

# CS50P Final Project

## AgRoll : Agricultural Data and Payroll Management System



*By Kelvin Mafurendi*

Video Demo: [AgRoll Software Demo](#)



## Description

This project is an agricultural management system that combines weather forecasting, staff management, and crop monitoring into a single platform. It uses fake data generated to simulate real world entities like crop yield, pests, people, as well as weather data and images obtained from the internet via APIs.

All these are then manipulated and visualised to ideally provide insights and support decision-making for agricultural operations. The project assumes the user has already carried out the setup process outlined in the [Abstract](#) below and that the system has been running for over 3 months. Also, for better user experience the project uses a Graphical User Interface instead of a Command Line Interface.

## Abstract

In a practical setup, the user (most likely a farmer) would be able to do the following:

- Set up their user profile. This includes their user credentials (name, username, email address, password, user role etc.)
- After setting up the profile, the user could then load their own custom parameters. These would include:
- The names/varieties of the crops that they produce

- Their farm location (and potentially the farm map/map data) for more personalised weather details and analysis
- The pests that may be prevalent on their farm with respect to the various crops.
- Have a profile created for each employee on the farm. This would also include syncing an employee's biometrics e.g., fingerprints and face to their profile. This would be useful for Remote Clocking.
- This would all be saved to a non-flat file database
- The software would be accompanied by a Remote Clocking hardware via which staff members would clock in or out by scanning their biometrics under supervision of course. As soon as the employee clock in/out an update is transmitted to the software and that person's record is updated.
- Based on the employee's pay grade, hours worked, incentives received as well as deductions, the software automatically computes the employee's wage at the end of the month/week. With proper integrations with online payment systems, the software would be able to transfer wages to the employee's bank account.
- The farm would then be able to track which employee is currently at work or not and alerts would be triggered if any anomalies are detected by software.
- The farmer would be able to view records of yields or pests spotted or other farm operations. These records would be displayed statistically in order to aid decision making or inference. They can also download records for further analysis.

## Project Scope

However, due to the fact that this application requires a lot of moving parts; gradual data generation as well as collaboration with external hardware devices and the participation of many people which are beyond the scope of this course, I have used a few python libraries to simulate the data that can be used as proof of concept. Thus, this project is limited to the following functionality:

- **Staff Management**



The app has got 3 distinct tabs; Weather, Fields and HR (Human Resources) which is the staff management tab.

- The current farm has got a hundred employees and the user can view the employee database in tabular form.

Staff Table

Name	Role	Department	Pay Grade	Is Working
filter data...				
Thomas Marshall	Researcher	Research & Development	B4	Online
Karen Dickson	Machine Operator	Transport	E	Offline
Dennis Chen	General Hand	Propagation	A	Offline
Keith Smith	Machine Operator	Engineering	A2	Offline
Taylor Vazquez	Manager	Propagation	B1	Offline

« < 1 / 20 > »

- When user clicks the name of a particular employee, an employee card pops up and the user can then view more details about that particular employee.

Employee Details



Staff Number: 8617434  
 Department: Research & Development  
 Role: Researcher  
 Grade: B4

Overall Uptime:  
 2.38092 Hours

Thomas Marshall  
 Online

Gross Wage: \$135.71  
 Deductions: 10%  
 Incentives: 1%  
 Current Net: \$123.50

Clock In

Clock Out






- On the employee card, the user can manually clock an employee in or out.
- The card also shows payroll details like the cumulative number of hours worked by the employee in the current work season (i.e., week or month). If the employee got any incentives along the way, the % incentive as well as deductions are also shown on the card. Finally, the employee's current net wage/salary is stated. These values are always updated whenever an employee clocks out. \* >It is important to note that in this case the user/farmer is responsible for clocking in/out their employees since remote clocking is not implemented. In practice, this role can be delegated by having many users (possibly managers/supervisors) log into the web app from their various devices through role-based access and then clock in/out their own subordinates.

- The employees table also allows the user to browse the staff database using filters. For example, the user can filter the table data to only display people who are currently working i.e., clocked in. Or They can search for a particular worker by name. They can also filter the workers table by pay grade role, and department.

