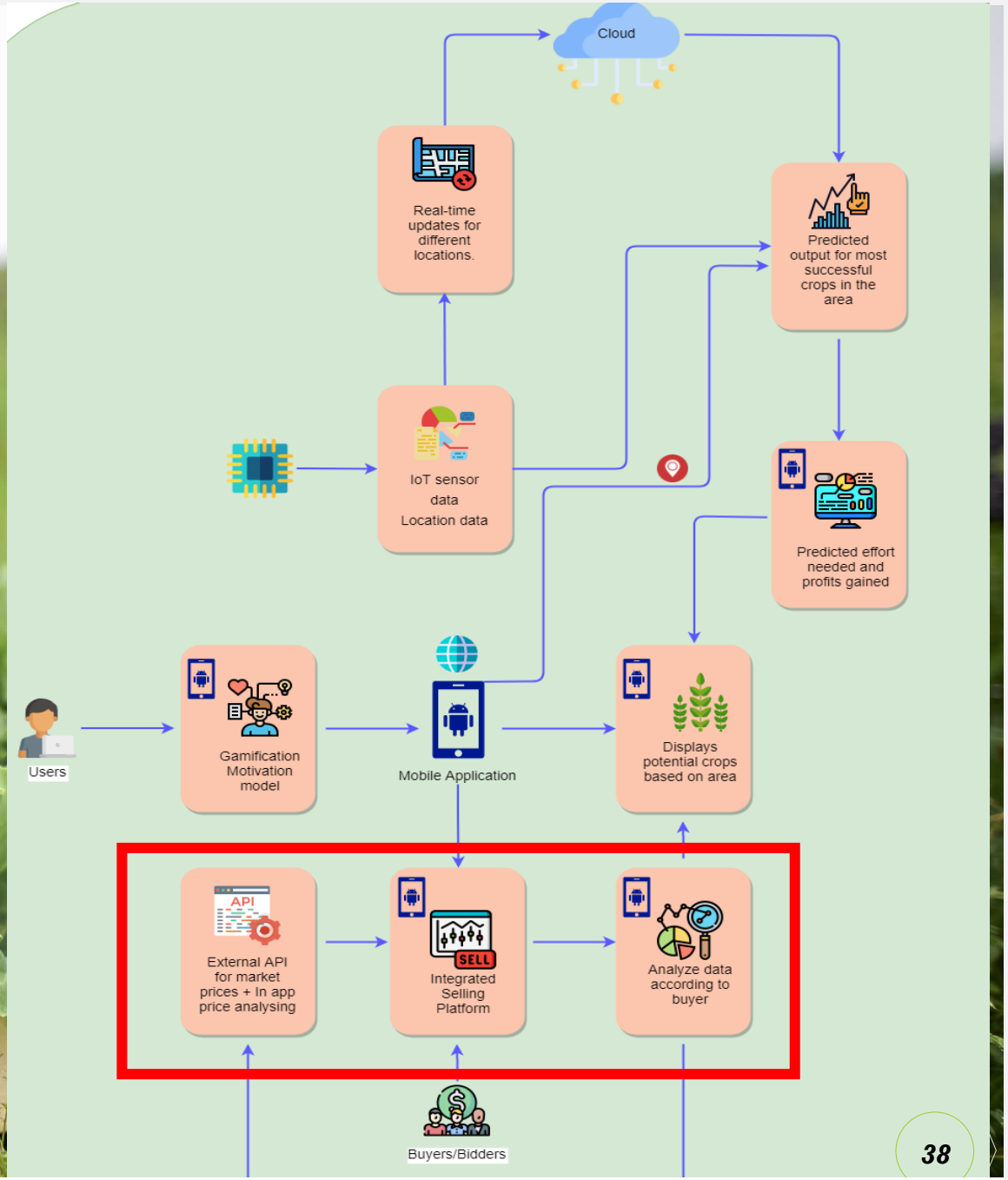
IT17023610 | A.G.J.L.P Rajapakse

Specialization: Software Engineering



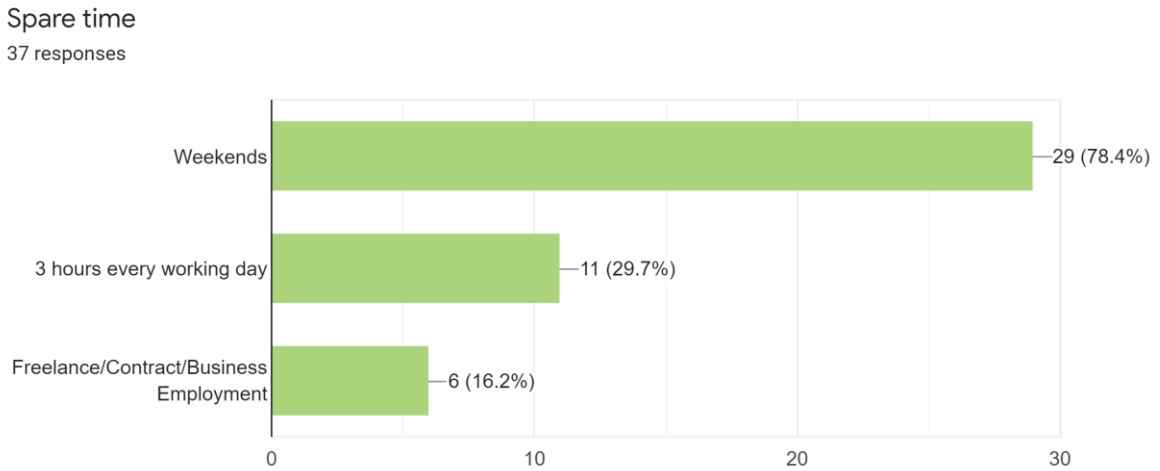
Integrated Selling Platform

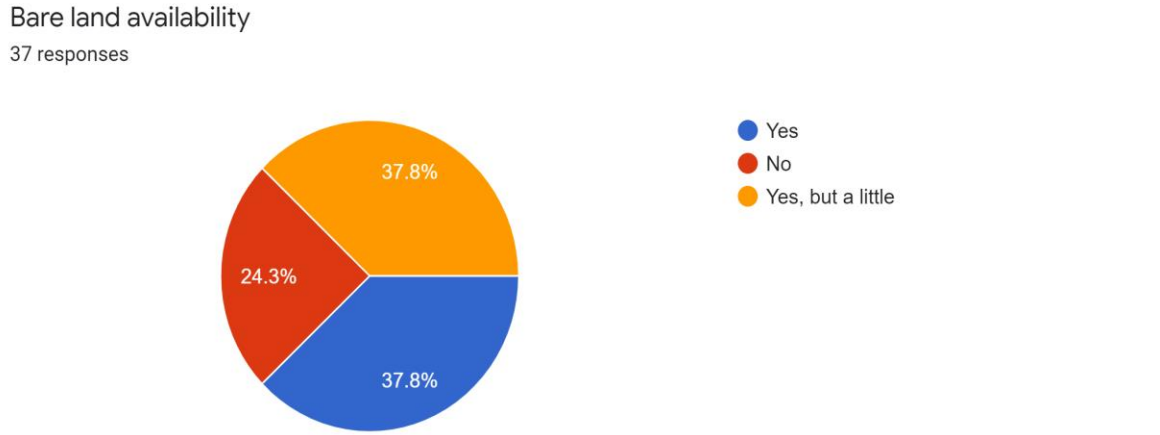
- Integrated Selling Platform
- Predicting trending crops based on analyzed data



Research Gap

- Creation of an integrated bidding platform.
- Creation of a data analysis model that uses machine learning in order to predict future trending crops.



