Application to help small scale farmers in East Africa attain quality farm produce

Kiningu Stephen

MSc Information Technology project report, Department of Computer Science and Information Systems, Birbeck College, University of London 2022

12/09/2022

Contents

Declaration

This project is fully my original work.

Signature ………………… Date……………… kiningu stephen

Acknowledgement

I cannot fully express my appreciation for my friends in the field of IT for their unwavering support. I would want to thank them for providing their time and experience in guiding me through this project. Also, without the assistance of my classmates and friends, I would not have been able to complete my assignment. I am grateful to all of you for allowing me to spend some time away from you in order to conduct research and compose my report. Additionally, I would like to thank my parents and sister for their constant support during this journey.

Abstract

The application to help small scale farmers in East Africa attain quality farm produce is a system developed with the ability to send planting procedures and advisories to farmers via text message while also taking into account the farmers' preferences regarding agricultural practices and geographic locations. To put it more plainly, this website was made available to the agricultural stakeholders and related sectors within agriculture in all parts of the country. This gave them the ability to share information with farmers across East Africa on a wide range of topics, and it was made available to them through this portal. Concerns include providing information to farmers on the ideal times to plant, what to sow, when to apply pesticides, when to harvest, and where to market their products. In addition, farmers can find information on a wide variety of subjects by using the internet.

This system intends to close the gap by increasing engagement between farmers and the agricultural stakeholders, such as farmer societies and the Ministry of Agriculture, which will facilitate planting procedures for French beans, potatoes, and avocados. This will ensure that small-scale farmers improve the quality of their products and, as a result, gain access to global markets.

2 Introduction

Building application to help small scale farmers in East Africa attain quality farm produce is the major objective of this project, and I want to accomplish this objective by making use of the skills and information that I have acquired via the MSc Information Technology course. In addition to this, one of my goals is to acquire the expertise necessary to manage a project from its beginning stages all the way through its completion and to be able to write a report that describes the construction process in detail.

The purpose of this project report is to outline how I implemented an Automated SMS system for farmers within East Africa. This system allows member state government entities in agriculture and related sectors to transmit information, alerts, or advisories to farmers through SMS while also taking into consideration the farmers' preferences about agricultural techniques and locales. To put it bluntly, this portal was made available to the agricultural stakeholders and related sectors within agriculture in all parts of the country, allowing them to share information with farmers across East Africa on a wide range of topics. Issues include informing farmers of the best times to plant, what to plant, when to apply pesticides, when to harvest, and where to sell their goods. In addition, the web will inform farmers on a wide range of topics.

3 Background

Agricultural extension, also known as agricultural advisory services, plays an essential part in raising agricultural production, enhancing food security, enhancing rural lives, and encouraging agriculture as an engine of economic growth that is beneficial to the underprivileged. However, within the East African context, there is a major challenge. According to the findings of a study conducted by Kovacevic (2020), the majority of the agricultural extension services currently provided in East African countries (Kenya, Uganda, Tanzania, and Burundi) depend on face-to-face interactions between extension officers and small-scale farmers. These countries include Kenya, Uganda, Tanzania, and Burundi. However, due to a lack of available human and financial resources, this method is not a realistic means to provide agricultural guidance to farmers in very remote places or on a large scale.

Even while there has been significant progress made to ameliorate the situation on the extension front, the varied needs of the mass of East African farmers have not been able to be met in their entirety. On a wide scale, it was not possible to meet the specific needs of the farmers, which varied depending on the products they grew and the agro-climatic conditions of their region.

As a result of this, the system that will be developed to be of assistance to small-scale farmers in East Africa will be conceptualized to give a major breakthrough in coverage of farmers and geographical locations in a prompt, precise, comprehensive, and need-based knowledge transfer among the farmers. This will be accomplished by harnessing the benefits of mobile communications in such a way that all sectors such as the ministries involved in agriculture, agricultural research organizations, and other major stakeholders such as market players utilize this platform to not only reach out to the farmers but also to voice their concerns and questions. This will be accomplished by leveraging the power of mobile.

This is an aspect that will ensure the essential areas that need to be addressed among East African small-scale farmers such as quality control and standardization of farm yields are realized. The international food market is constructed on a foundation of standardization (Deichman et al., 2016). Notably, the level of agricultural product standards planted by East African small-scale farmers is very low to meet international standards required by international buyers. The prevailing issue concerns the fact that the small-scale farmers have yet to benefit from facilitation of information that is required to meet these standards. The essential knowledge held by government does not get to the small-scale farmer on time to help them improve on the aspect of standardizing their farm produce, and not all of these farmers are registered to farming societies to assist them gain access to markets for their produce (Deichman et al., 2016). Therefore, there is need for an automation system that will entail the functions of all concerned stakeholders: farmers, the government ministries, research organizations, farmer's society, and markets.

3.1 Problem Statement

This approach aims to address a variety of issues that farmers confront throughout East Africa. One such issue is information on how to plant crops for the worldwide market, such as the supply of potatoes to multinational firms. Standardization and quality control, for example, are two concerns that Kenyan potato growers must address. MNC food chains are constructed on the foundation of uniformity.

In Kenya, there are roughly 800,000 small-scale potato growers (CIP, 2019), and a lack of information on how to standardize their goods suggests that they will continue to miss out on possibilities to exploit new and expanding markets throughout the world. There is a particular need for education on Good Agricultural Practices certifications such as GLOBALGAP (GAP). Globally enforced standards for food freshness, storage, and quality on farms and gardens. This system aims to close the gap by increasing engagement between farmers and the agricultural stakeholders—farmer society, Ministry of agriculture, who will facilitate planting procedures for French beans, potatoes, and avocado to ensure small-scale farmers improve the standards of their commodities and therefore get access to worldwide markets.

Small-scale farmers in Kenya and the rest of East Africa are benefiting significantly from value addition, which provides them with easy markets for their production, extends the shelf life of products, and ensures that people have access to healthy food (Business Daily Africa, 2018). Across East Africa, unfortunately, there are not nearly enough farmers who have the financial resources necessary to invest in value addition. Agriculture is the primary contributor to Kenya's economy, as it is in other East African nations. This sector is driven by small-scale farmers, whose relatively small plots of land supply food to millions of households living in rural and urban areas. This system's goal is to connect farmers with farming society groups that may provide assistance to farmers in gaining quick access to value-adding technologies and, as a result, enhance the overall outcome of their crop.

In addition, research conducted by Giller (2021) reveals that the ability of small-scale farmers in East Africa to sustain their livelihoods is frequently hindered by a lack of access to markets and limited entrepreneurial skills for adding value to their final products, with only a small portion of 11 percent having access to these services. As a result of low income and inadequate infrastructure in rural regions, a significant portion of the crops are wasted on the farms because farmers do not have access to the appropriate market linkages. Growers that want to make a profit off of their goods are forced to engage in business with intermediaries because they have no other option. This approach aims to increase the amount of effort being put in and guarantee that it fills the void left by the lack of market-links by ensuring small-scale farmers are linked to the market, this is in order to radically alter the agricultural environment in East Africa.

3.2 Current Applications Available

At the moment, there is a variety of software that can be used to provide farmers with information that may help them increase the quantity and quality of their harvests. For example, there is an application called iCow that was developed to assist farmers in monitoring the cycles of fertility experienced by their cows. This application is part of a larger concept that envisions the platform being used to give farmers with particular agricultural information on a wide range of issues." iCow is a voice-based application that has grown into an SMS as a result of the progression of technology (text messaging service). The primary reason for this is that the vast majority of the farmers do not make use of cellphones, and the capabilities of SMS regarding retention and sharing are superior.

In addition, there is Twiga Foods, which started operations in the year 2014. Twiga purchases goods from Kenyan farmers and food producers and then passes them on to licensed merchants to sell. In doing so, Twiga ensures that both farmers and sellers have access to an appropriate market. Twiga Foods distributes product to the metropolitan areas of Kenya after first obtaining it from Kenyan farmers who have grown their own fruits and vegetables (Twiga Website, 2022). The marketplace system offered by Twiga is being utilized by more than 30,000 merchants and 3,500 providers at the present time (Twiga Website, 2022). Twiga is a company that takes great pride in its transparent business practices and its fast delivery of high-quality items. The system gives small scale farmers with the peace of mind that the things they produce will bring them a profit. Through its commitment to openness and provision of a secured market, Twiga Foods facilitates the sellers and buyers of Kenyan food by ordinary Kenyan farmers and merchants (Twiga Website, 2022).

Additionally, there is a mobile app called DigiCow. Established by Farmingtech Solutions, a technology firm that focuses on agricultural data management, is the company that developed DigiCow with the intention of providing small scale farmers with a resource (DigiCow Website, 2022). Farmers in Kenya were forced to rely on educated assumptions and imprecise projections prior to the advent of services such as DigiCow; however, today they are able to make better decisions on actual data, giving them a significant advantage. Because of this program, users are able to base their decisions on genuine knowledge rather than speculation (DigiCow Website, 2022). The application gives users access to a range of services, such as, but not limited to, digital tracking of feeding, insemination, and milking; notifications for significant dates and statistical results; and virtual instruction and conversation forums for farmers to interact with one another. In April of this year, DigiCow accomplished a significant new goal and milestone (DigiCow Website, 2022). The World Bank's Innovative Agricultural Technologies competition was won by the Farmingtech Solutions team from Kenya with their cutting-edge invention, DigiCow (DigiCow Website, 2022). With the use of the DigiCow app, farmers are now able to retain large datasets and make decisions based on accurate information.

3.3 Aims and Objectives

This application seeks to ensure the timely facilitation of crop planting procedures to farmers on a weekly basis and thus aid them produce quality crops. The farming society will register farmers into the system. Registered farmers will be able to receive automated SMS on aspects such as reminders to plant, seeds to use, when to fertilize, when to harvest, and the markets available for their produce. The role of the farming society will be to communicate to the farmer’s new information from research carried out by research organization’s on aspect such as new seedlings, chemicals, any new potato diseases and their need to spray to protect the standards of their farm yields.

Also, the farmer’s society acted as a link to the markets such as with major global markets and the standards they require for their potato produce. The society will then communicate these standards to research organizations and facilitate farmer training on the better crop standards required by small-scale farmers in order to meet international standards. The farmer’s society, as such, will be aware of the standards required by the global trade market and will work to add value to potato yields by farmers in order to secure international markets for its local small-scale farmers. On the whole, this system will be beneficial to farmers as it will not only facilitate them with the required information on a timely basis, but it will link them to all these essential stakeholders and thus improve on their farm yield standards.

3.4 The implication of this Project

Farmers are driven to satisfy quality standards of crop production under pressure from the global market. Producers, businesses, and customers benefit from increased compliance with such standardization regulations. This implies that farmers get a better return on their produce investment, business gets a better product, and consumers get peace of mind knowing their food is safe (Xia et al., 2014). However, small-scale farmers have difficulty getting access to the resources they need to grow their crops and expand their markets, such as high-quality seedlings, fertilizers, other inputs, and training and equipment. Many farmers are held back when it comes to farming techniques and a lack of market knowledge or product quality control. With this system in place, East African farmers will be able to produce enough high-quality crops for processing to meet the needs of both domestic and international clients.

The number of mobile phone devices that were able to access mobile networks reached 59.0 million as of September 2021, according to a study by Standard Media (2021). Of this total, 33.0 million were feature phones, while 26.0 million were smartphones. Given that the World Bank and the African Development Bank estimate that there are 650 million mobile subscribers across Africa, this represents around 9.07 percent of the total population of the African continent that possesses a smartphone. This necessitates a deeper investment of the resources needed to guarantee that this system is practicable, that it can be put into action, and that it will provide improved outcomes.

4 Specification

The main requirement for the system is to offer Automated SMS information to farmers across East Africa. The functional and non-functional requirements are outlined below:

4.1 Functional Requirements

4.1.1 User Authentication

- The user should be able to sign onto the website.
- Farmers should be able to login into the system with email and password.

1.1.1. Registration

- Farmers should be able to register and get automated SMS notifications.
- Enable new users to register to the automated SMS service system
- Enable a registered user to update his profile which includes his location, farm type and crops preferred for planting.

1.1.2. Crop selection

- Farmers should be able to choose from variety of crops to plant: Avocado; French Beans; and Potatoes.
- Farmer should be able to receive SMS notifications on the information required to plant the crops selected

1.1.3. Planting period selection

- Farmer starts date whenever the data h/she has started scheduling
- Farmer should be able to receive SMS notification on the planting procedures for crops selected.
- Farmers should be able to send and receive SMS.

4.2 Non-Functional Requirements

4.2.1 Operational Requirement

The application need to function properly on every web browser.

4.2.2 Performance Requirement

The application must to be dependable and should not take too much time. The user and the system should have a quick and easy time interacting with one another.

4.2.3 Security

A super user is the only type of user who can view the profile information of other users and edit that information if necessary. The website will have the ability to protect itself against the most prevalent types of attacks.

5 Design

5.1 Database design

Below are the specs for our four database tables. There are foreign keys because the tables are connected.

Table design for the farmer

Key	**Name**	**Description**	**Type**	**Compulsory**	**D/value**	**Input**	**Update**
Primary key with Auto Increment	Id	The farmer assigned during registration	Integer	Yes	N/A	At registration	N/A
	Email address	Farmers email address	Email	Yes	N/A	At registration	N/A
	First name	Farmers first name	Character String	Yes	N/A	At Registration	N/A
	Second name	Farmers Second Name	Character String	Yes	N/A	At registration	N/A
	Idn	Farmers identification number	Integer	Yes	N/A	At Registration	N/A
	Phone number	Users actual phone number included with the country code e.g. Kenya (+254712345678)	Integer	Yes	N/A	At Registration	N/A

	crop_id	Crop_id is the identifier of the crop that a farmer selects once logged in and obtains by filling out the schedule for that crop.	Integer	No	0	N/A	N/A
	Start_date	The date in which the user has decided to start scheduling	DateTime	Yes	N/A	At schedule date through the form in figure 11	N/A
		schedule a procedure					
	Is_scheduled	Farmers scheduled line up	Boolean	Yes	N/A	At schedule date	N/A
	Country	Farmers country location	String	Yes	N/A	At Registration	N/A
	Village	Farmers country village location	String	Yes	N/A	At Registration	N/A
	sizeofland	Farmers actual size of land	Integer	Yes	N/A	At Registration	N/A
	Password	Farmers password choice	String	Yes	N/A	At Registration	N/A

The procedures to be carried out are kept in a database where each tables are linked via foreign keys. The crop_id is the id which is given to the farmer when s/he registers and then later selects the type of crop in which s/he wants to start scheduling. Crop_id is triggered when a farmer selects the type of crop s/he wants to plant. A farmer can only schedule one type of a specific crop at a time since the crop scheduled completes a cycle of sending scheduled messages to users before the farmer may choose another type.

<u>Table for procedure</u>

Tablename is **procedure**

Key	**Name**	**Description**	**Type**	**Input**
Primary key	Id	The id assigned to each procedure or crop to be planted	Integer	Only at setup
Relationship	weeks	Has a relationship with weeks' table?	N/A	Only at setup
	maintitle	The of the crop to be scheduled e.g. planting potatoes	String	Only at setup
	Description	The description of the crop to be scheduled	String	Only at setup

<u>Table design for Weeks</u>

Table name is **week**

key	**Name**	**Description**	**Type**	**Input**
Primary key	Id	The id assigned to each week inside a procedure to be followed	Integer	only at Setup
Foreign key	Procedure_id	This is the foreign key that links the table week with procedure through the id of the procedure	Integer	Only at Setup

Title	title	This is the title of the procedure inside a week e.g. the title of what you will do inside week2 of planting procedure(maintitle) is land preparation	String	Only at Setup
Relationship	activities	This is the variable that links the table week with the table activity	N/A	Only at setup

<u>Table design for Activity</u>

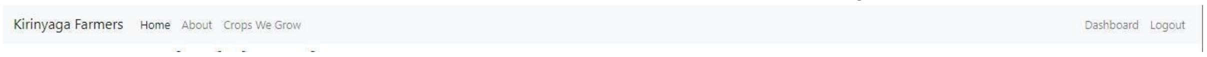
<u>Table name is activity</u>

key	**Name**	**Description**	**Type**	**Input**
Primary key	Id	The id assigned to each activity in week	Integer	Only at setup
	Activity	This is the activities inside the week	String	Only at setup
Foreign key	Week_id	This is the foreign key that links the table activity with week through the id of the week(week.id)		

5.2 User interface design

The online application contains four screens in total. When a user signs in, a dashboard bar will appear on the screens, allowing them to move between the website's many sections. There is also a logout button and a dashboard, and once a user signs in, crops we grow on all of the displays.

The dashboard is the screen where a user can click a button to schedule a job.



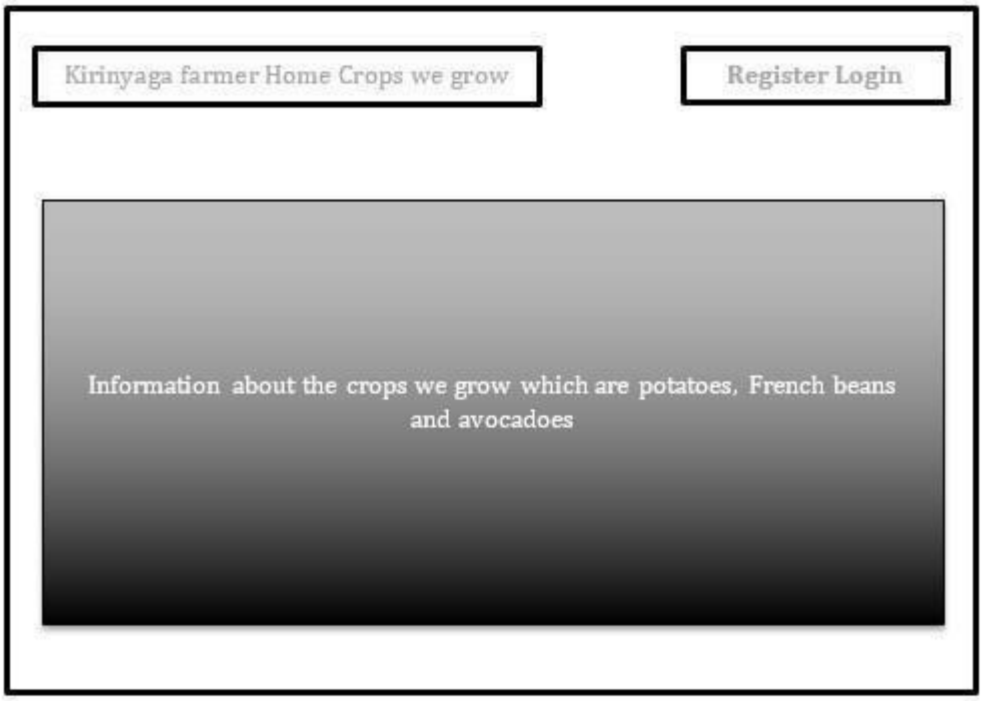
By selecting either Kirinyaga Farmers or dashboard, the farmer can return to the Main Page.