Employee Details

Select Staff Name

Staff Table


Name	Role	Department	Pay Grade	Is Working
 filter data... 				 Online
Thomas Marshall	Researcher	Research & Development	B4	Online
Jason Miller	Manager	Pest Control	B1	Online
Jennifer Bates	Supervisor	Security	B4	Online

- Also, each employee has got a profile photo on their employee card. This was originally intended to be the employee's picture but because of the random nature of the images returned by the Pixabay API regardless of my attempts to filter the response, some pictures would be inappropriate for use as employee's photos. Some images would actually be a face of a cute animal. Also, the random nature of employee details generated by the program, some profiles would end up having females with male profiles and vice versa which would be inappropriate. Therefore, these images are rather profile pictures than face IDs.

Change Theme
Weather Fields HR
AgRoll Systems
Sign Out

---

Employee Details



Jennifer Bates  
Online

Staff Number: 1438750    Gross Wage: \$142.37

Department: Security    Deductions: 15%

Role: Supervisor    Incidents: 1%

Grade: B4    Current Net: \$122.44

Overall Uptime:  
2:27:287 Hours

Clock In
Clock Out

---

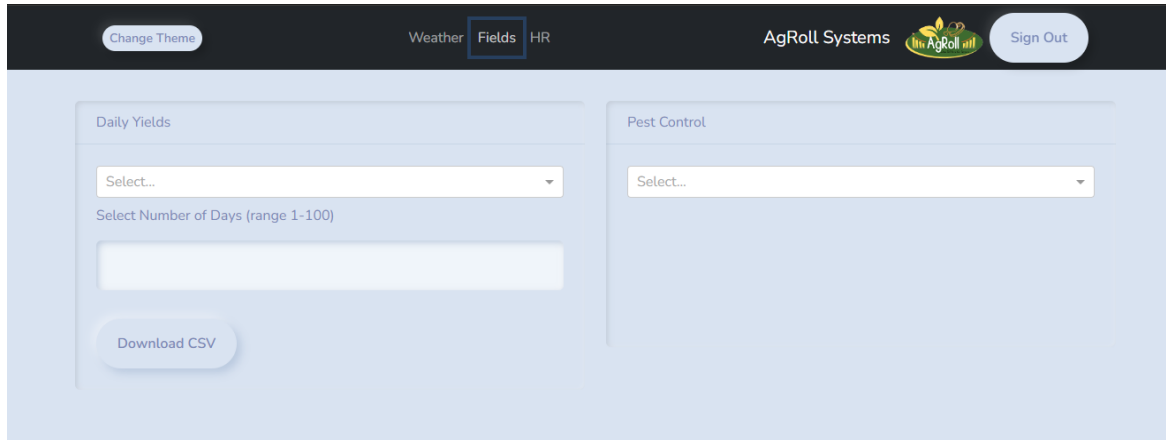
Staff Table

Name	Role	Department	Pay Grade	Is Working
Filter Data...				
Jennifer Bates	Supervisor	Security	B4	Online
Shannon Watkins	Agriculturist	Administration	B3	Offline
Frank Foster	Machine Operator	Storage Services	A3	Offline
Brittney Johnson	General Hand	Storage Services	C	Offline
Daniel Martinez	General Hand	Engineering	B	Offline