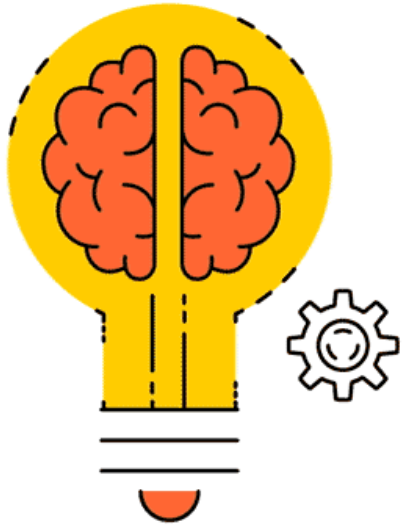


Research Question

- Is it possible to create a data analysis model that will help predict future trending crops in a given period of time?

Objectives

- Create an interactive bidding platform
- Using Machine Learning algorithms to predict future trending crops.



Sub objectives

- Designing and developing interactive interfaces for bidding platform.
- Training machine learning model using a clean dataset.
- Test the prediction accuracy of machine learning model.
- Create external API to retreive prices.

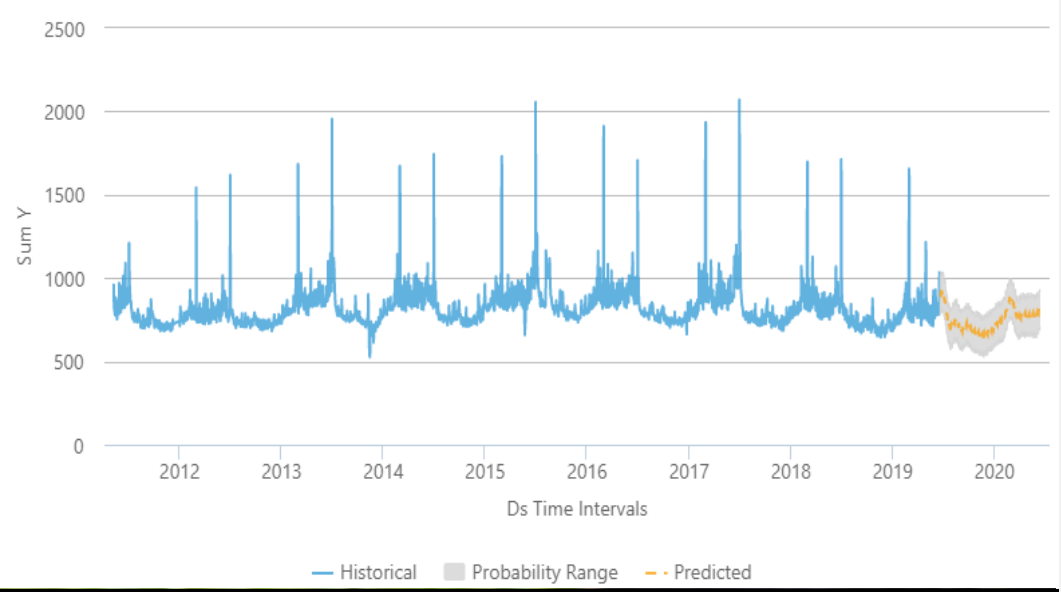
Tools and Technologies

Technologies

- React Native – Frontend
- Python Django Framework – Backend
- SQLite– Database

Tools

- Visual Studio Code



Algorithms

- Prophet algorithm (Time Series Model)

Evidence for Completion

```python
from django.urls import reverse
from django.db.models import Max
from django.utils import timezone
from django.contrib import messages
from django.shortcuts import render
from django.db import IntegrityError
from django.contrib.auth.decorators import login_required
from django.http import HttpResponse, HttpResponseRedirect
from django.contrib.auth import authenticate, login, logout

from .models import User, Category, AuctionListing, Bid, Comment


def index(request):
    obj = AuctionListing.objects.filter(active=True)
    return render(request, "auctions/index.html", {
        "objects": obj
    })


def all(request):
    obj = AuctionListing.objects.all()
    return render(request, "auctions/index.html", {
        "objects": obj
    })


def login_view(request):
    if request.method == "POST":