5.2.1 Main Screen

The main screen either with or without logging in allows the farmer to read information about the crops available by selecting the crops we grow button on the navigation bar. The main screen comprises different parts depending on whether the user is logged in or not.

I. User is not logged in

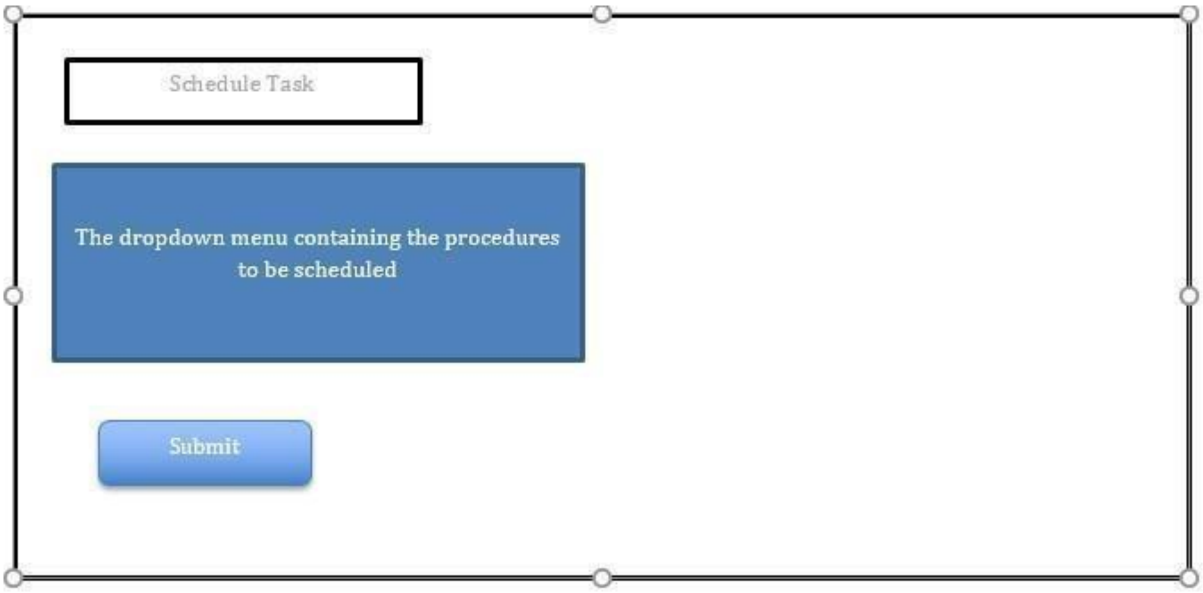
The viewer can examine information on the crops we crow option in addition to the website's main information.


Kirinyaga farmer Home Crops we grow
Register Login
Information about the crops we grow which are potatoes, French beans and avocadoes

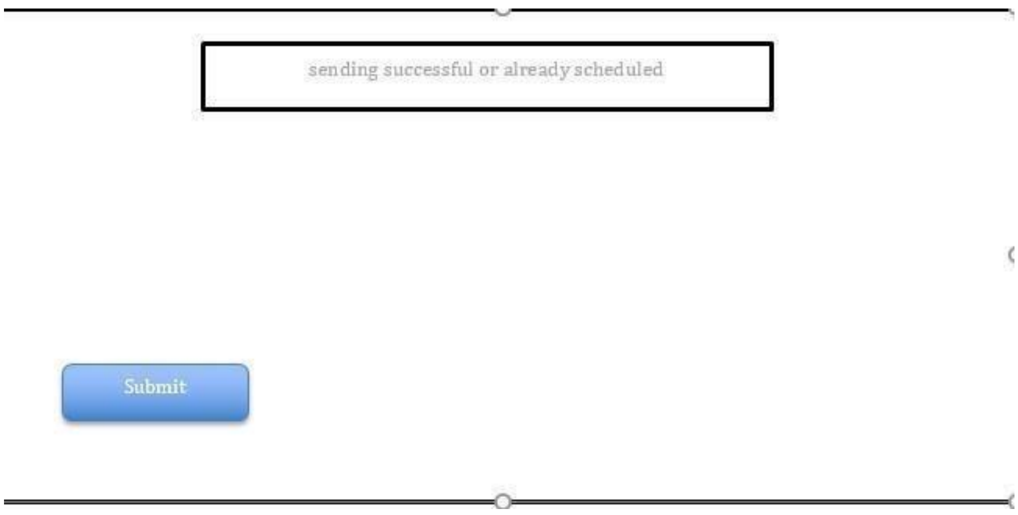
II. The user logs in by registering, then, a welcome message is displayed with the name of the farmer

III. A farmer who is logged in but has not planned any procedures clicks the button to do so, and a new page for doing so opens.

IV. A farmer who has logged in but has not yet scheduled a process can view information on the crops to schedule by using a drop-down menu form.

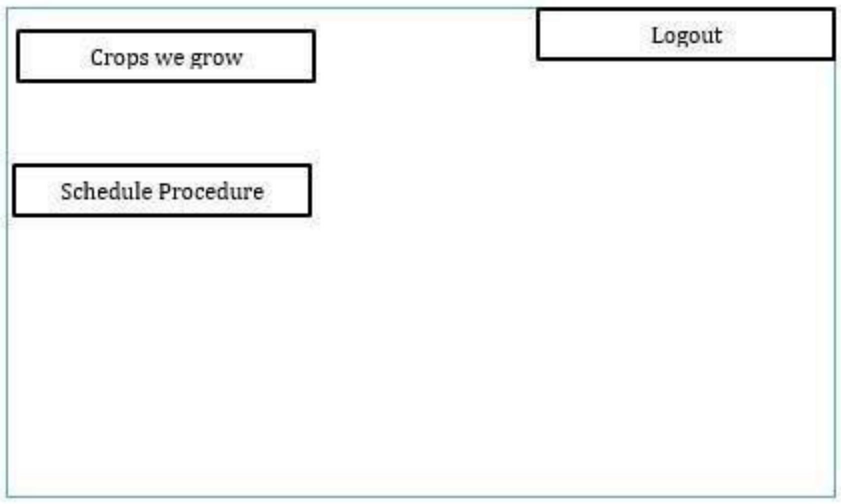


V. After logging in and choosing the crop to schedule, the farmer completes the process. They then wait for their phone to ring with the notification.



5.2.2 Dashboard screen

The dashboard screen is the screen where a user is directed after logging in and contains the link where you can navigate to schedule task



5.2.3 Log in Screens
The farmer must enter their email address and password in the form fields on the screen in order to log in. There is also a button to access their website account.

5.2.4 Register screen
The user must complete all the essential fields when registering on the screen (first name, second name, identification number, phone number, email address, county, village, land size, password) Once all the fields have been filled out, there is a button to click in order to register.

5.2.5 The functioning of buttons and links

Button, Link	**Pages**	**On Click**	**Visibility**	**Navigate To**
Log in Link	Home /Navigation bar	the user is signed into the account.	On all pages when the user is not logged in	Home
Logout link	On all pages only when farmer is logged in	The user is sighed out and returned to login pager		Home
Register Button	Home/navigation bar	Registers Farmer to the database	on navigation bar and on home page when the user is not logged in	Dashboard
Schedule procedures Button	Dashboard	Enable farmer to view procedures	On Schedule Task screen	Dashboard

Dashboard link	All screens when logged in	The farmer is redirected to schedule procedure page	On all screen when user is logged in	Dashboard
Schedule procedure button	Schedule procedures screen	The farmer sends SMS and schedule task	On schedule procedures screen	Schedule procedures

6 Implementation

6.1 Introduction

A plan, idea, model, design, specification, standard, algorithm, or policy can be put into action more easily with the help of an implementation plan.

6.2 Programming language, framework and file structure

6.2.1 Framework

I required a framework with a collection of reliable components in order to accelerate the web development process. code for functionality implementation, especially for user authentication The most popular web frameworks available in Python are Django and Python Flask (Ghimire, 2020). Both theories perform similarly in terms of effectiveness. Since flask is largely what we taught in this program and is a lightweight, minimalist framework that offers a number of features, I decided to use it. Here are a few of the factors that led me to choose a flask: Due to Flask's adaptability, developers can incorporate a variety of other apps within the framework. several different third-party programs by incorporating add-ons, libraries, third-party plugins, and modules, it also enables programmers to manage complex developing challenges.

Contrary to Flask, Django offers comprehensive documentation with a range of tutorials, allowing me to familiarize myself with the framework while creating my own web application (Idris et al., 2020). When compared to Flask, which permits developers to freely incorporate code from other libraries, Django offers a far larger selection of built-in packages, allowing developers to select the package that will most effectively support the creation and use of an application. As opposed to Django, which has its own built-in login and forms capabilities, a Flask developer might utilize Flask login for login and Flask WTF for forms (Idris et al., 2020). Both of these are offered by independent libraries. Due to these increased dependencies, learning new tools and reading new documentation becomes a challenge.

Developers can swiftly handle the application's core using Flask, while Django's high-level Python Framework has a steeper learning curve (Lokhande et al., 2015). Programmers may add new features and make the application more sophisticated with Flask's expandable framework, giving it an advantage over Django, which is not the best choice for projects that change frequently (Lokhande et al., 2015). Django and Flask guard against cross-site scripting (XSS) attacks, which transfer harmful code to a third party, user cross-site request forgery (CSRF), which allows a malicious user to log in using another user's credentials, and SQL injection, which allows a user to run SQL code on a database (Lokhande et al., 2015).

Some of the packages used to achieve scheduling and SMS service are

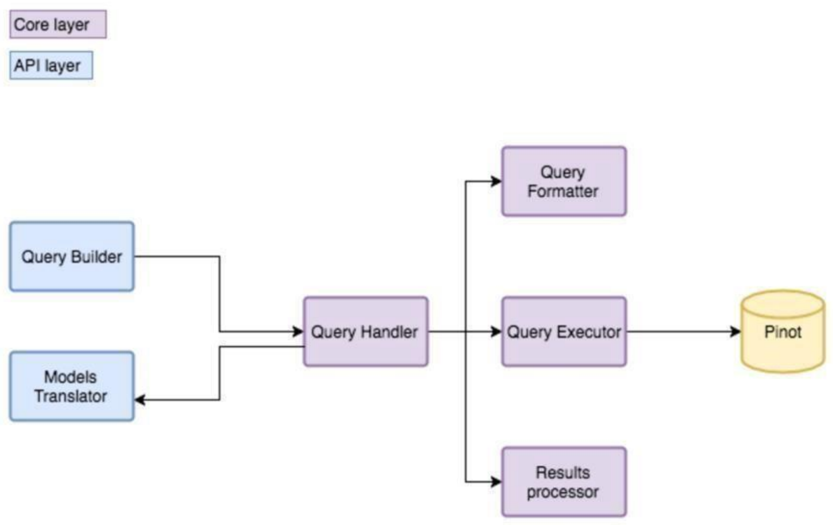
- **APSscheduler Package**

  With the help of APSscheduler Library, a task can be scheduled to run at a specific time on a specific day of the week or every day of the week (Zmora et al., 2019). In this scenario, the APSscheduler library will be used to set up weekly SMS notifications for farmers regarding the planning stages they are expected to carry out on the different types of crops they have chosen.

- **Sending Task**

  This system will make use of Twilio’ s two-way SMS and MMS messaging capabilities, which allow for the continuation of a conversation through the simultaneous sending and receiving of text and multimedia messages. This will make it possible for users of the system to have conversations by text message (SMS) with other users of the system, such as the farmers. Additionally, this will make it possible for users of the system to reschedule appointments via text message and receive automatic responses (Choi, 2021).

Figure 5: Flask Architecture



Source: Flask Website (2022)

6.2.3 File structure

This flask project has a single application that connects to a single database that has numerous tables. The farmer must first register their name and provide any pertinent information that may be required of them to submit.

6.3 Website Content

A top-down technique is used to construct an overview of the system, which identifies any first-level subsystems but does not elaborate on them. Then, each subsystem is enhanced in much greater detail, possibly at several more subsystem levels, until the entire specification is reduced to fundamental parts. "Black boxes" are widely utilized to create a top-down model to enable manipulation (Ashley, 2020). Black boxes, however, might not accurately reflect fundamental mechanisms or be accurate enough to evaluate the model in a realistic manner. Responsive web design is no longer just a UX issue due to the tremendous rise in mobile device usage. The website was designed with the intention that it will be responsive to all device sizes and be usable on small, medium, and large ones.

It was necessary to turn the content into a dictionary list before it could be used with the Python programming language. The forms contained in the web app are handled by the frontend toolkit known as Bootstrap (Ashley, 2020). Bootstrap is a strong, adaptable, and feature-rich framework. Using a pre-built grid system and components, building and customizing aspects using Sass, and bringing projects to life with powerful JavaScript plugins are all feasible options.

6.3.1 External Content

The messages that are displayed to a farmer when s/he chooses an option of successful completion of a task on the website and an error to either repeat doing it again correctly or has already been done are a flash message from flask flash package, they are represented by a shade of green for success, and an error message for shade of red. Both of these flask flash messages are supported by the Jinja template tool.

6.4 Function implementation

The system is put in a way to enable the user to pick the choice of the crops they wish to plant from the database. The choice of the user is then implemented into their weekly task which will ensure that the user receives timely SMS notifications on the procedures of planting the crop selected.

the following functions are implemented

- Scheduled_message

  The function calls the send_message function, which loops through the activities of that week before sending a message to the user via Twilio. The function queries all the users stored in the database and checks the attribute is_scheduled inside the database. If the user registered has not yet scheduled any messages, it first retrieves the data from the dictionary where the data has been stored. The phone_number retrieves the user's phone number from the database, while the message parameter attaches to the activity.

- Start_schedule

  This function invokes the schedule message function and makes use of the apsScheduler package.

  Because the function is invoked each time the app is launched, the start schedule function is called at the application level.

- Get_week

  This function verifies which week the farmer began scheduling.

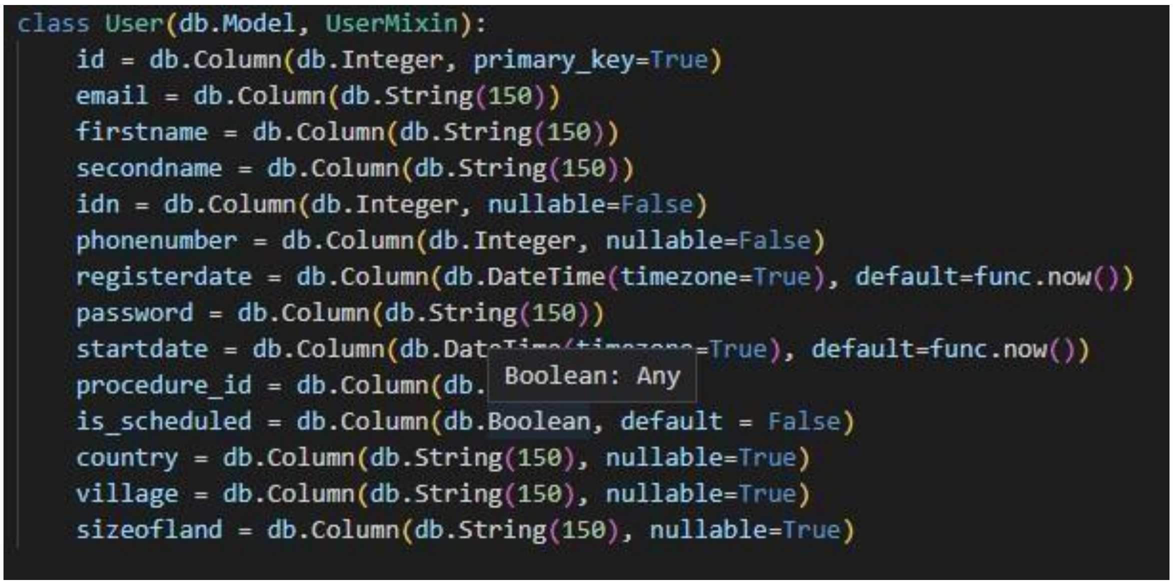
- Planting_procedures function

  This function keeps a list of information inside a dictionary. This is the information we'll use on our website.

6.5 Database Implementation

The table from database have been transformed into Flask models. We employ SQLite for local development, as detailed in the models.py file on the Flask website. Any commands supplied through our Flask models are instantly transformed to SQLite in order to preserve the SQLite database. The project's web application's models file contains a User database table that has been constructed. The following tables was created:

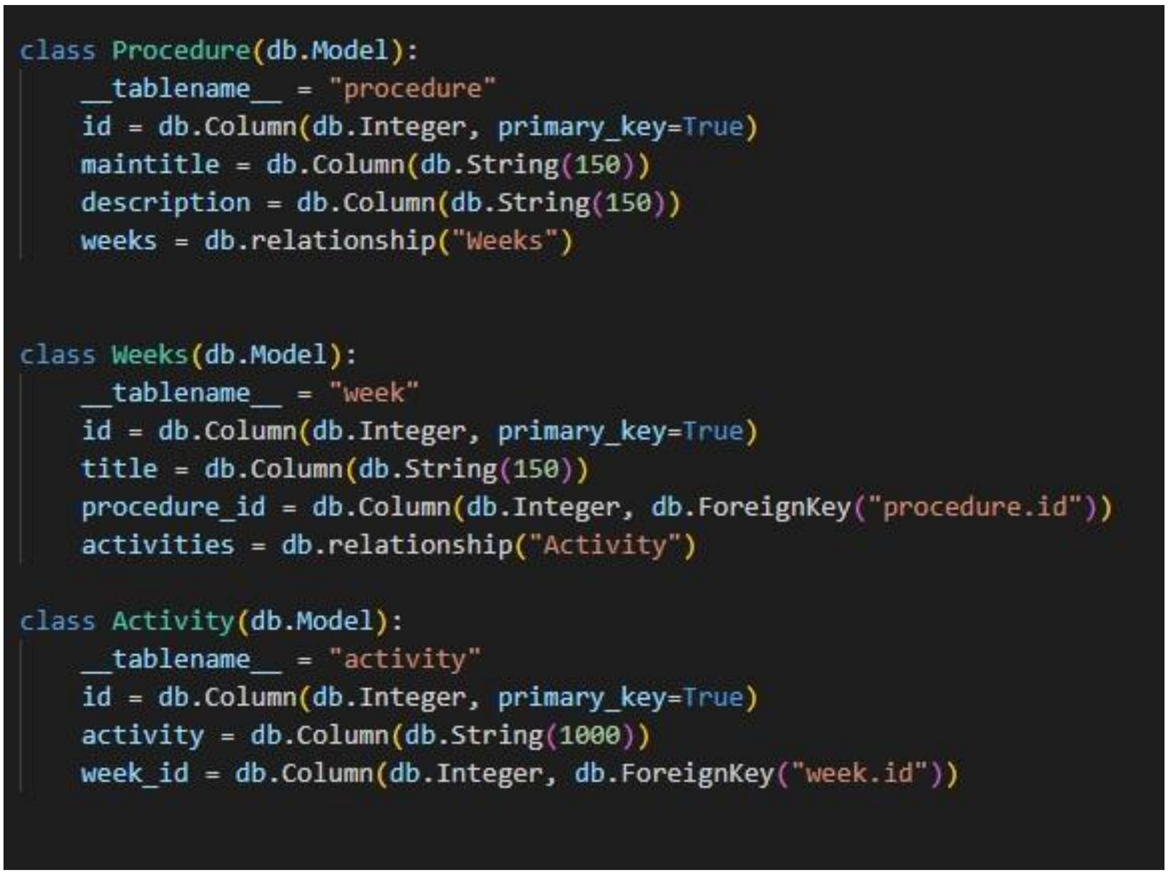
 Class User (db. Model)



```
class User(db.Model, UserMixin):
    id = db.Column(db.Integer, primary_key=True)
    email = db.Column(db.String(150))
    firstname = db.Column(db.String(150))
    secondname = db.Column(db.String(150))
    idn = db.Column(db.Integer, nullable=False)
    phonenumber = db.Column(db.Integer, nullable=False)
    registerdate = db.Column(db.DateTime(timezone=True), default=func.now())
    password = db.Column(db.String(150))
    startdate = db.Column(db.DateTime(timezone=True), default=func.now())
    procedure_id = db.Column(db. Boolean: Any
    is_scheduled = db.Column(db.Boolean, default = False)
    country = db.Column(db.String(150), nullable=True)
    village = db.Column(db.String(150), nullable=True)
    sizeofland = db.Column(db.String(150), nullable=True)
```