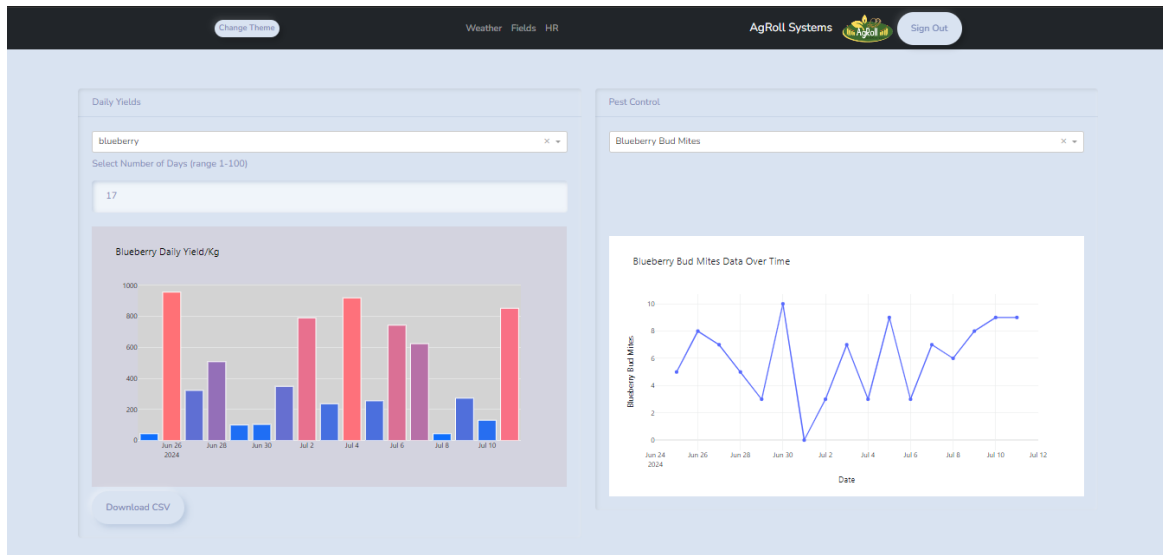
<
1 / 28
 >

- **Crop Monitoring**

In the Fields tab, the user can get report of their crop yields and any pests that might have been observed per each crop.



- From the `Yields Data` dropdown, the user gets to choose the crop they intend to review.
- They then choose the number of days prior to the present day whose records they want to see.
- A bar graph will pop up showing the requested records.
- On the `Pest Sightings` dropdown, the user gets to view corresponding pests observations. Due to the randomness of the pest data generation, there will most probably be no meaningful trends or correlation between yield data and pest infestation. \* >Note that, Yield data must first be defined before the pests results can be returned on the `Pest Sightings` dropdown and graph
- For any given crop, the user can choose from 3 up to 100 days of records to view and analyse.
- The user can download a CSV document of these observations for further analysis.