```

Evidence for Completion

```
30
31          # Attempt to sign user in
32          username = request.POST["username"]
33          password = request.POST["password"]
34          user = authenticate(request, username=username, password=password)
35
36          # Check if authentication successful
37          if user is not None:
38              login(request, user)
39              return HttpResponseRedirect(reverse("index"))
40          else:
41              return render(request, "auctions/invalidLogin.html", {
42                  "message": "Invalid username and/or password."
43              })
44      else:
45          return render(request, "auctions/login.html")
46
47
48  @login_required
49  def logout_view(request):
50      logout(request)
51      return HttpResponseRedirect(reverse("index"))
52
53
54  def register(request):
55      if request.method == "POST":
56          username = request.POST["username"]
57          email = request.POST["email"]
58
59          # Ensure password matches confirmation
```

Evidence for Completion

```
        # Ensure password matches confirmation
        password = request.POST["password"]
        confirmation = request.POST["confirmation"]
        if password != confirmation:
            return render(request, "auctions/invalidRegister.html", {
                "message": "Passwords must match."
            })

        # Attempt to create new user
        try:
            user = User.objects.create_user(username, email, password)
            user.save()
        except IntegrityError:
            return render(request, "auctions/invalidRegister.html", {
                "message": "Username already taken."
            })
        login(request, user)
        return HttpResponseRedirect(reverse("index"))
    else:
        return render(request, "auctions/register.html")


@login_required
def createListing(request):
    if request.method == 'POST':
        title = request.POST["title"]
        description = request.POST["description"]
        startBid = request.POST["startBid"]
        category = Category.objects.get(id=request.POST["category"])
        user = request.user
```

Evidence for Completion

```
            name=title, category=category, date=timezone.now(), startBid=startBid, description=description, user=user, imageUrl=imageUrl, act
        listing.save()
        return HttpResponseRedirect(reverse("index"))
    return render(request, "auctions/createListing.html", {
        'categories': Category.objects.all()
    })


def details(request, id):
    item = AuctionListing.objects.get(id=id)
    bids = Bid.objects.filter(auctionListing=item)
    comments = Comment.objects.filter(auctionListing=item)
    value = bids.aggregate(Max('bidValue'))['bidValue__max']
    bid = None
    if value is not None:
        bid = Bid.objects.filter(bidValue=value)[0]
    return render(request, "auctions/details.html", {
        'item': item,
        'bids': bids,
        'comments': comments,
        'bid': bid
    })


def categories(request):
    if request.method == 'POST':
        category = request.POST["category"]
        new_category, created = Category.objects.get_or_create(
            name=category.lower())
```

Evidence for Completion

```
122         else:
123             messages.warning(request, "Category already Exists!")
124         return HttpResponseRedirect(reverse("categories"))
125     return render(request, "auctions/categories.html", {
126         'categories': Category.objects.all()
127     })
128
129
130 def filter(request, name):
131     category = Category.objects.get(name=name)
132     obj = AuctionListing.objects.filter(category=category)
133     return render(request, "auctions/index.html", {
134         "objects": obj
135     })
136
137
138 @login_required
139 def comment(request, id):
140     if request.method == 'POST':
141         auctionListing = AuctionListing.objects.get(id=id)
142         user = request.user
143         commentValue = request.POST["content"].strip()
144         if(commentValue != ""):
145             comment = Comment.objects.create(date=timezone.now(
146             ), user=user, auctionListing=auctionListing, commentValue=commentValue)
147             comment.save()
148         return HttpResponseRedirect(reverse("details", kwargs={'id': id}))
149     return HttpResponseRedirect(reverse("index"))
150
151
```

Evidence for Completion

```
157            args = Bid.objects.filter(auctionListing=auctionListing)
158            value = args.aggregate(Max('bidValue'))['bidValue__max']
159            if value is None:
160                value = 0
161            if float(bidValue) < auctionListing.startBid or float(bidValue) <= value:
162                messages.warning(
163                    request, f'Bid Higher than: {max(value, auctionListing.startBid)}!')
164                return HttpResponseRedirect(reverse("details", kwargs={'id': id}))
165            user = request.user
166            bid = Bid.objects.create(
167                date=timezone.now(), user=user, bidValue=bidValue, auctionListing=auctionListing)
168            bid.save()
169        return HttpResponseRedirect(reverse("details", kwargs={'id': id}))
170
171
172    @login_required
173    def end(request, itemId):
174        auctionListing = AuctionListing.objects.get(id=itemId)
175        user = request.user
176        if auctionListing.user == user:
177            auctionListing.active = False
178            auctionListing.save()
179            messages.success(
180                request, f'Auction for {auctionListing.name} successfully closed!')
181        else:
182            messages.info(
183                request, 'You are not authorized to end this listing!')
184        return HttpResponseRedirect(reverse("details", kwargs={'id': itemId}))
185
186
```