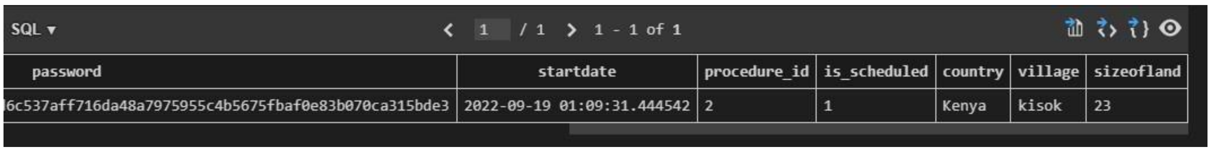
The table design with is_scheduled attribute so that a user can only schedule one task either

for planting one of the crop as shown below:



```
class Procedure(db.Model):
    __tablename__ = "procedure"
    id = db.Column(db.Integer, primary_key=True)
    maintitle = db.Column(db.String(150))
    description = db.Column(db.String(150))
    weeks = db.relationship("Weeks")


class Weeks(db.Model):
    __tablename__ = "week"
    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(150))
    procedure_id = db.Column(db.Integer, db.ForeignKey("procedure.id"))
    activities = db.relationship("Activity")

class Activity(db.Model):
    __tablename__ = "activity"
    id = db.Column(db.Integer, primary_key=True)
    activity = db.Column(db.String(1000))
    week_id = db.Column(db.Integer, db.ForeignKey("week.id"))
```

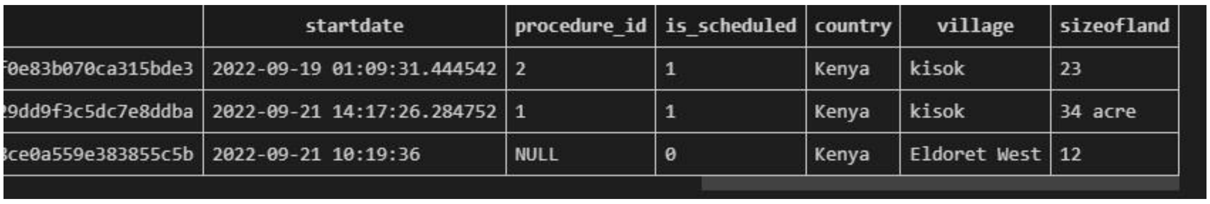
The user after scheduling a task:



SQL ▾ ‹ 1 / 1 › 1 - 1 of 1

password	startdate	procedure_id	is_scheduled	country	village	sizeofland
16c537aff716da48a7975955c4b5675fbaf0e83b070ca315bde3	2022-09-19 01:09:31.444542	2	1	Kenya	kisok	23

A registered user before scheduling a task:



	startdate	procedure_id	is_scheduled	country	village	sizeofland
f0e83b070ca315bde3	2022-09-19 01:09:31.444542	2	1	Kenya	kisok	23
29dd9f3c5dc7e8ddba	2022-09-21 14:17:26.284752	1	1	Kenya	kisok	34 acre
3ce0a559e383855c5b	2022-09-21 10:19:36	NULL	0	Kenya	Eldoret West	12

In the case of a user selecting potatoes, the crop_id selects the crop's ID of 2, according to the information provided in the database.

The farmer's crop_id serves as a representation of the sort of crop the farmer has chosen. Since the farmer has not yet chosen the type of crop s/he wants to select, crop_id is not assigned during registration. However, once the farmer chooses a type of crop to select from the form in figure 11, or rather a procedure, it takes the id from the database table and the farmer is_schedule attribute will indicate that the farmer has begun scheduling which is 1 procedure at single time.

To prevent the date from falling inside the previous week when the user launches Scheduling operations for the first time, the default value for the last starting date must be advanced from the time of the launch this is achieved from the function get_week with the parameter of start_date.

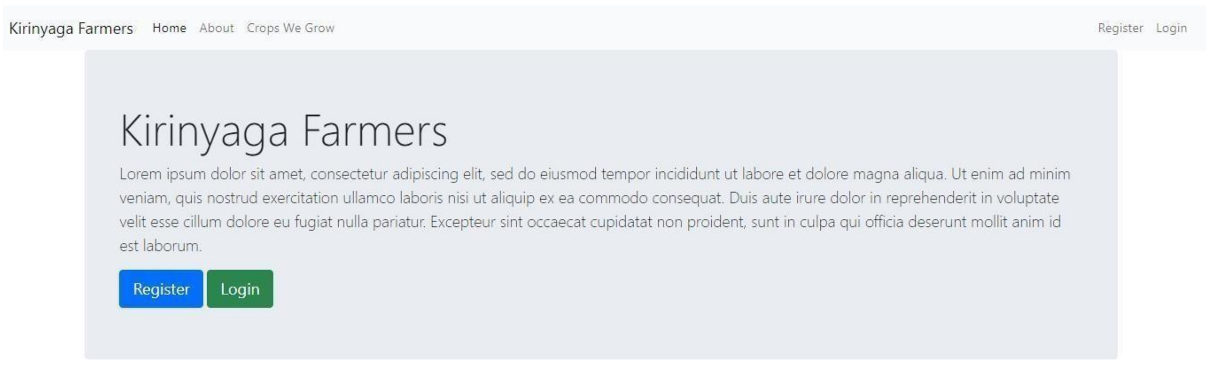
6.6 User Interface Implementation

The HTML templates for the screens of our web application may be found in the themes directory. These templates are used to generate the screens. There is a layout template in our repository that goes by the name layout.html, and it is the progenitor of all of our other HTML templates. We can use the layout template to incorporate Bootstrap into our project by adding a link to its CSS stylesheet in the head of layout.html. This will allow us to use the layout template. The building of websites can be sped up and made more efficient by using Bootstrap, which is a free front-end framework (Taneja and Gupta, 2014). The vast majority of the most popular browsers are supported, and Bootstrap's adaptable CSS automatically changes itself to fit a variety of devices. Every other template possesses a wrapper that is used to encapsulate the content of that template. For example, the home template's home.html file is enclosed within the class jumbotron-jumbotron-center so that the content can be centralised on that page (Taneja and Gupta, 2014).

Every template utilizes the predefined buttons that are provided by Bootstrap. Include is the name of a new folder that is generated inside the folder that contains the template, and it is where the form helpers, navigation, and error templates may be found once they have been added to the layout template. When the user is signed in, the menu bar of the website displays links to the pages of the website. When the user is not signed in, the menu bar displays navigation to the home page and other web sites that do not require login.

Home screen

Figure: user not Logged In

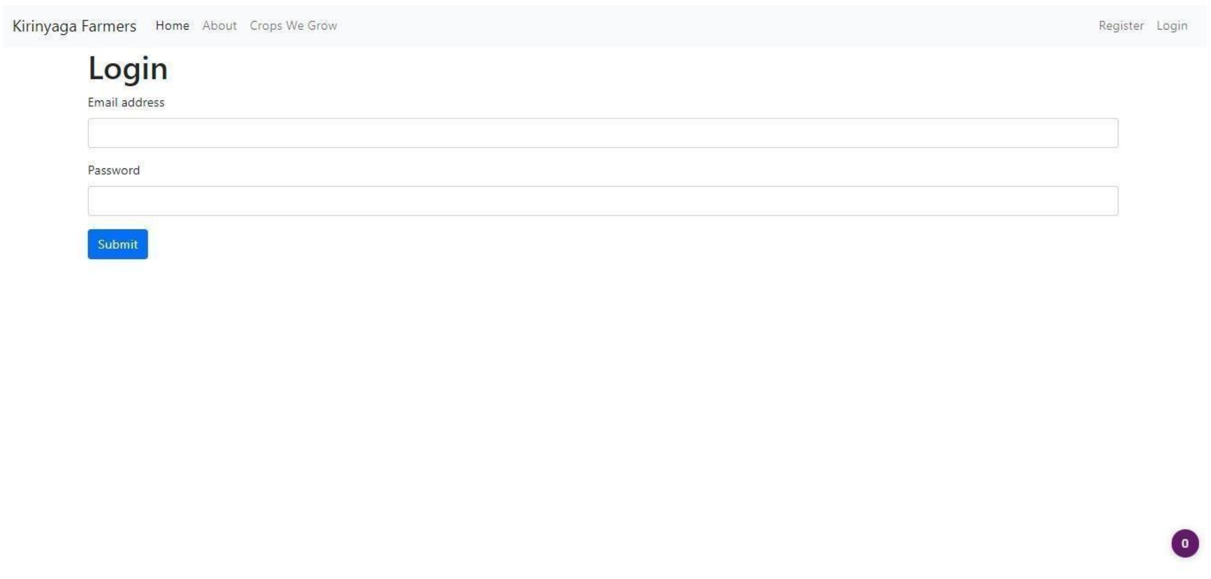


Kirinyaga Farmers

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Figure 6: **Login page**

This is the login page in which a farmer will log in after a successful registration



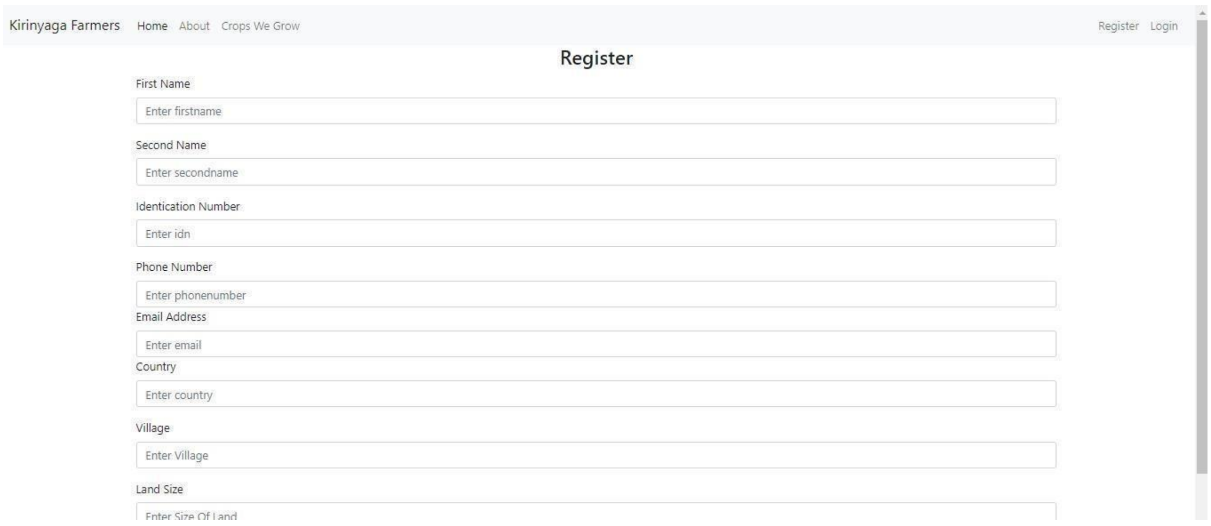
Login

Email address

Password

Figure 7: Farmer sign up Screen

This is the page in which a farmer registers before scheduling any of the crop or procedures



Register

First Name
Enter firstname

Second Name
Enter secondname

Identication Number
Enter idn

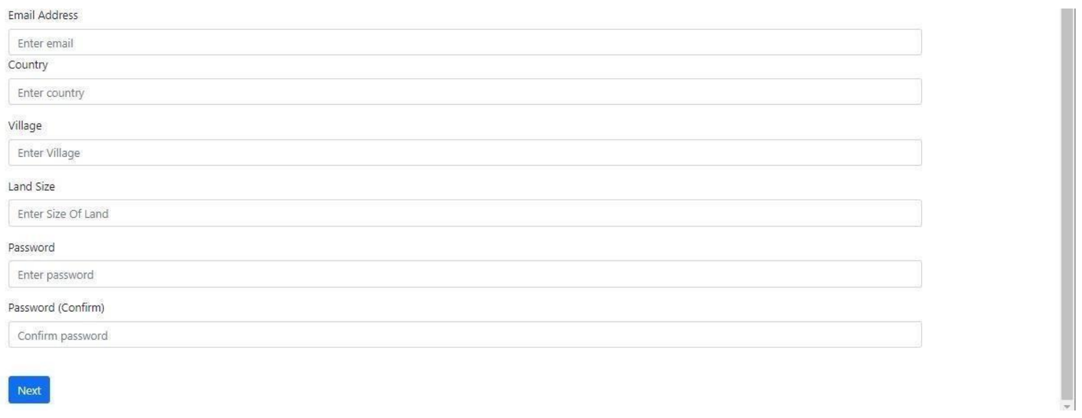
Phone Number
Enter phonenumber

Email Address
Enter email

Country
Enter country

Village
Enter Village

Land Size
Enter Size Of Land


Email Address

Enter email

Country

Enter country

Village

Enter Village

Land Size

Enter Size Of Land

Password

Enter password

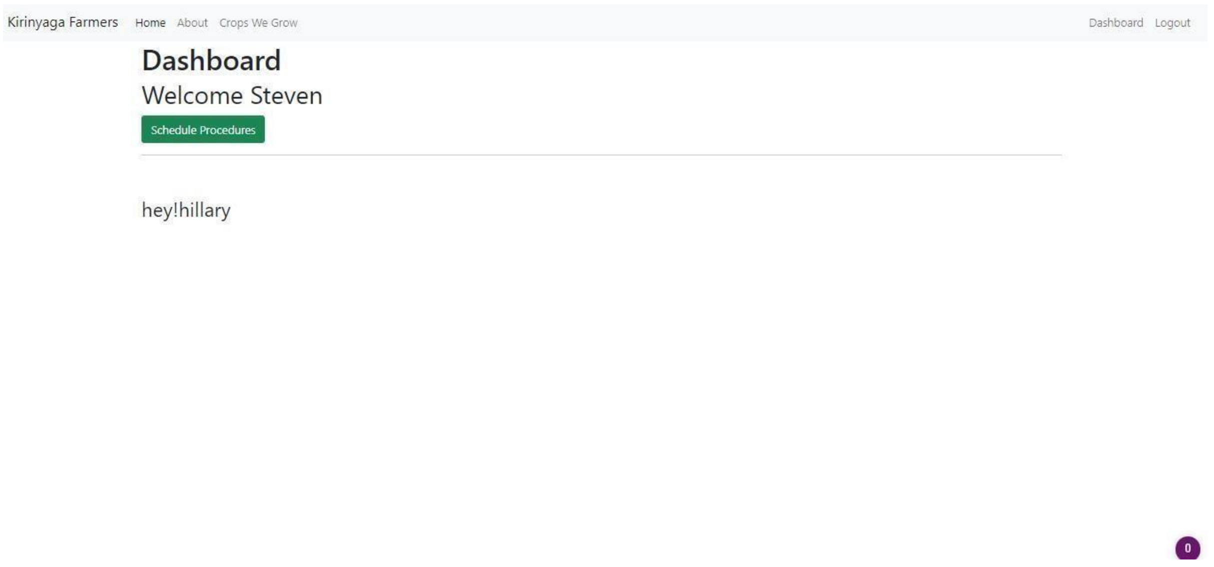
Password (Confirm)

Confirm password

Next

Figure 8: User logged In

This is the page in which the user is directed after a successful log in and a display of the first name and second name of the registered farmer


Kirinyaga Farmers Home About Crops We Grow Dashboard Logout

Dashboard

Welcome Steven

Schedule Procedures

hey!hillary

Figure 9: user Viewing Crop Scheduling

This is the page in which a farmer can view the type of crops available for scheduling and is available either before a farmer has logged in or not

