**FarmBuddy - Documentation**

Folder Structure:

Capstone-|

-|- views -|

-|- index.ejs

-|- login.ejs

-|- register.ejs

-|- settings.ejs

-|- .env

-|-Crop\_recommendation.csv

-|-dataBaseValueSimulator.py

-|-package-lock.json

-|-package.json

-|-passport-config.js

-|-server.js

-|- main.py

* Framework used:
* Node js
* Express js
* Flask
* Libraries used in node app:
* bcrypt
* passport
* express-flash
* express-session
* method-override
* firebase
* request
* open weather map
* Libraries used in flask app:
* pandas
* sklearn
* request
* main.py:
* Pandas, flask and sklearn libraries are imported.
* dataset is read from the storage.
* features and target is declared
* dataset is split into training and testing dataset
* RF classifier is declared and trained
* Accuracy score is calculated
* Flask app is created
* Declaring route for API call that receives a JSON object as a parameter in the form of an HTTP request
* First the argument is extracted and converted into an array
* Then it is passed through the classifier and the recommended is returned with a 200 response code
* App runs on Port 5000
* server.js: This is the entry point for the application. All the views are rendered and redirected using this file. This code can be divided into four modules:
* Authentication
* Open Weather API
* Random forest classifier
* Firebase Real-time Database
* Authentication Module:
* Authentication module utilizes the bcrypt, passport, session and flash library to create a login and register page in which the user is required to provide his/her name, email and password. Which is then stored in the app.
* Upon logging in the user is taken to the settings page where he/she has to enter the factors for their field
* Upon successful completion of the forms the user is redirected to the dashboard where they are allowed to monitor their field data in real time.
* Open Weather API:
* An API key and city is declared, using which a URL is generated
* This URL is used to send an HTTP request which returns a JSON object
* JSON object is used to collect the temperature, humidity and pressure of the location and passed as a response variable to display on the dashboard.
* Random Forest Classifier:
* This part of the code sends an HTTP request to the flask app by encoding the parameters in an URL
* Once the user submits the settings form this HTTP request is sent in which all the seven factors are encoded in the URL
* The flask server evaluates the request and returns a suitable crop for the user.
* This crop is logged and user is redirected to the dashboard.
* Firebase Real-time Database:
* An app is created in the Firebase console on (https://console.firebase.google.com/).
* Which generates a configuration JSON object that is used to register the web app to the database.
* Using firebase.initializeApp and the firebase configuration object and instance of the firebase app is created which is then later used to fetch data from the database.

Use of other Files:

* Package.json : it records important metadata about the project
* package-lock.json: to keep track of the exact version of every package that is installed so that a product is 100% reproducible in the same way even if packages are updated by their maintainers
* passport-config.js: Declares basic functionality of the passport and defines the function that initializes the passport instance.
* dataBaseValueSimulator.py: Artificially simulates the values various sensors will generate and uploads them to the database.

**Views/ login.ejs:**

This is a view created using embedded JS which allows us to dynamically generate web pages.

This is the page you will be taken to first when visiting the web app.

The CSS styles used for this page is coded using internal CSS. A form is created which takes the email and password from user to login and displays a link which will take the user to the registration page in case they don’t have an account.

Upon logging in the user is redirected to the

**Views/ register.ejs:**

The CSS styles used for this page is coded using internal CSS. A form is created which requires the name, email and password from the user, upon submission the user will be created and they will be redirected to the login page. A hyperlink to the login page is provided in case the user already has an account.

**Views/ settings.ejs**

The CSS styles used for this page is coded using internal CSS. A form is created which requires the user to input seven factors that are required to predict an ideal crop. Upon submission the HTTP call is made to the flask server and the user is redirected to the index.ejs page.

**Views/ index.ejs**

This is the Dashboard of the application. The CSS styles are done internally and JS code for firebase is written internally so the calls can be made in front-end without utilizing server resources. Because firebase provides Back-end as a Service.

Using the Open Weather API calls the web-page displays the temperature, humidity and pressure of the location. using the firebase app and snapshot method the twenty latest values of N, P, K and pH are extracted from the database and plotted on four separate graphs using charts.js

Logout option and an option to change the settings is also available in the far top-right corner of the page.