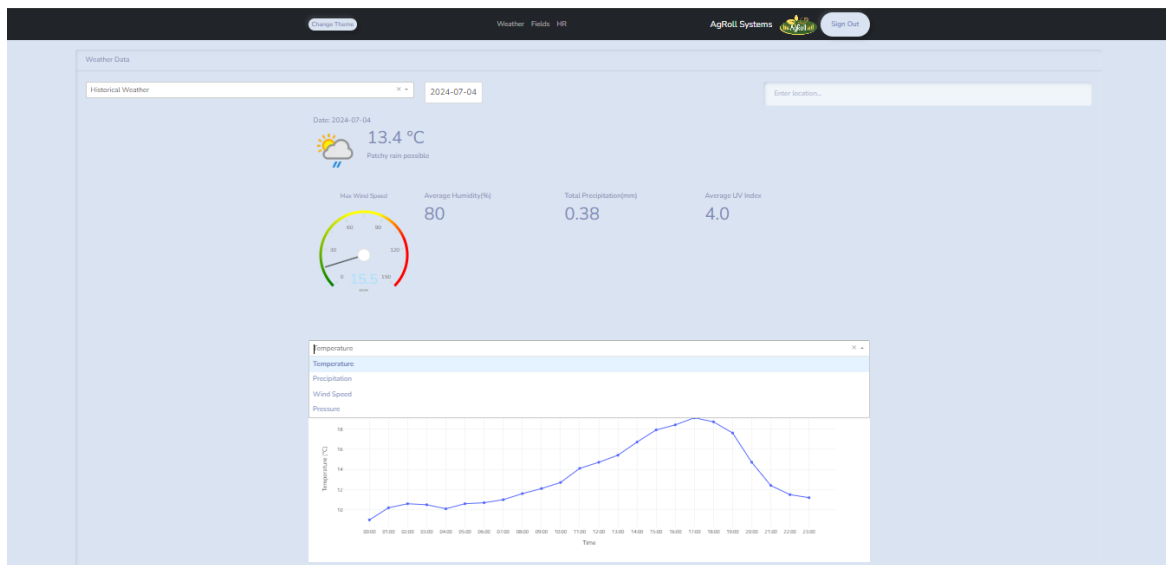


- **Weather Forecasting**

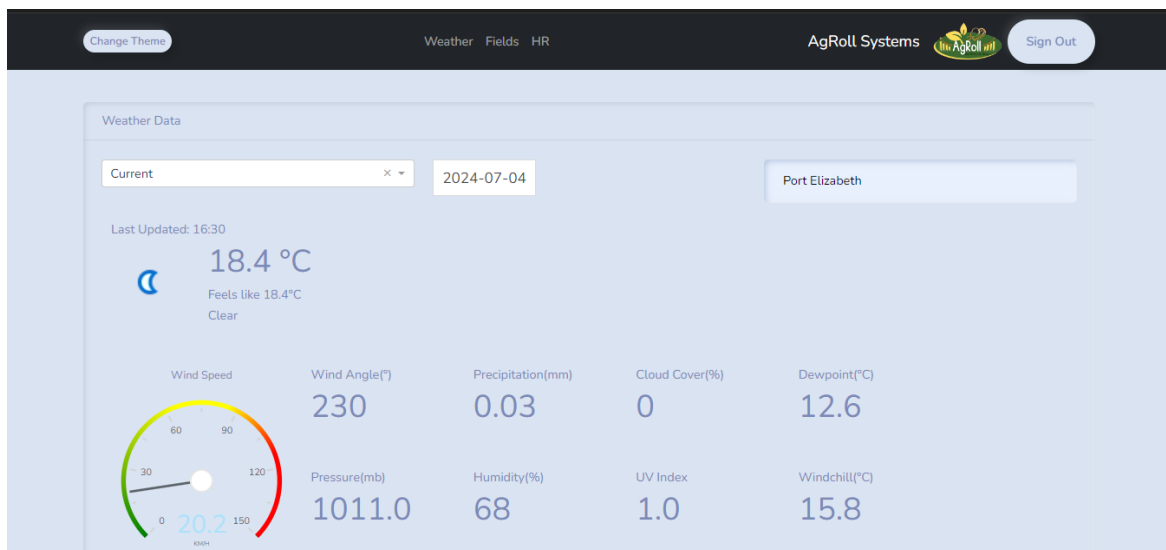
To help the farmer with future planning and to help them to analyse past trends on their farm and how the weather will or did affect their operations, AgRoll gives the user access to historical and current weather information. The weather forecast feature only caters for one day ahead due to the limitations of the free API that I used. To get beyond that, I would have had to purchase a subscription.

The screenshot shows the 'Weather Data' section of the AgRoll Systems dashboard. It includes a 'Select...' dropdown menu, a date input field showing '2024-07-12', and a text input field labeled 'Enter location...'. Below these is a green button labeled 'Select Weather timeframe...'. The top navigation bar is identical to the previous screenshot.

- Once in the tab, the user can select a timeframe (current weather, past weather or future weather) for which they want to view the weather details
- If the option is either past or future weather, the user can select a date on which they want to focus.
- For past weather, the API I used is limited to 7 days in the past.



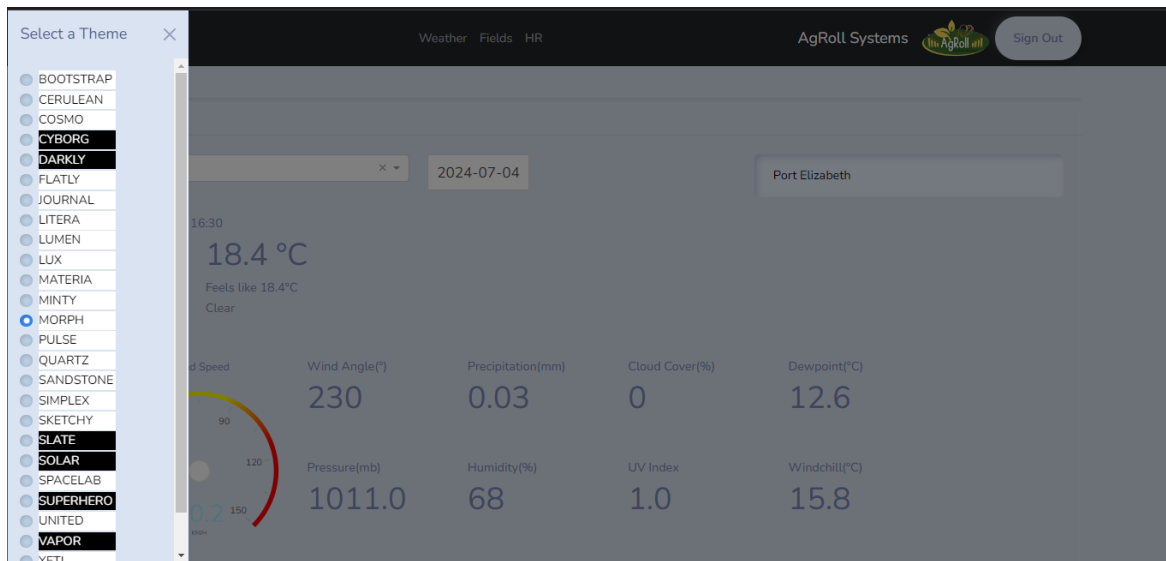
- The past/future weather returns the average values for that day as well as a graph detailing how the weather changed or will change during the course of the day.
- The current weather option returns almost real-time weather data which is updated every 15 minutes.