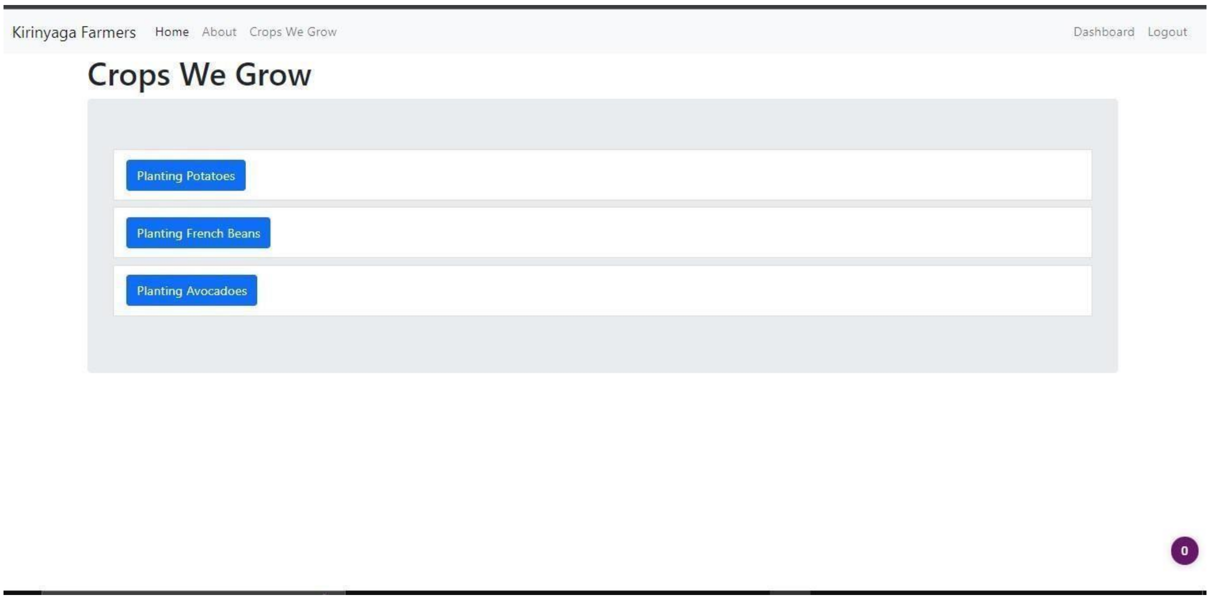
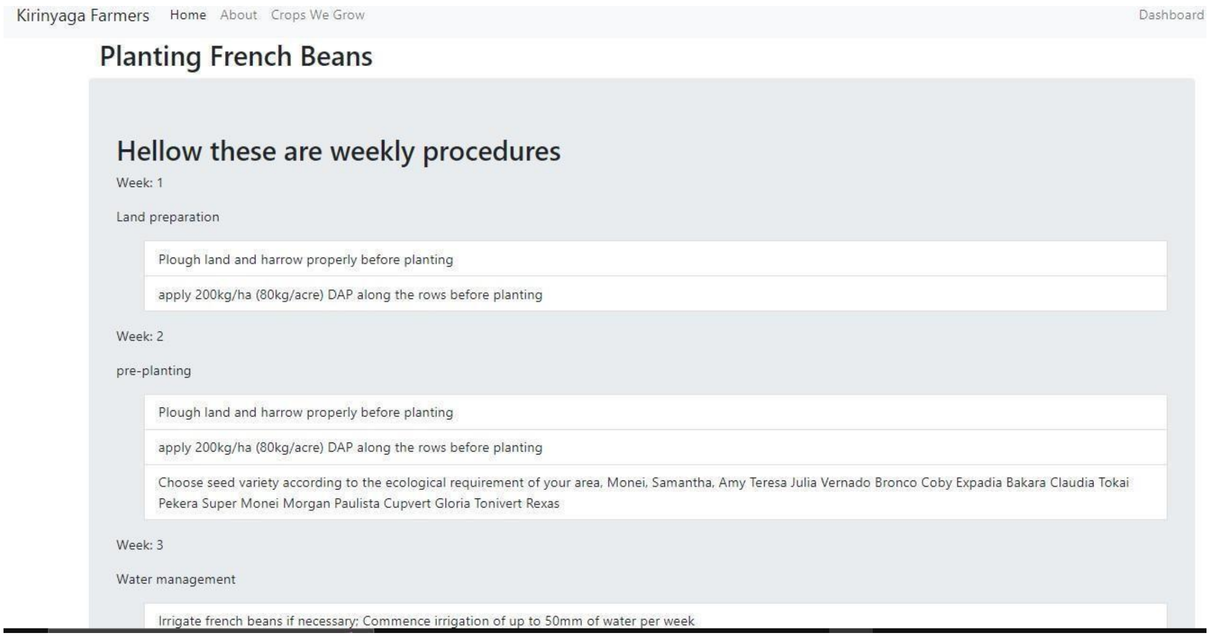


Figure 10: User viewing each step before scheduling

This is the information about the crops and details of weekly procedures in which a farmer will get weekly notification of each



Planting French Beans

Hellow these are weekly procedures

Week: 1

Land preparation

- Plough land and harrow properly before planting
- apply 200kg/ha (80kg/acre) DAP along the rows before planting

Week: 2

pre-planting

- Plough land and harrow properly before planting
- apply 200kg/ha (80kg/acre) DAP along the rows before planting
- Choose seed variety according to the ecological requirement of your area, Monei, Samantha, Amy Teresa Julia Vernado Bronco Coby Expadia Bakara Claudia Tokai Pekera Super Monei Morgan Paulista Cupvert Gloria Tonivert Rexas

Week: 3

Water management

- Irrigate french beans if necessary; Commence irrigation of up to 50mm of water per week

Figure 11: User Logged in to start weekly scheduling

This is the page in which the crops are listed through a drop down menu and a farmer is allowed to choose only one type of crop to schedule a procedure

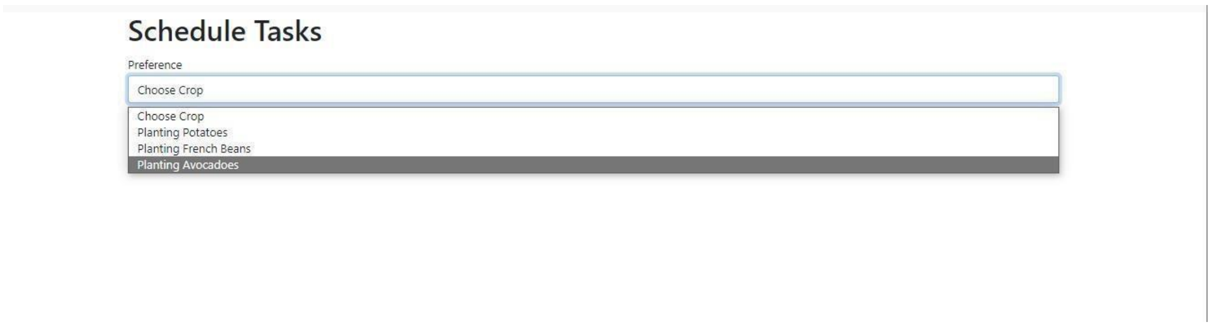


Figure 12: user selecting one type of crop

The user has selected one of the crop to schedule and a success message is flashed on the screen

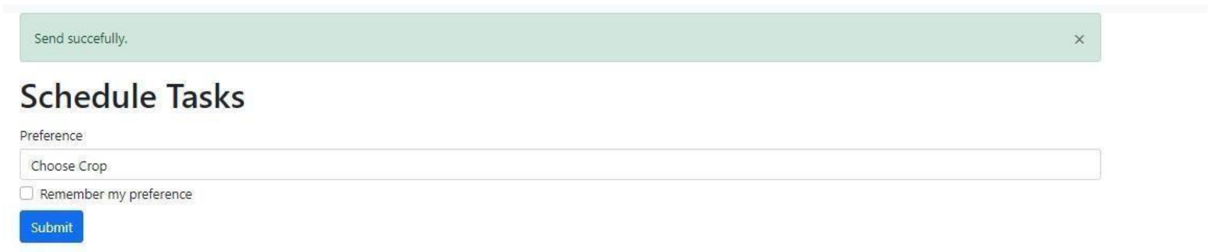

Send succefully.

Figure 13: user submitting for the second time

It shows that no type of crop to select because s/he has already scheduled because only one type of crop has to be selected at that period of time

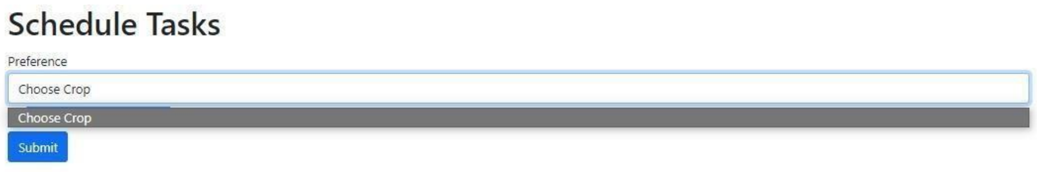
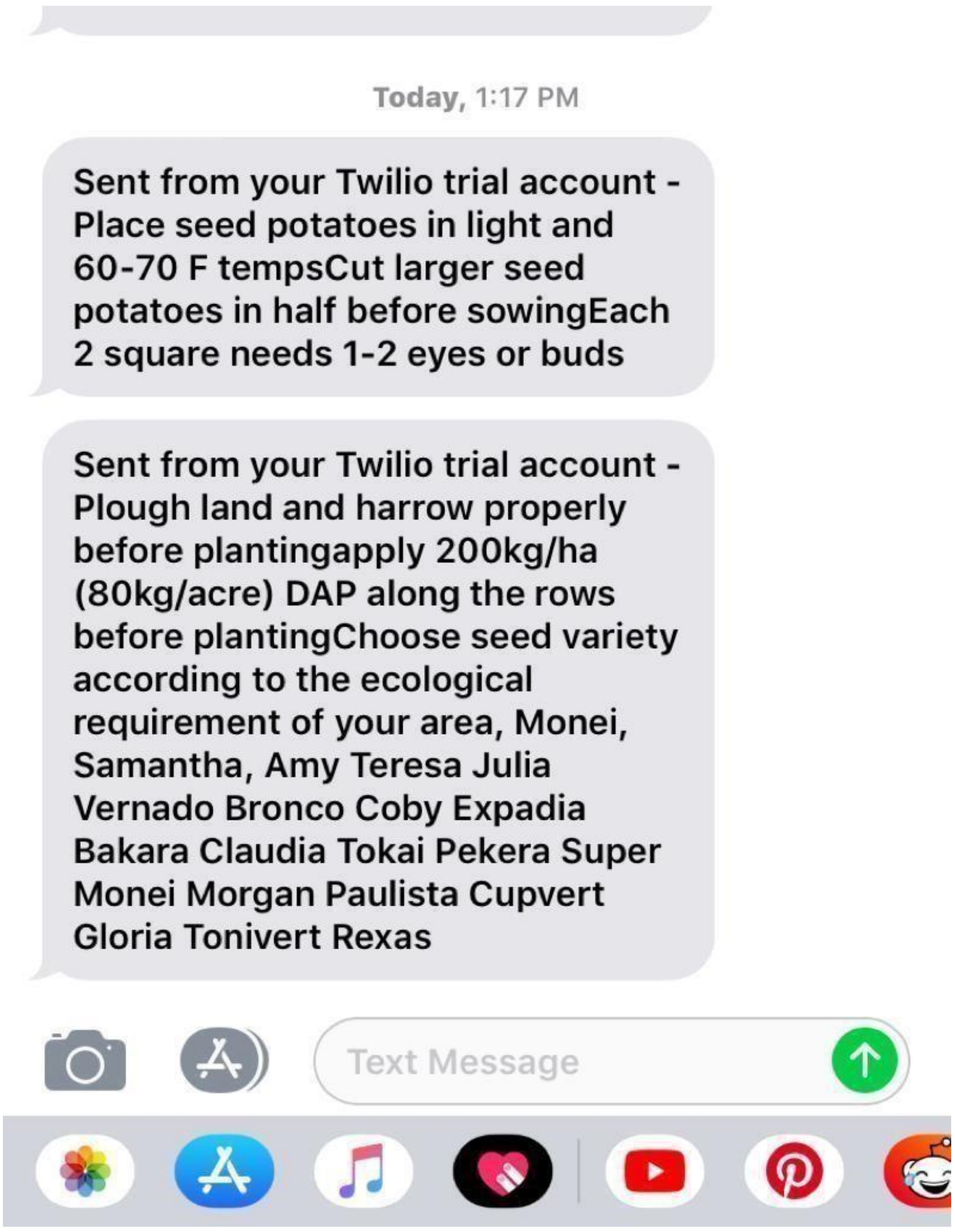


Figure 14: information received from the user phone number after five minutes as explained

in testing



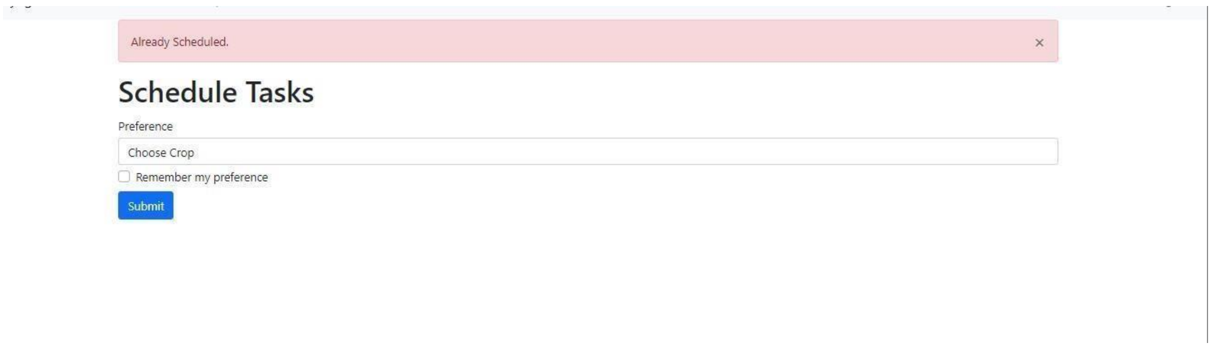
Today, 1:17 PM

Sent from your Twilio trial account - Place seed potatoes in light and 60-70 F tempsCut larger seed potatoes in half before sowingEach 2 square needs 1-2 eyes or buds

Sent from your Twilio trial account - Plough land and harrow properly before plantingapply 200kg/ha (80kg/acre) DAP along the rows before plantingChoose seed variety according to the ecological requirement of your area, Monei, Samantha, Amy Teresa Julia Vernado Bronco Coby Expadia Bakara Claudia Tokai Pekera Super Monei Morgan Paulista Cupvert Gloria Tonivert Rexas

Figure 15: user logging out and trying to schedule again

The user will try to log out and then start again scheduling then s/he will receive a notification that s/he had already scheduled a procedure


Already Scheduled.

Schedule Tasks

Preference

Choose Crop

Remember my preference

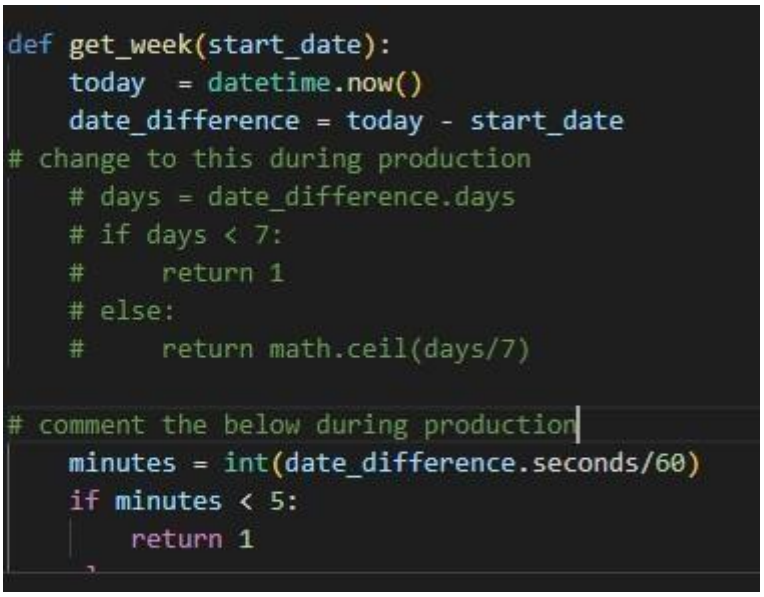
Submit

7 Testing

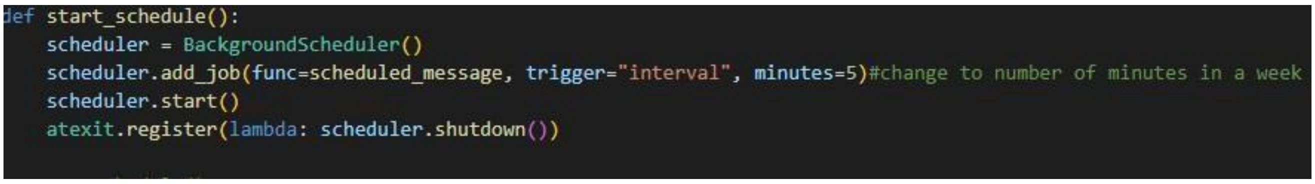
7.1 Introduction

In the test plan, both the testing technique and the overall framework that will be utilized to direct the testing of the Automated SMS System are provided. The following items will be discussed in this document:

- The test strategy contains not only the principles on which the tests will be based, but also their goals, presumptions, and descriptions of the methods that are necessary to carry out legitimate testing.
- The procedures that need to be taken in order to discover, report, and correct problems are outlined in the execution strategy for the test.
- Test management is a process that handles the logic of tests as well as any situations that are relevant to their execution
- Unit Testing for a variety of reasons, testing websites or web services is challenging. You have Code-within-Code, which includes SQL and HTML template engines. Additionally, you have dependencies like databases, which are challenging to imitate. Unit tests and manual tests were both employed in the development stage. Pytest module was actively used in unit testing.
- Twilio was my choice for messaging and due to limited resources the application module only allows a subscribed phone number to verify and send messages in this account I choose to use my phone number as a testing subject. Twilio allows sending messages to many unsubscribed phones numbers through a paid account.
- In terms of testing for receiving notifications the code was changed to five minutes instead of one week. This provided a practical testing because waiting for a week to receive a notification was not possible that’s why an approach of taking five minutes to represent one week for timely notification was more practical. During production the code can be adjusted as shown below:

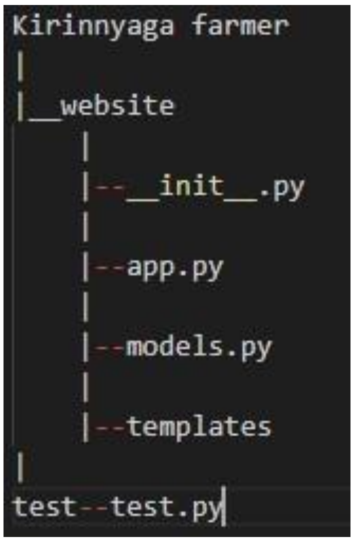


```
def get_week(start_date):
    today  = datetime.now()
    date_difference = today - start_date
# change to this during production
    # days = date_difference.days
    # if days < 7:
    #     return 1
    # else:
    #     return math.ceil(days/7)

# comment the below during production
    minutes = int(date_difference.seconds/60)
    if minutes < 5:
        return 1
```



```
def start_schedule():
    scheduler = BackgroundScheduler()
    scheduler.add_job(func=scheduled_message, trigger="interval", minutes=5)#change to number of minutes in a week
    scheduler.start()
    atexit.register(lambda: scheduler.shutdown())
```

- Weekly testing provided little practical benefit in terms of testing technique, so a five minute’s test time limit was taken into consideration.
- The hierarchical structure in which testing followed is as follows:



```
Kirinnyaga farmer
|
|__website
     |
     |--__init__.py
     |
     |--app.py
     |
     |--models.py
     |
     |--templates
|
test--test.py
```

7.2 Unit testing

The testing tools provided by Flask were used to create a number of unit tests. Pytest as explained above was used to create standardize procedures for testing farmers.

The ability to perform unit tests with each modification to the code is one of their key benefits. The output of this project is heavily date-dependent. As in the week attribute, the user may schedule a task for the week and have it automatically move to the following week. As expected test values fluctuate over time, this meant that unit tests were frequently not the best option.

8 Evaluation

I generated a list of questions concerning the software's user interface, content, and structure. In order to get the most useful input possible during usability testing, I followed recommended practices and asked participants open-ended queries about their perspective using the site. I also had the opportunity to watch one participant (the farmer) use the site in its entirety so that I could better understand the user experience.

Online website evaluation was done by Five people whom included the farmer located in central Kenya and her one of the employer.

Information sheet is available in appendix A

8.1 Registration, Usability, Design

The participants all agreed that the registration process was straightforward, and they had no trouble navigating the website. Testers gave positive feedback that the website's layout and content were easy to understand and not overly complicated. One of the people who took part in the discussion remarked that the menu bar may have benefited from being subdivided into more categories. Two of the respondents mentioned that it was simple to navigate to the crop selection panel, and they also mentioned that it was simple to sign up to receive SMS notifications.

8.2 Content

Three out of the Five testers indicated that they received timely and informative procedures on how to plant the crops they had selected.