Evidence for Completion

```
187 @login_required
188 def watchlist(request):
189     if request.method == 'POST':
190         user = request.user
191         auctionListing = AuctionListing.objects.get(id=request.POST["item"])
192         if request.POST["status"] == '1':
193             user.watchlist.add(auctionListing)
194         else:
195             user.watchlist.remove(auctionListing)
196         user.save()
197         return HttpResponseRedirect(
198             reverse("details", kwargs={'id': auctionListing.id}))
199     return HttpResponseRedirect(reverse("index"))
200
201
202 @login_required
203 def watch(request):
204     user = request.user
205     obj = user.watchlist.all()
206     return render(request, "auctions/index.html", {
207         "objects": obj
208     })
209
```

Requirements

System Requirements

- Laptop (Programming device)
- Internet
- GPS
- Smart Android Phone

Personal Requirements

- React Native, Python , Firebase
- Clean dataset to train Machine Learning model
- Train Machine Learning algorithm correctly

Summary

- Combination of agriculture , entrepreneurship and technology
- Motivational Gamification Model to make users always involved.
- Image processing model to find diseased crops and predict appropriate fertilizer solutions.
- Machine learning model to predict best crops according to sales in the auction functionality.
- IoT sensors to get data real time according to the location/town.

References

- H. S. Rohitha Rosairo, David J. Potts, A study on entrepreneurial attitudes of upcountry vegetable farmers in Sri Lanka, Vol. 6 Issue: 1, Journal of Agribusiness in Developing and Emerging Economies, 2016.
- S. Mellon-Bedia,d, *, K. Descheemaekerb , B. Hundie-Kotua , S. Frimpongc , J.C.J. Groot ,Motivational factors influencing farming practices in northern Ghana , 2020
- Arun Kumar, Naveen Kumar and Vishal Vats , "EFFICIENT CROP YIELD PREDICTION USING MACHINE LEARNING ALGORITHMS", International Research Journal of Engineering and Technology (IRJET) Volume: 05 Issue: 06 | June-2018 e-ISSN: 2395-0056 |p-ISSN: 2395-0072, 2018.
- Leisa J. Armstrong and Sreedhar A. Nallan, "Agricultural Decision Support framework for visualization and prediction of western Australian crop production", - IEEE 978-9-3805-4421-2/16 – 201
- C. S. Herath, "The impact of motivation on farmers decision making on technology adoption with reference to Sri Lanka and the Czech Republic," *Knowl. Manag. Innov. A Bus. Compet. Edge Perspect. - Proc. 15th Int. Bus. Inf. Manag. Assoc. Conf. IBIMA 2010*, vol. 2, no. November 2010, pp. 790–801, 2010.
- N. Dobryagina, "Agricultural Entrepreneurship Motivation Policies: European Union Experience and Decision Theory Application," *Int. J. Rural Manag.*, vol. 15, no. 1, pp. 97–115, 2019, doi: 10.1177/0973005219834739.
- H. R. Rosairo, D. J. J. J. o. A. i. D. Potts, and E. Economies, "A study on entrepreneurial attitudes of upcountry vegetable farmers in Sri Lanka," 2016
- J. Kim and J.-W. Lee, "OpenIoT: An open service framework for the Internet of Things," in *2014 IEEE world forum on internet of things (WF-IoT)*, 2014, pp. 89-93: Ieee.
- C. Akshay *et al.*, "Wireless sensing and control for precision Greenhouse management," in *2012 Sixth International Conference on Sensing Technology (ICST)*, 2012, pp. 52-56: IEEE.


Thank
You
2021-090