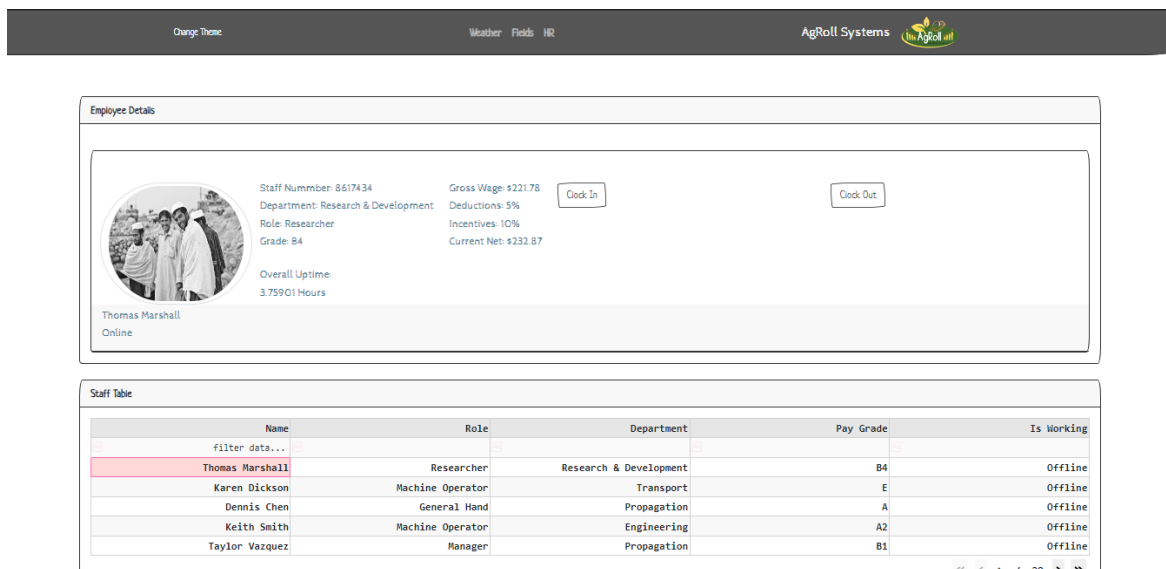
- **In ALL weather selections, the user should be sure to input their target location lest they get weather details from an unexpected location**
- **Theme Options**