Two testers also indicated that their SMS queries were responded to in a timely and customized way to address their crop planting needs.

9 Critical Analysis

9.1 Aims and Objectives

This application was successful in accomplishing its main objective, which was to assure the timely facilitation of crop planting operations to farmers on a weekly basis and, as a result, to assist them in producing crops of a higher quality. Farmers will be entered into the system after being registered by the farming society. Farmers who have registered their accounts will be able to receive automated text message alerts about a variety of topics, including reminders to plant, suggestions for which seeds to use, information about when to fertilize and information about when to harvest.

Users were able to successfully register their particulars, select crops, provide their phone numbers, and receive SMS notification on the planting procedures according to each crop hey had selected.

9.2 Limitations and Improvement

The user interface could appear more polished in terms of the style and layout; nonetheless, the objective was to keep the interface as simple and straightforward as possible for users to navigate. A few of the individuals who provided feedback on the application noted that moving between the app's many tabs was more time consuming. As an alternative to listing all of the possible planting options on a single page, for example, a "drop-down" tab might be a more effective way to guide people through the process of selecting appropriate crops for their gardens.

10 Conclusion

The project's goals have been realized in the final version of the app, which allows users to sign up with ease, choose the crops they want to plant, and receive timely SMS alerts with helpful planting instructions.

The evaluation phase was the most exciting part of the project because I got so much positive feedback from different farmer users. I got to know their opinion of the site and benefited from their insightful feedback and suggestions. More people from rural parts of East Africa would have been great. Unfortunately, the timeline prevented that from happening.

Reference List

Ashley, D., 2020. Using flask and jinja. In *Foundation Dynamic Web Pages with Python* (pp.

159-181). Apress, Berkeley, CA.

Choi, B., 2021. Python Network Automation Labs: Ansible, pyATS, Docker, and the Twilio API. In *Introduction to Python Network Automation* (pp. 675-732). Apress, Berkeley, CA.

Choi, B., 2021. Python Network Automation Labs: cron and SNMPv3. In *Introduction to Python Network Automation* (pp. 629-673). Apress, Berkeley, CA.

Deichmann, U., Goyal, A. and Mishra, D., 2016. Will digital technologies transform agriculture in developing countries? *Agricultural Economics, 47*(S1), pp.21-33.

Desikan, S. and Ramesh, G., 2006. *Software testing: principles and practice*. Pearson Education India.

DigiCow Website. 2022. *Our Products – Farmingtech*. [online] Available at: https://digicow.co.ke/our-products/ [Accessed 12 Sep. 2022].

Ghimire, D., 2020. Comparative study on Python web frameworks: Flask and Django.

Idris, N., Foozy, C.F.M. and Shamala, P., 2020. A generic review of web technology: Django and flask. *International Journal of Advanced Science Computing and Engineering*, *2*(1), pp.34-40.

Lokhande, P.S., Aslam, F., Hawa, N., Munir, J. and Gulamgaus, M., 2015. Efficient way of web development using python and flask.

Lutz, M., 2001. *Programming python*. " O'Reilly Media, Inc.".

Singh, S.K. and Singh, A., 2012. *Software testing*. Vandana Publications.

Stone, D., Jarrett, C., Woodroffe, M. and Minocha, S., 2005. *User interface design and evaluation*. Elsevier.

Taneja, S. and Gupta, P.R., 2014. Python as a tool for web server application development. *Int. J. Information, Commun. Comput. Technol*, *2*(1), pp.2347-7202.

Tripathy, P. and Naik, K., 2011. *Software testing and quality assurance: theory and practice*. John Wiley & Sons.

Twiga Foods. 2022. *Kenyan startup develops mobile-based service to save farmers the difficulty in dealing with brokers*. [online] Available at: https://farmbizafrica.com/market/2267-kenyan-startup-developsmobilebasedserviceto-save-farmers-the-difficulty-in-dealing-with-brokers [Accessed 12 Sep. 2022].

Zmora, N., Jacob, G., Zlotnik, L., Elharar, B. and Novik, G., 2019. Neural network distiller: A python package for dnn compression research. *arXiv preprint arXiv:1910.12232*.

11 Appendices

11.1Appendix A: evaluation forms

<u>A web based application for small scale farmers based on east Africa</u>

<u>Details regarding the application</u>

The website's automated messaging service provides farmers in the east African region with weekly instructions on how to enhance plant growth and produce more.

The application has many benefits which include: weekly update on what you will need to do to your crop, actual procedures without auto-messaging and many others benefits.

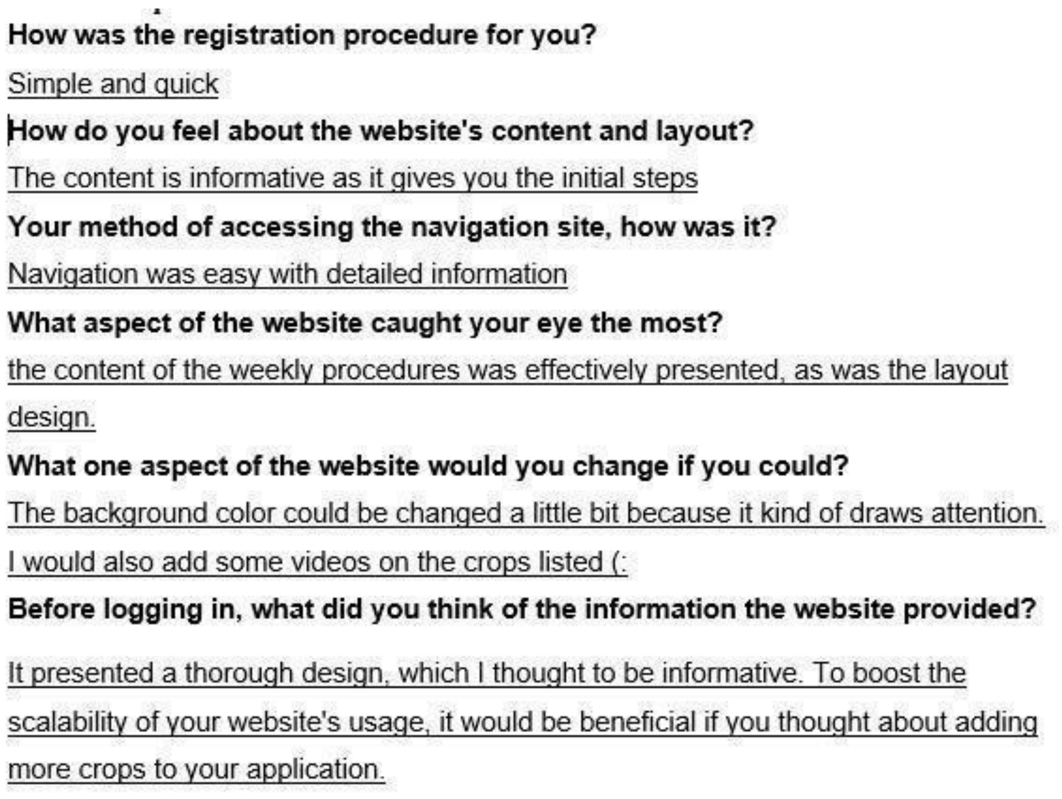
The application data is based data or rather on how you will get automated messaging is dependent on the date in which you will provide during farmer registration as shown **in Figure 7,** since it was an online evaluation the farmer was suggested to create an account and provide his or her phone number correctly.

<u>Evaluation Form</u>

- How was the registration procedure for you?
- How do you feel about the website's content and layout?
- Your method of accessing the navigation site, how was it?
- What aspect of the website caught your eye the most?
- What one aspect of the website would you change if you could?
- Before logging in, what did you think of the information the website provided?

11.2 Appendix B: evaluation forms response

1. **Response one 1**



How was the registration procedure for you?

Simple and quick

How do you feel about the website's content and layout?

The content is informative as it gives you the initial steps

Your method of accessing the navigation site, how was it?

Navigation was easy with detailed information

What aspect of the website caught your eye the most?

the content of the weekly procedures was effectively presented, as was the layout design.

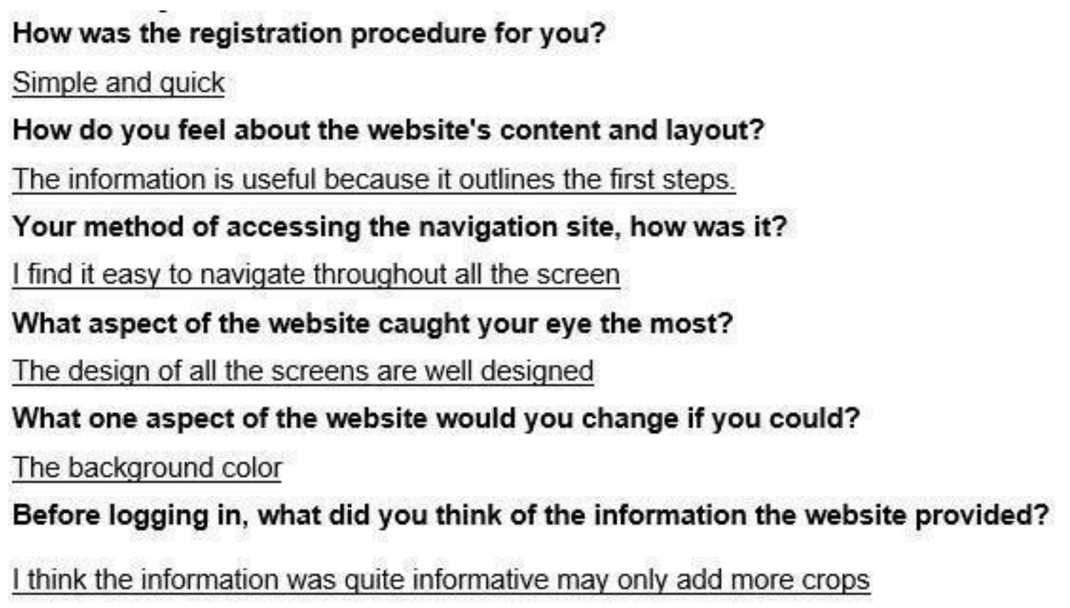
What one aspect of the website would you change if you could?

The background color could be changed a little bit because it kind of draws attention. I would also add some videos on the crops listed (:

Before logging in, what did you think of the information the website provided?

It presented a thorough design, which I thought to be informative. To boost the scalability of your website's usage, it would be beneficial if you thought about adding more crops to your application.

2. Response 2



How was the registration procedure for you?

Simple and quick

How do you feel about the website's content and layout?

The information is useful because it outlines the first steps.

Your method of accessing the navigation site, how was it?

I find it easy to navigate throughout all the screen

What aspect of the website caught your eye the most?

The design of all the screens are well designed

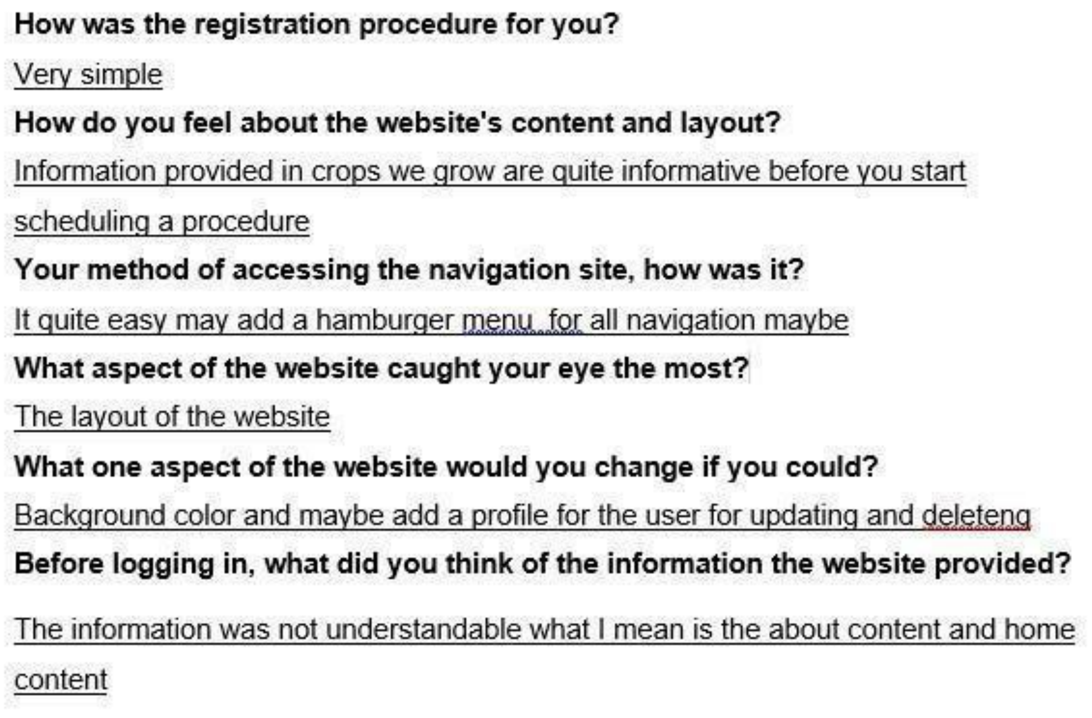
What one aspect of the website would you change if you could?

The background color

Before logging in, what did you think of the information the website provided?

I think the information was quite informative may only add more crops

3. Response 3



How was the registration procedure for you?

Very simple

How do you feel about the website's content and layout?

Information provided in crops we grow are quite informative before you start scheduling a procedure

Your method of accessing the navigation site, how was it?

It quite easy may add a hamburger menu for all navigation maybe

What aspect of the website caught your eye the most?

The layout of the website

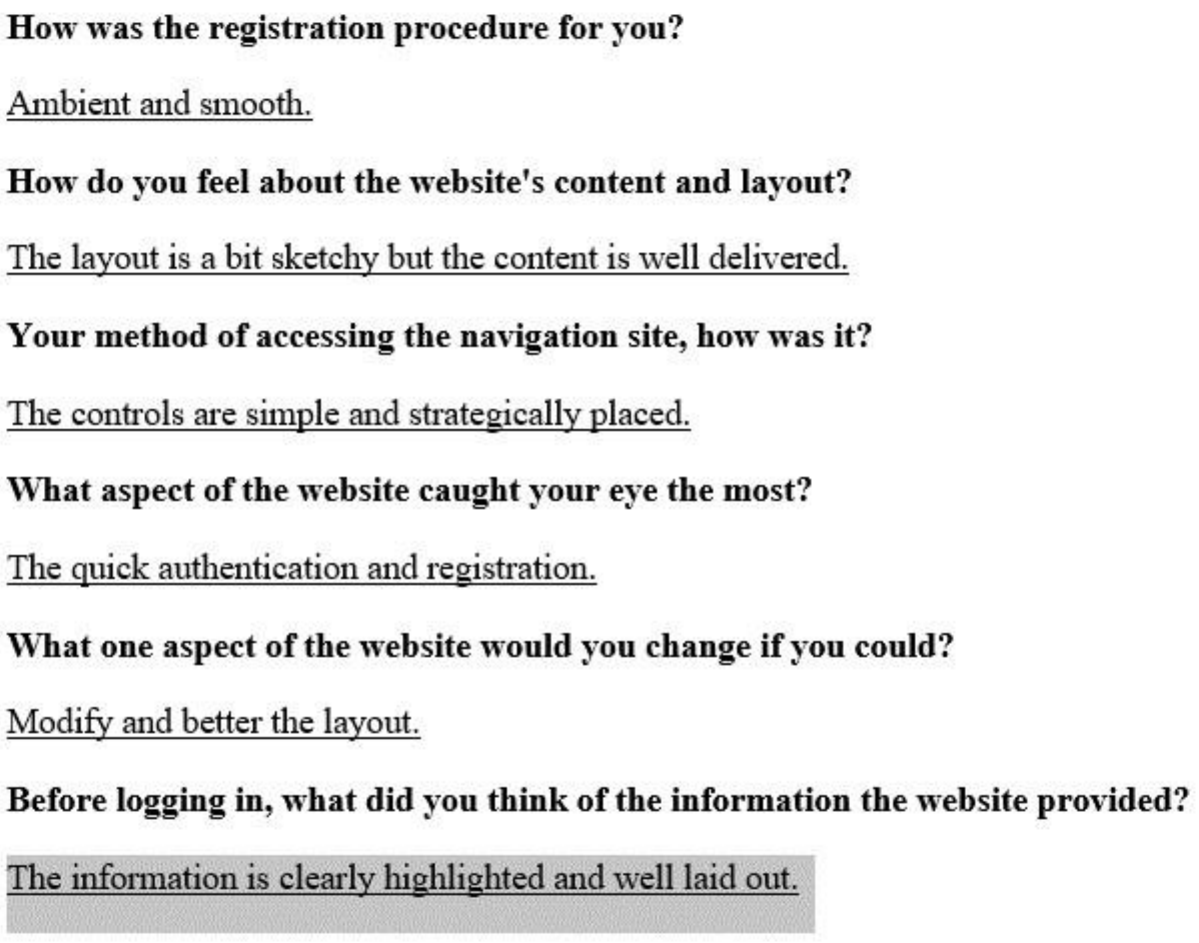
What one aspect of the website would you change if you could?

Background color and maybe add a profile for the user for updating and deleteng

Before logging in, what did you think of the information the website provided?

The information was not understandable what I mean is the about content and home content

4 Response 4



How was the registration procedure for you?

Ambient and smooth.

How do you feel about the website's content and layout?

The layout is a bit sketchy but the content is well delivered.

Your method of accessing the navigation site, how was it?

The controls are simple and strategically placed.

What aspect of the website caught your eye the most?

The quick authentication and registration.

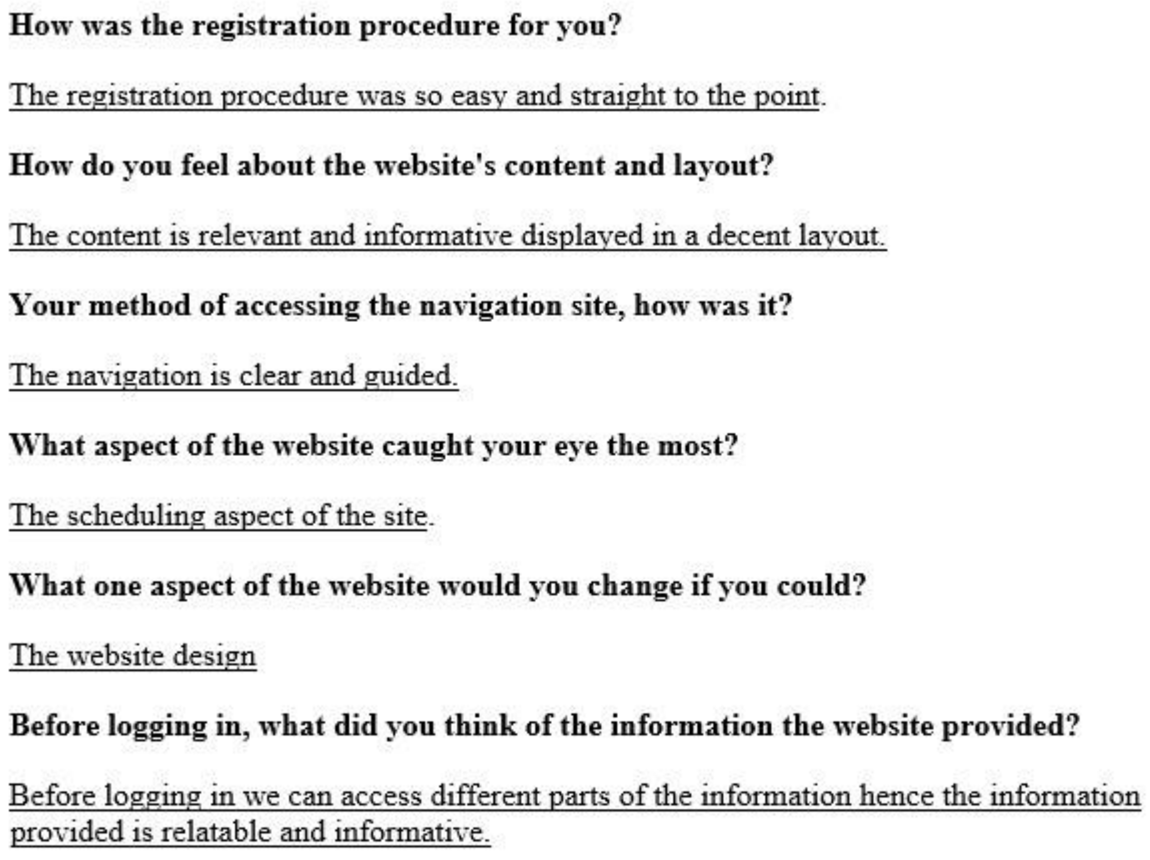
What one aspect of the website would you change if you could?

Modify and better the layout.

Before logging in, what did you think of the information the website provided?

The information is clearly highlighted and well laid out.

5 Response 5



How was the registration procedure for you?

The registration procedure was so easy and straight to the point.

How do you feel about the website's content and layout?

The content is relevant and informative displayed in a decent layout.

Your method of accessing the navigation site, how was it?

The navigation is clear and guided.

What aspect of the website caught your eye the most?

The scheduling aspect of the site.

What one aspect of the website would you change if you could?

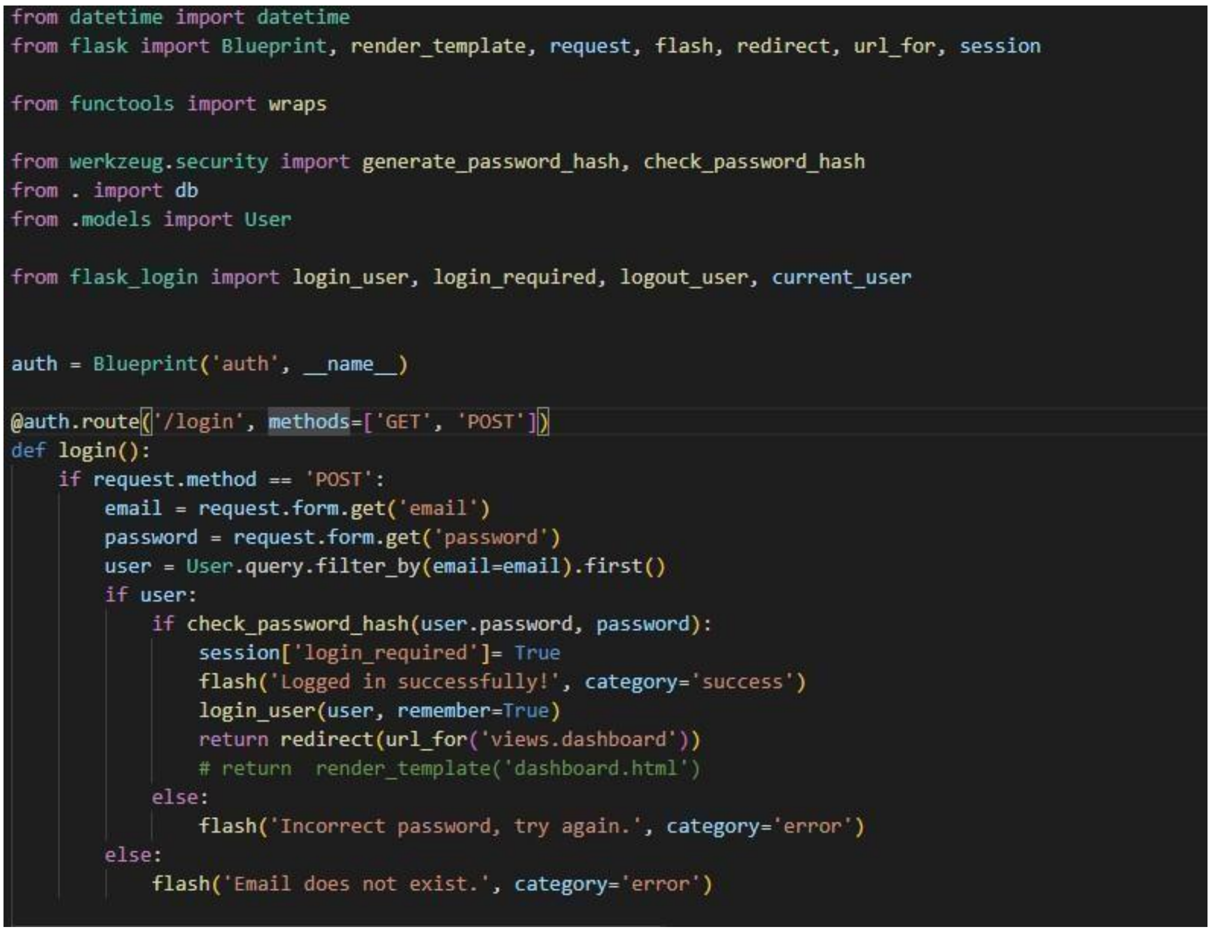
The website design

Before logging in, what did you think of the information the website provided?

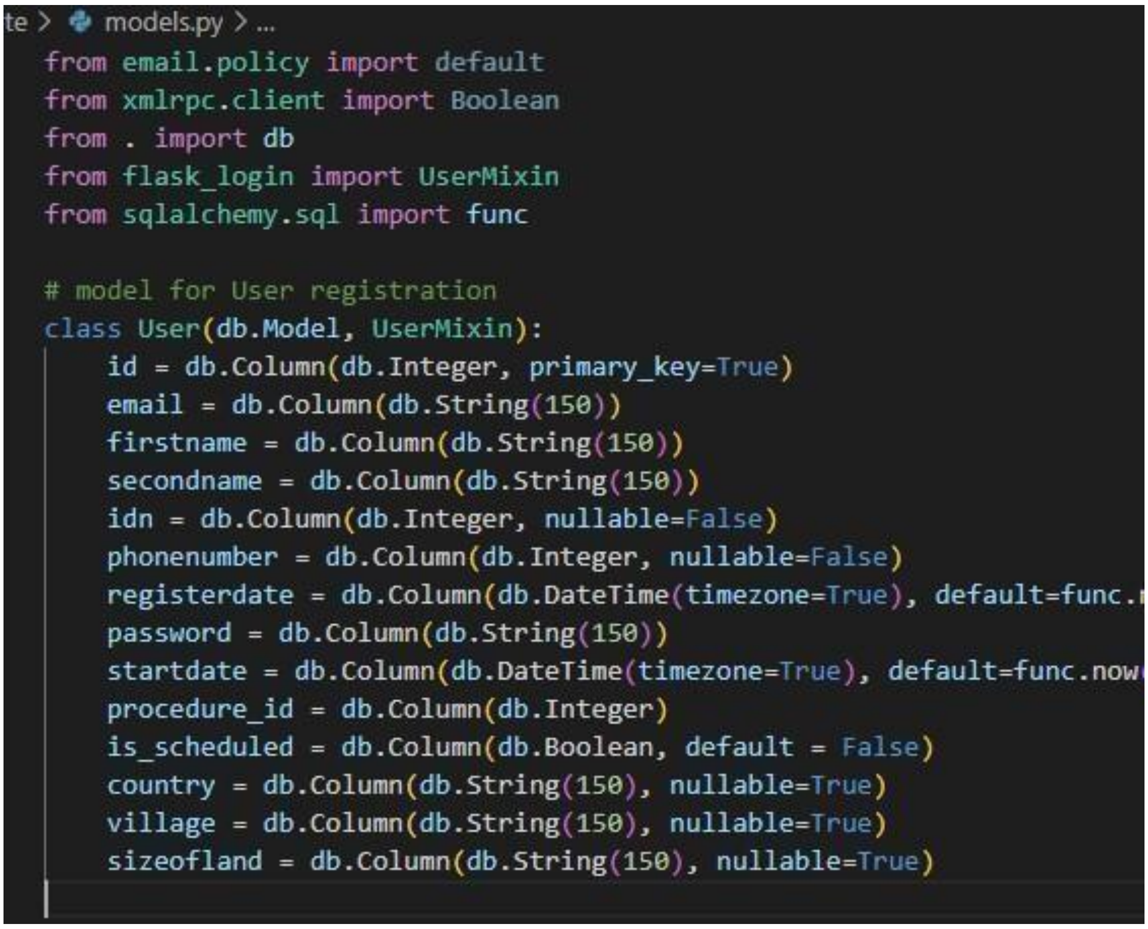
Before logging in we can access different parts of the information hence the information provided is relatable and informative.

11.3 Appendix B: program listing extract

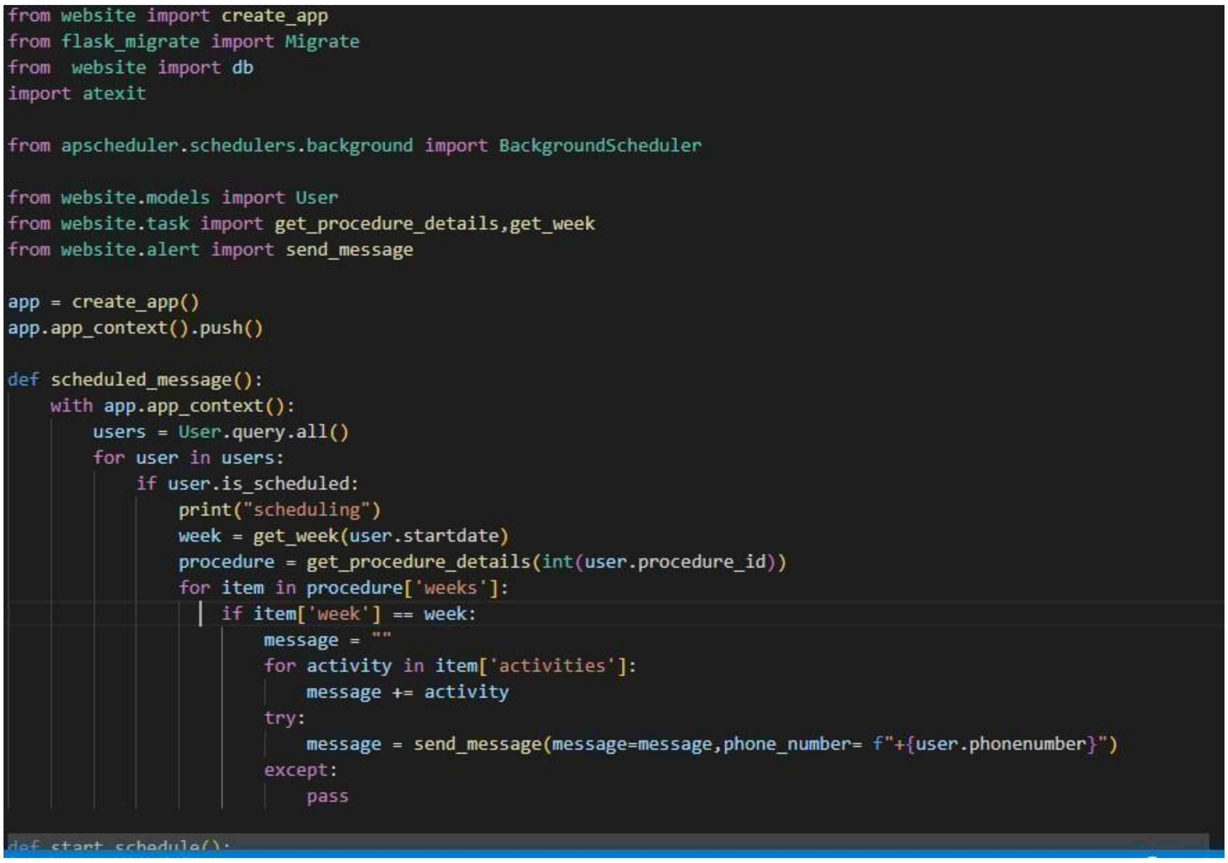
1 User model



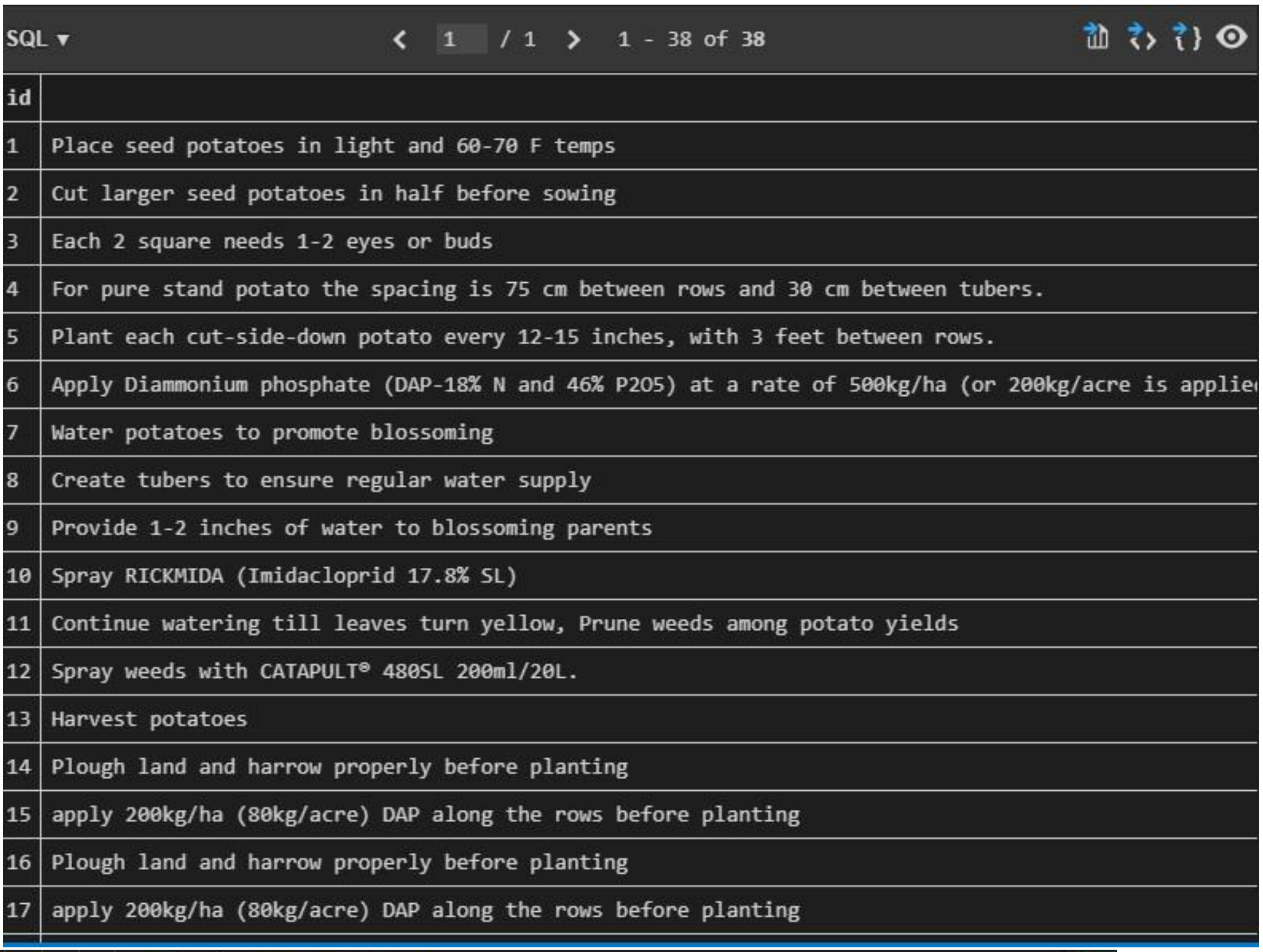
```python
from datetime import datetime
from flask import Blueprint, render_template, request, flash, redirect, url_for, session

from functools import wraps

from werkzeug.security import generate_password_hash, check_password_hash
from . import db
from .models import User

from flask_login import login_user, login_required, logout_user, current_user


auth = Blueprint('auth', __name__)

@auth.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        email = request.form.get('email')
        password = request.form.get('password')
        user = User.query.filter_by(email=email).first()
        if user:
            if check_password_hash(user.password, password):
                session['login_required']= True
                flash('Logged in successfully!', category='success')
                login_user(user, remember=True)
                return redirect(url_for('views.dashboard'))
                # return  render_template('dashboard.html')
            else:
                flash('Incorrect password, try again.', category='error')
        else:
            flash('Email does not exist.', category='error')
```



```
te > models.py > ...
  from email.policy import default
  from xmlrpc.client import Boolean
  from . import db
  from flask_login import UserMixin
  from sqlalchemy.sql import func

  # model for User registration
  class User(db.Model, UserMixin):
      id = db.Column(db.Integer, primary_key=True)
      email = db.Column(db.String(150))
      firstname = db.Column(db.String(150))
      secondname = db.Column(db.String(150))
      idn = db.Column(db.Integer, nullable=False)
      phonenumber = db.Column(db.Integer, nullable=False)
      registerdate = db.Column(db.DateTime(timezone=True), default=func.
      password = db.Column(db.String(150))
      startdate = db.Column(db.DateTime(timezone=True), default=func.now
      procedure_id = db.Column(db.Integer)
      is_scheduled = db.Column(db.Boolean, default = False)
      country = db.Column(db.String(150), nullable=True)
      village = db.Column(db.String(150), nullable=True)
      sizeofland = db.Column(db.String(150), nullable=True)
```