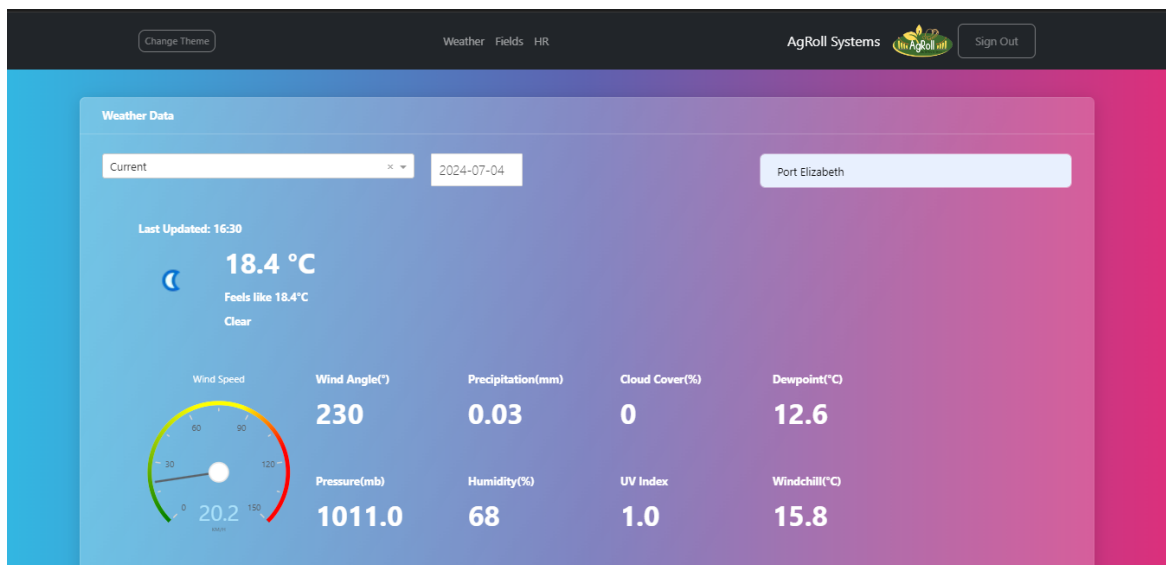
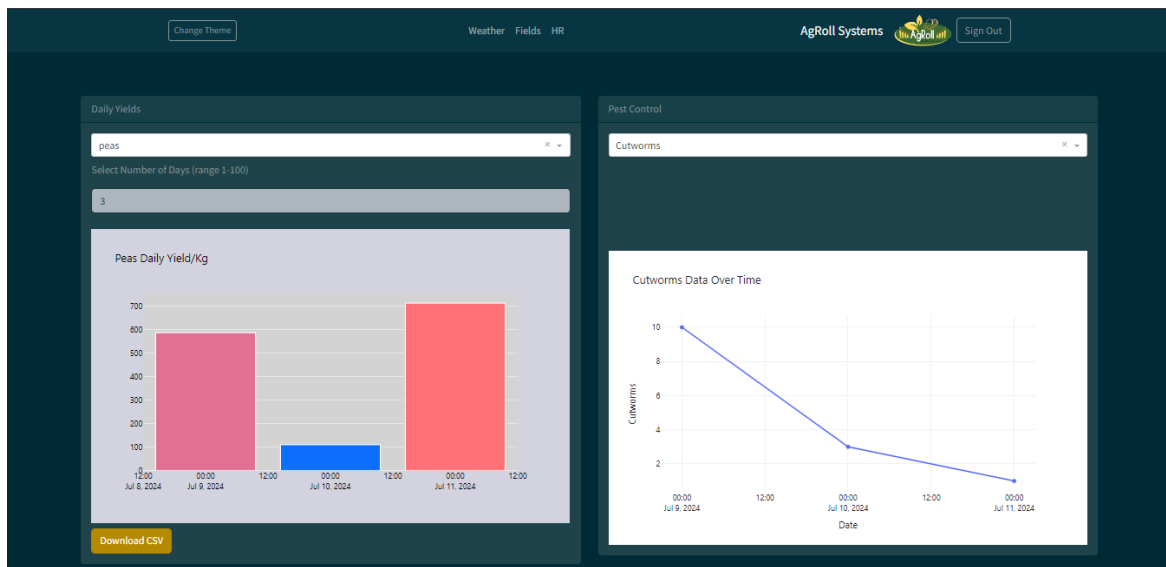


For improved user experience, the user has the option to customise the appearance of the app by changing the theme. There is a collection of 26 themes to choose from and the default for this app is MORPH which just happened to be my favourite. When the user clicks the Change Theme button on the navigation bar, they get to see all the available options.



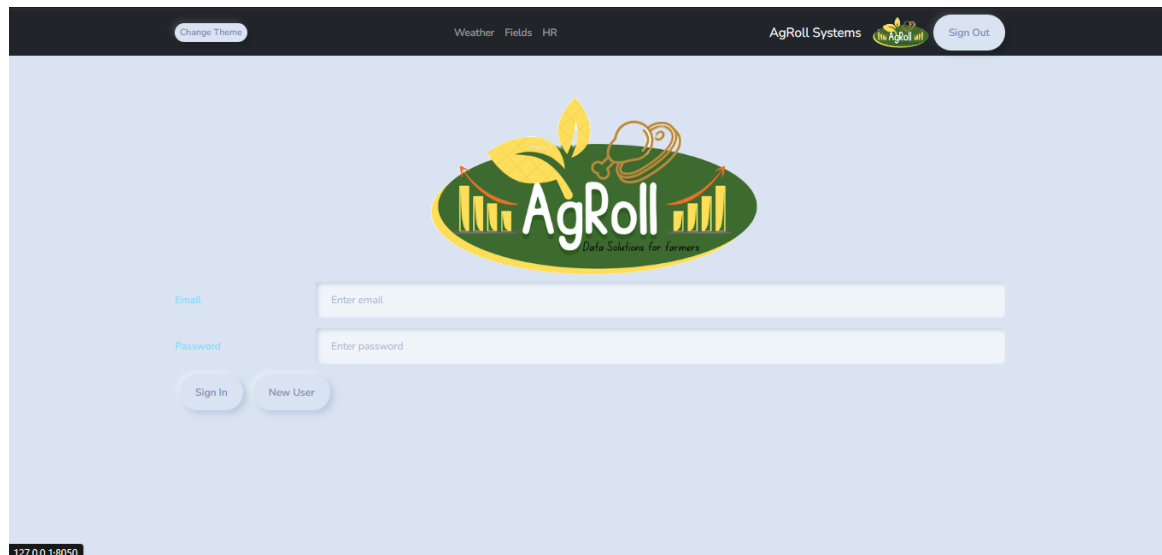
Below, are the different tabs viewed with different themes:



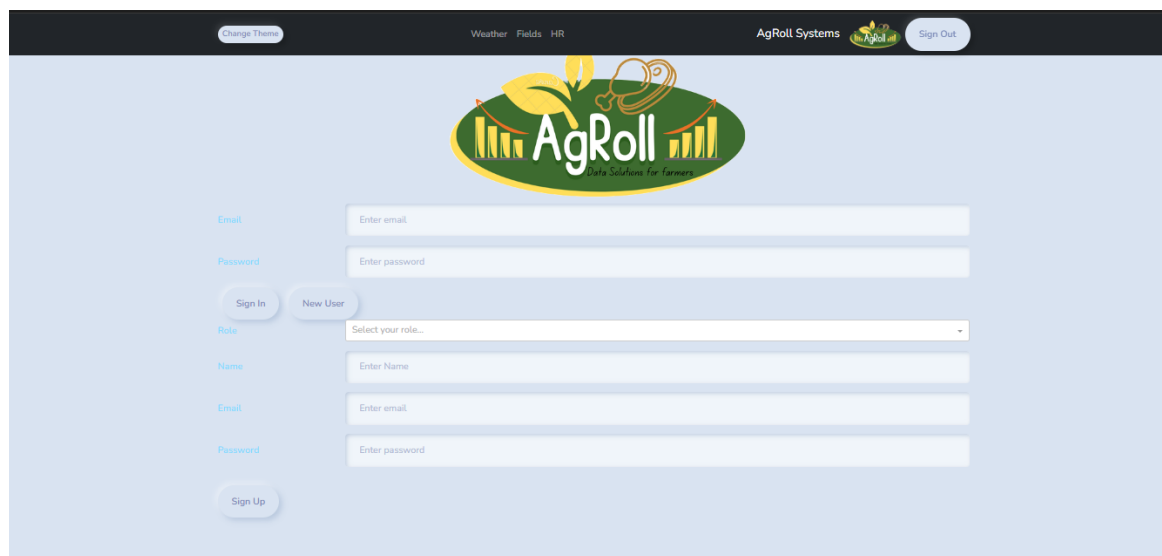


## • User Authentication

Ideally, this app ought to use role-based access, however I did not implement that.



Nevertheless, I have implemented a basic user authentication where an existing users (existing in the flat file database) can sign in and new users can sign up.



Also, if a user tries to sign in and they are not in the database, they are taken to a sign-up page.



Once logged in, the user can through the `Sign Out` button, log out of the app. The Users database gets updated every time a user sign in or out as well. Unless a user is signed in, they don't have access to any data from the different tabs. If they click the tabs, they simply won't respond.

## The Source Code

To best express my project idea using a Command Line Interface was not such an exciting idea, so I had to look for an entirely Python framework that would allow me to use the skills I learned in CS50P to express my idea. After some research, I stumbled upon Dash. I spent some time studying the documentation, looking for videos on YouTube and practising with simple designs until I got comfortable enough to start building AgRoll. Honestly, I got better as the project evolved and sometimes, I had to undo stuff and re-implement them in a better way.

In this section, I explain the secret sauce behind AgRoll, the Source Code!

## Project Structure

Initially, I wrote the entire app layout in a Jupiter notebook as I experimented with the various features of Dash. Then I transferred from the notebook to VS Code where I added call-back functions and the entire app was in one huge document. However, for good practice and professionalism, I had to restructure the code into meaningful modules. Not an easy task, but I eventually got everything working out. Below is the structure I eventually settled for:

```
Project/
├── assets/
│   ├── styles/
│   └── images
├── database/
│   └── csv files
├── api_functions.py
├── classes.py
├── config.py
├── project.py
├── README.md
├── requirements.txt
└── test_project.py
```