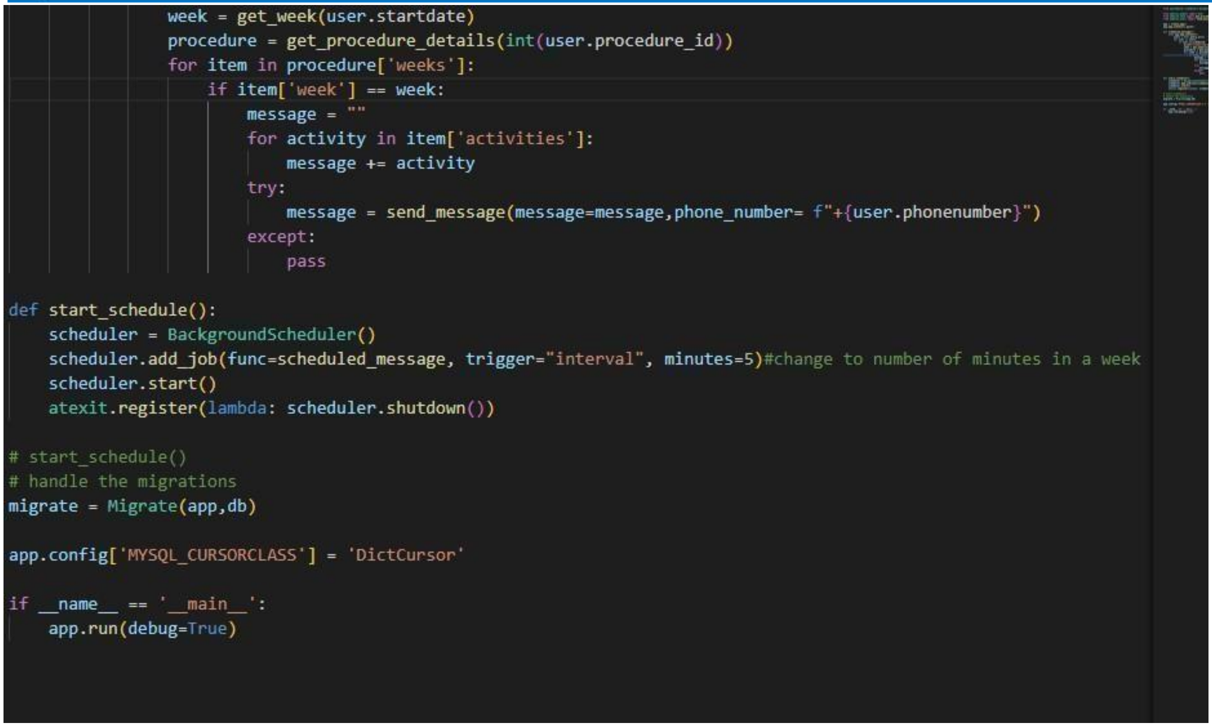


```
from website import create_app
from flask_migrate import Migrate
from  website import db
import atexit

from apscheduler.schedulers.background import BackgroundScheduler

from website.models import User
from website.task import get_procedure_details,get_week
from website.alert import send_message

app = create_app()
app.app_context().push()

def scheduled_message():
    with app.app_context():
        users = User.query.all()
        for user in users:
            if user.is_scheduled:
                print("scheduling")
                week = get_week(user.startdate)
                procedure = get_procedure_details(int(user.procedure_id))
                for item in procedure['weeks']:
                    if item['week'] == week:
                        message = ""
                        for activity in item['activities']:
                            message += activity
                        try:
                            message = send_message(message=message,phone_number= f"+{user.phonenumber}")
                        except:
                            pass

def start_schedule():
```

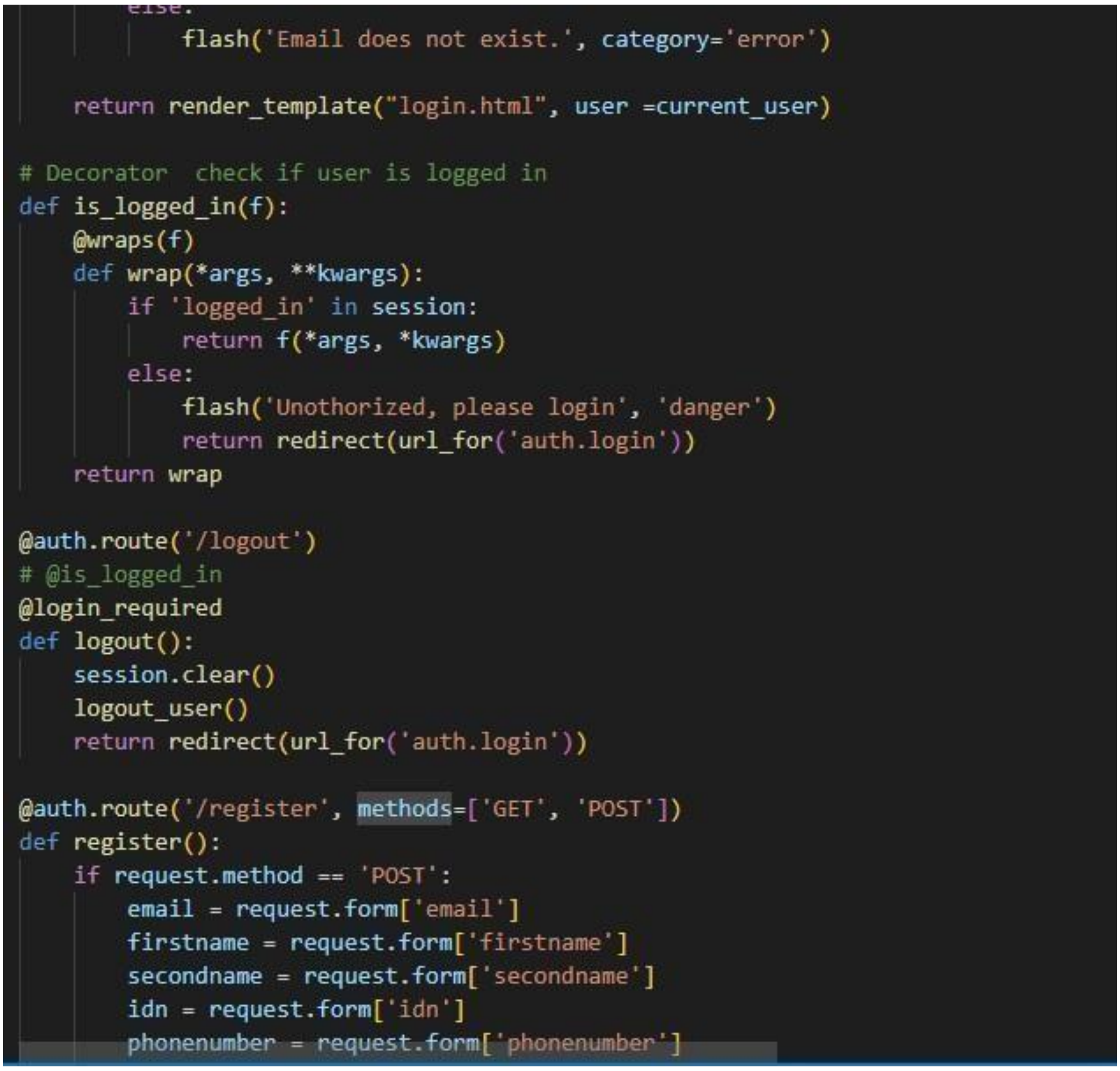


SQL ▾ 1 / 1 1 - 38 of 38

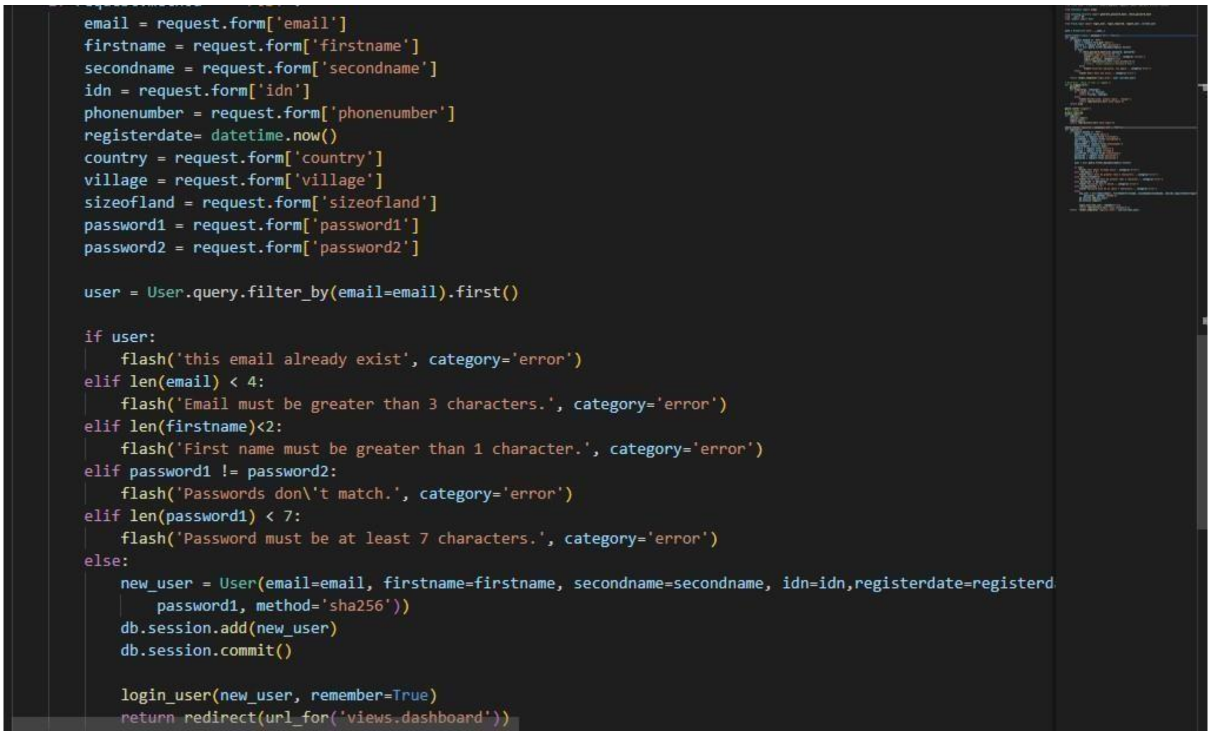
id	
1	Place seed potatoes in light and 60-70 F temps
2	Cut larger seed potatoes in half before sowing
3	Each 2 square needs 1-2 eyes or buds
4	For pure stand potato the spacing is 75 cm between rows and 30 cm between tubers.
5	Plant each cut-side-down potato every 12-15 inches, with 3 feet between rows.
6	Apply Diammonium phosphate (DAP-18% N and 46% P2O5) at a rate of 500kg/ha (or 200kg/acre is applie
7	Water potatoes to promote blossoming
8	Create tubers to ensure regular water supply
9	Provide 1-2 inches of water to blossoming parents
10	Spray RICKMIDA (Imidacloprid 17.8% SL)
11	Continue watering till leaves turn yellow, Prune weeds among potato yields
12	Spray weeds with CATAPULT® 480SL 200ml/20L.
13	Harvest potatoes
14	Plough land and harrow properly before planting
15	apply 200kg/ha (80kg/acre) DAP along the rows before planting
16	Plough land and harrow properly before planting
17	apply 200kg/ha (80kg/acre) DAP along the rows before planting



```
            week = get_week(user.startdate)
            procedure = get_procedure_details(int(user.procedure_id))
            for item in procedure['weeks']:
                if item['week'] == week:
                    message = ""
                    for activity in item['activities']:
                        message += activity
                    try:
                        message = send_message(message=message,phone_number= f"+{user.phonenumber}")
                    except:
                        pass

def start_schedule():
    scheduler = BackgroundScheduler()
    scheduler.add_job(func=scheduled_message, trigger="interval", minutes=5)#change to number of minutes in a week
    scheduler.start()
    atexit.register(lambda: scheduler.shutdown())

# start_schedule()
# handle the migrations
migrate = Migrate(app,db)

app.config['MYSQL_CURSORCLASS'] = 'DictCursor'

if __name__ == '__main__':
    app.run(debug=True)
```

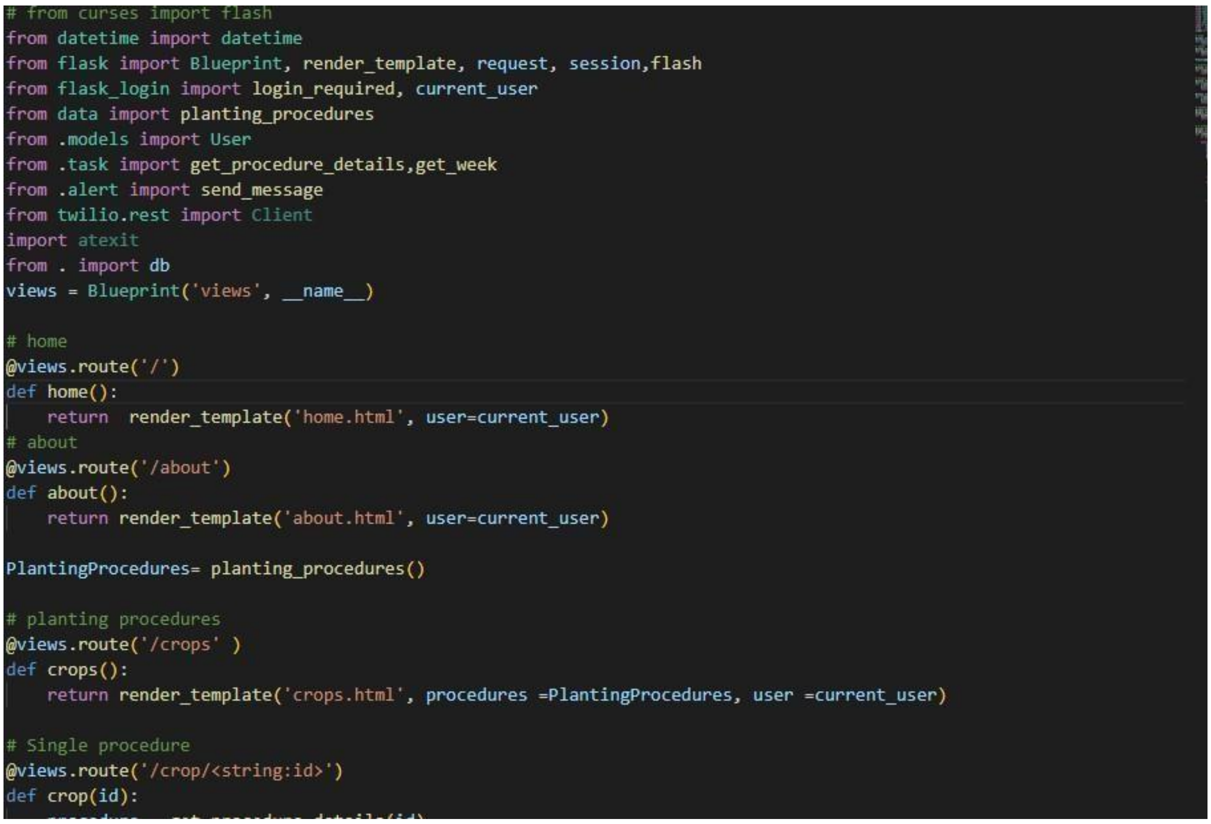


```
        else:
            flash('Email does not exist.', category='error')

    return render_template("login.html", user =current_user)

# Decorator  check if user is logged in
def is_logged_in(f):
    @wraps(f)
    def wrap(*args, **kwargs):
        if 'logged_in' in session:
            return f(*args, *kwargs)
        else:
            flash('Unothorized, please login', 'danger')
            return redirect(url_for('auth.login'))
    return wrap

@auth.route('/logout')
# @is_logged_in
@login_required
def logout():
    session.clear()
    logout_user()
    return redirect(url_for('auth.login'))

@auth.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        email = request.form['email']
        firstname = request.form['firstname']
        secondname = request.form['secondname']
        idn = request.form['idn']
        phonenumber = request.form['phonenumber']
```

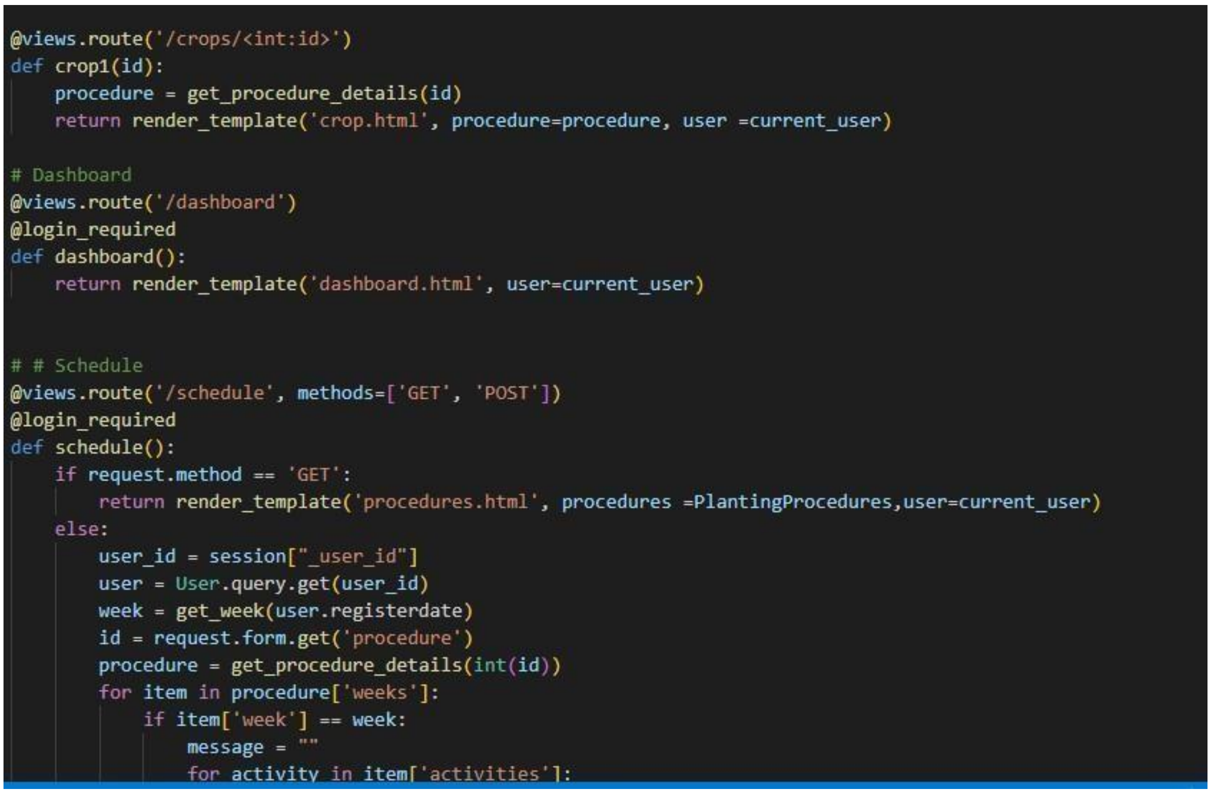


```
        email = request.form['email']
        firstname = request.form['firstname']
        secondname = request.form['secondname']
        idn = request.form['idn']
        phonenumber = request.form['phonenumber']
        registerdate= datetime.now()
        country = request.form['country']
        village = request.form['village']
        sizeofland = request.form['sizeofland']
        password1 = request.form['password1']
        password2 = request.form['password2']

        user = User.query.filter_by(email=email).first()

        if user:
            flash('this email already exist', category='error')
        elif len(email) < 4:
            flash('Email must be greater than 3 characters.', category='error')
        elif len(firstname)<2:
            flash('First name must be greater than 1 character.', category='error')
        elif password1 != password2:
            flash('Passwords don\'t match.', category='error')
        elif len(password1) < 7:
            flash('Password must be at least 7 characters.', category='error')
        else:
            new_user = User(email=email, firstname=firstname, secondname=secondname, idn=idn,registerdate=registerd
                password1, method='sha256'))
            db.session.add(new_user)
            db.session.commit()

            login_user(new_user, remember=True)
            return redirect(url_for('views.dashboard'))
```