- The assets folder contains a folder for bootstrap components for the `dash_bootstrap_components`. It was not really necessary to have a copy as the web app can still have access to the components online. This was just a backup. The folder also contains the images that I have used in this documentation as well as the Logo image for AgRoll.
- In the database folder I the AgRoll web app reads and stores copies of the documents that it uses for its operations in csv format. This forms a sense of data persistence instead of using memory. The documents include staff records, crop data as well as the `Userbase.csv` which keeps track of everyone who logs into AgRoll. Had it not been an entirely Python project, this feature would have been a relational database instead of flat files.
- `api_functions.py` : Is a module of custom functions, specifically those that perform API requests. I have two such functions.
- `get_weather` : This function takes in location and date as well as a timeframe (current, history, future) as inputs and performs API requests as per [weatherapi.com](https://openweathermap.org/api) official API documentation.
- `pix_list` : Takes in as input `n` which is the number of images required and returns a list of URLs of the individual images from [Pixabay](https://pixabay.com/) again as per their official API documentation.
- `classes.py` : This is a python file containing two classes that I used to abstract two entities, a `crop` and an `employee`. I also made use of the `Faker`

Python library to generate realistic data for names IDs and other details that would have been difficult to simulate.

- `Crop` : A crop basically has a name, a list of pests that are associated with it and a `yield` quantity which quantify the amount of crop harvested.
- `Staff` : Among other details, a staff member has got a full name, pay grade, and should be able to clock-in or clock-out. Also, they should be of work-going-age (18 -65)
- `config.py` : To avoid flaunting API keys all around the code, this small module was necessary to store all the API keys and other modules would have to request for keys form this module. Best practices!
- `project.py` : This is the main file. In this file is the layout of the dash app, as well as all the call-back functions and three more supporting functions that support the dash app for staff or crop data generation.
- `test_project.py` : In this module I perform unit tests on both the `api_functions` as well as those within `project.py`. All the tests I performed passed and I understand there is still room for more rigorous testing.
- `requirements.txt` : This file contains a list of all the pip-installable libraries that are required for this app to run. The library versions used are also specified.

## Installation and Setup

To run this code;

- Navigate to the directory where this `Project` folder is found. And navigate into the folder itself or simply type the command

```
```\n\ncd Project
```

- In the terminal, run

```
```\n\npip install -r requirements.txt
```

This ensures that you have all the required libraries installed if they do not already exist on your machine.

- Once the installation is done, again in the terminal, run

```
python project.py
```

Alternatively, click to open `project.py` and just click on the run python program button in your IDE. Either way, if everything goes well, this message will appear in the terminal

```
PS C:\Users\mafur\OneDrive\Documents\DS50P_Final Project\Project> python project.py
Dash is running on http://127.0.0.1:8050/

* Serving Flask app 'project'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:8050
Press CTRL+C to quit
```

Click on the link and you will be redirected to your default browser where the app will be running.



## Technologies used

I have used the following non-built-in Python libraries to do some of the heavy lifting:

- `Faker` : I used faker to generate fake yet realistic data about people and crop yields
- `Dash` : This is the framework on which the entire web app is based.
- `Pandas` : I used pandas to manipulate the data and generate datasets.
- `requests` : I used this library to handle API requests.
- `json` : This library was used mainly to extract relevant data from API responses.
- `dash-bootstrap-components` : This library adds a bootstrap touch to dash apps enhancing the overall appearance.
- `plotly-express` : Instead of using Matplotlib or seaborn, this library enables graphs that are interactive.
- `dash-daq` : This library also adds a few more components to the dash framework. I had started with this library for my major displays but I then migrated to dbc components. I only kept this to display the wind speed on the weather tab.
- `dash-bootstrap-templates` : This library enabled me to allow my users to customize the app theme. This was a game changer for me as I am not yet strong with UI/graphics design.
- `pytest` : I used pytest to perform unit tests for my project.

## Bugs

There are few bugs that I have not solved yet.

- `Back to back sign in` : When a user signs out after signing in, if they try to sign back in immediately, they do get back in but then they are not able to access the various tabs, as if they are not logged in. This is due to unexpected changes in a global variable that I am using to track the signing in and out of a user. However, after signing out if the user refreshes the browser or simply clicks the AgRoll logo on the navigation bar, they can sign in flawlessly. I am still trying to fix this bug.



- `Staff table filtering` : When the user filters the staff table, when user clicks on the name of an employee, instead of that particular employee's card to pop up, the corresponding employee on the unfiltered table pops up. Although the data displayed on the table changes, I still need to make sure that the underlying data is also filtered.

## Future Improvements

Adding the following features would greatly improve the functionality of AgRoll:

- `Role based access`
- `Remote Clocking and IoT integration`
- `Relational database integration`
- `Allowing users to modify staff and crop data`

**This was CS50P! (2024-07-13, 17:31)**

# Table of Contents

AgRoll Software Demo	1
Abstract	2