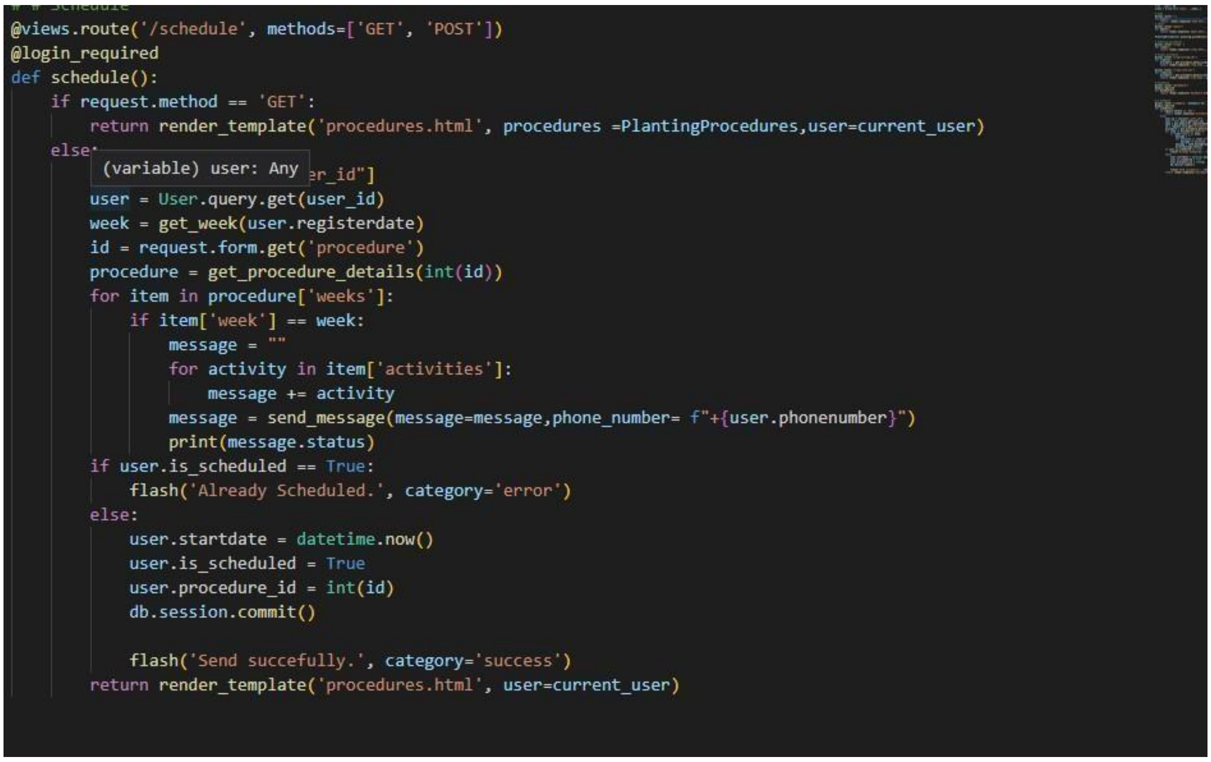
2 Farmer views



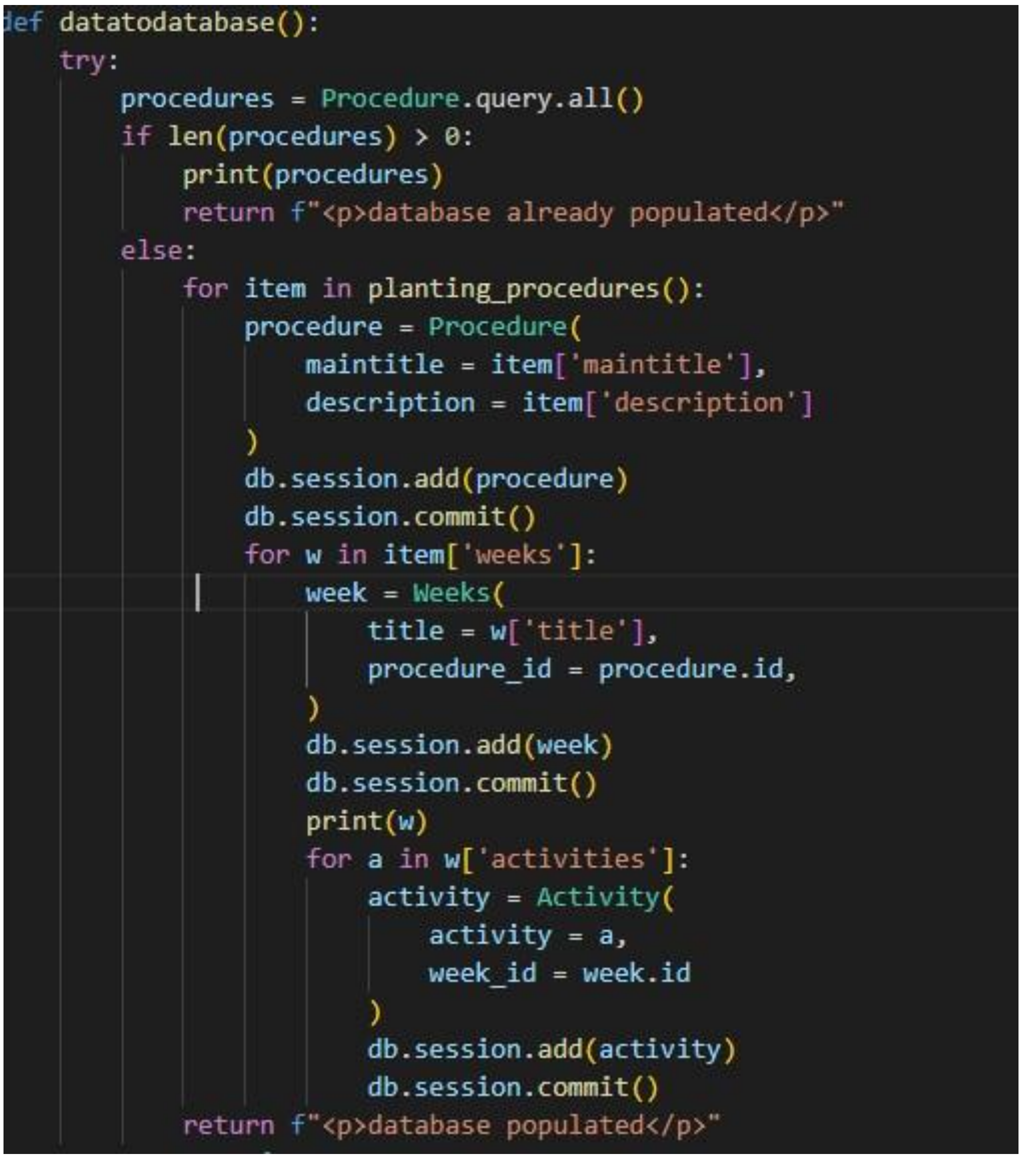
```
# from curses import flash
from datetime import datetime
from flask import Blueprint, render_template, request, session,flash
from flask_login import login_required, current_user
from data import planting_procedures
from .models import User
from .task import get_procedure_details,get_week
from .alert import send_message
from twilio.rest import Client
import atexit
from . import db
views = Blueprint('views', __name__)

# home
@views.route('/')
def home():
    return  render_template('home.html', user=current_user)
# about
@views.route('/about')
def about():
    return render_template('about.html', user=current_user)

PlantingProcedures= planting_procedures()

# planting procedures
@views.route('/crops' )
def crops():
    return render_template('crops.html', procedures =PlantingProcedures, user =current_user)

# Single procedure
@views.route('/crop/<string:id>')
def crop(id):
```



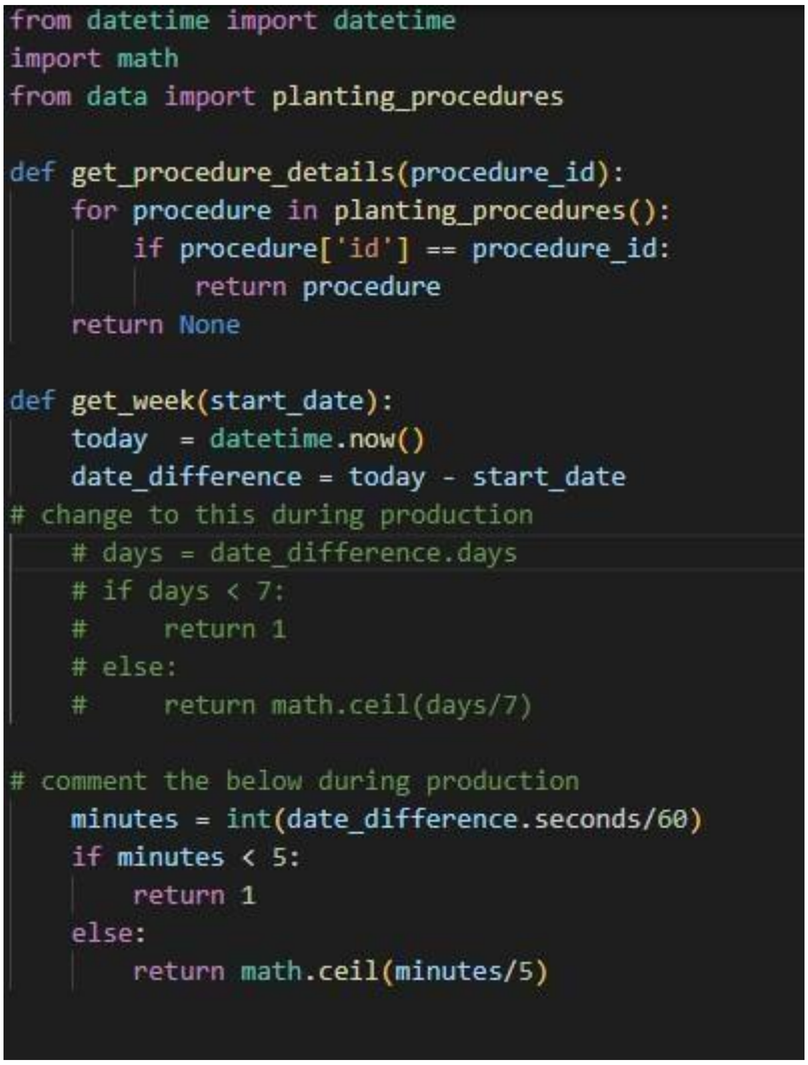
```python
@views.route('/crops/<int:id>')
def crop1(id):
    procedure = get_procedure_details(id)
    return render_template('crop.html', procedure=procedure, user =current_user)

# Dashboard
@views.route('/dashboard')
@login_required
def dashboard():
    return render_template('dashboard.html', user=current_user)


# # Schedule
@views.route('/schedule', methods=['GET', 'POST'])
@login_required
def schedule():
    if request.method == 'GET':
        return render_template('procedures.html', procedures =PlantingProcedures,user=current_user)
    else:
        user_id = session["_user_id"]
        user = User.query.get(user_id)
        week = get_week(user.registerdate)
        id = request.form.get('procedure')
        procedure = get_procedure_details(int(id))
        for item in procedure['weeks']:
            if item['week'] == week:
                message = ""
                for activity in item['activities']:
```



```python
@views.route('/schedule', methods=['GET', 'POST'])
@login_required
def schedule():
    if request.method == 'GET':
        return render_template('procedures.html', procedures =PlantingProcedures,user=current_user)
    else:
        (variable) user: Any er_id"]
        user = User.query.get(user_id)
        week = get_week(user.registerdate)
        id = request.form.get('procedure')
        procedure = get_procedure_details(int(id))
        for item in procedure['weeks']:
            if item['week'] == week:
                message = ""
                for activity in item['activities']:
                    message += activity
                message = send_message(message=message,phone_number= f"+{user.phonenumber}")
                print(message.status)
        if user.is_scheduled == True:
            flash('Already Scheduled.', category='error')
        else:
            user.startdate = datetime.now()
            user.is_scheduled = True
            user.procedure_id = int(id)
            db.session.commit()

            flash('Send succefully.', category='success')
        return render_template('procedures.html', user=current_user)
```

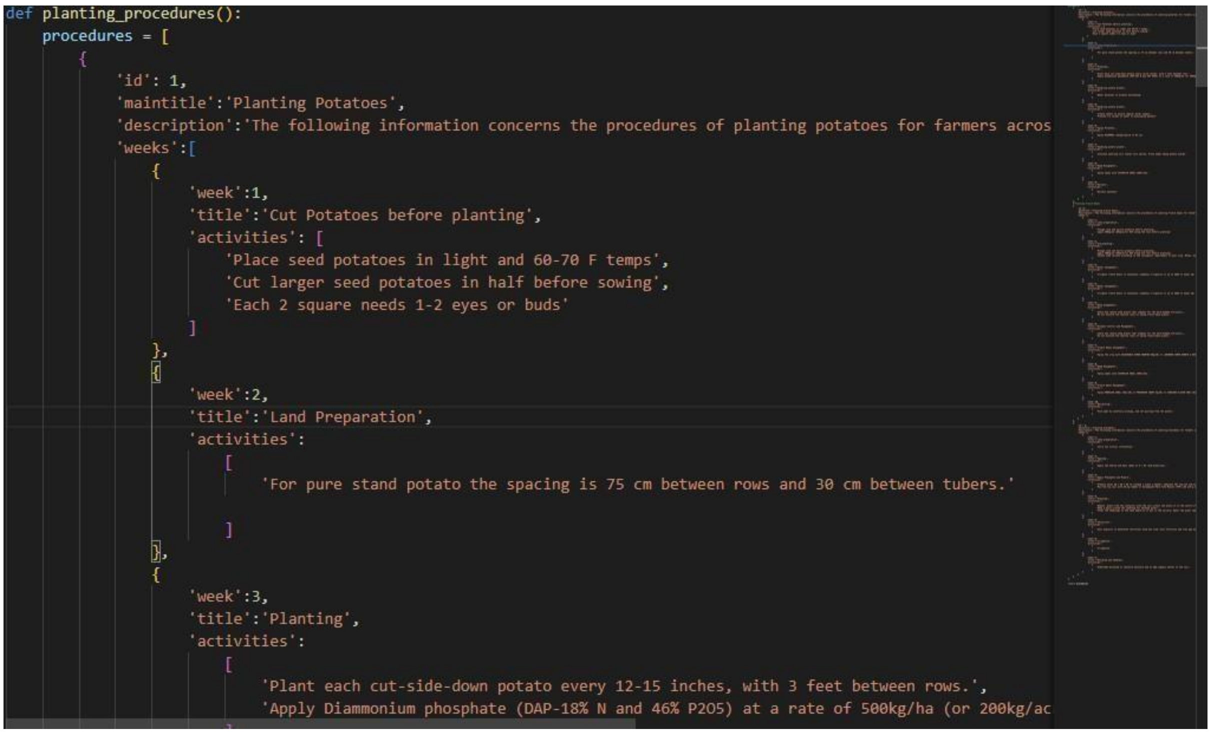


```
def datatodatabase():
    try:
        procedures = Procedure.query.all()
        if len(procedures) > 0:
            print(procedures)
            return f"<p>database already populated</p>"
        else:
            for item in planting_procedures():
                procedure = Procedure(
                    maintitle = item['maintitle'],
                    description = item['description']
                )
                db.session.add(procedure)
                db.session.commit()
                for w in item['weeks']:
                    week = Weeks(
                        title = w['title'],
                        procedure_id = procedure.id,
                    )
                    db.session.add(week)
                    db.session.commit()
                    print(w)
                    for a in w['activities']:
                        activity = Activity(
                            activity = a,
                            week_id = week.id
                        )
                        db.session.add(activity)
                        db.session.commit()
            return f"<p>database populated</p>"
```

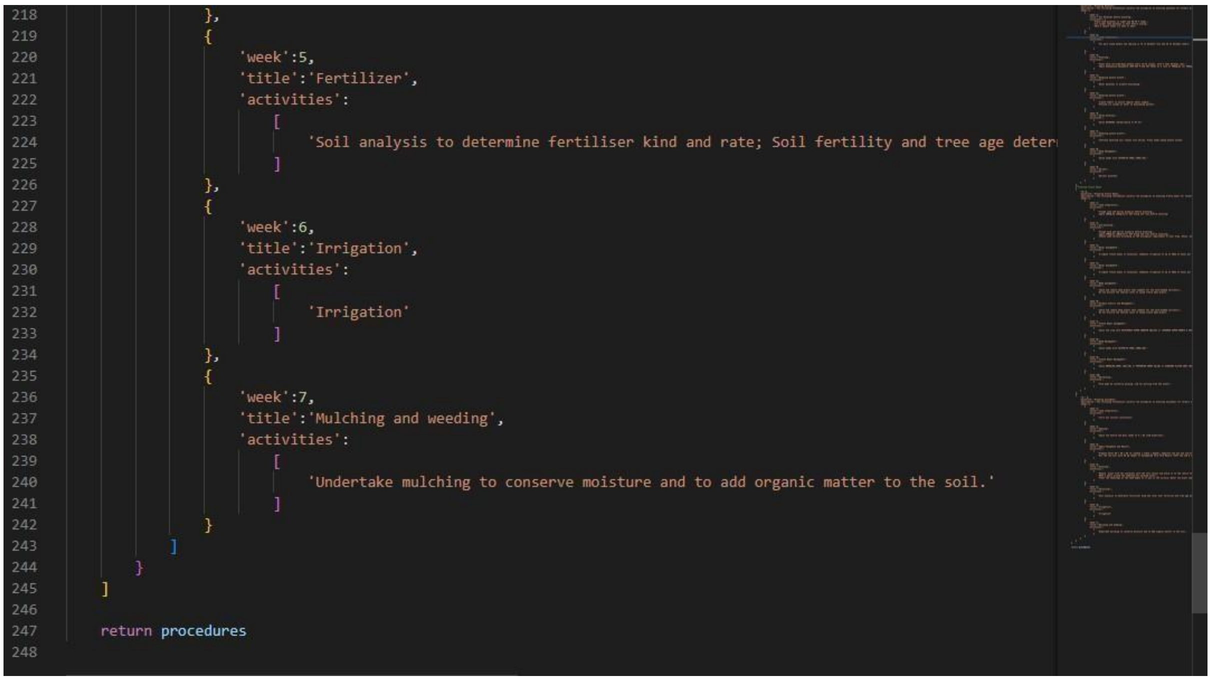


```python
from datetime import datetime
import math
from data import planting_procedures

def get_procedure_details(procedure_id):
    for procedure in planting_procedures():
        if procedure['id'] == procedure_id:
            return procedure
    return None

def get_week(start_date):
    today  = datetime.now()
    date_difference = today - start_date
# change to this during production
    # days = date_difference.days
    # if days < 7:
    #     return 1
    # else:
    #     return math.ceil(days/7)

# comment the below during production
    minutes = int(date_difference.seconds/60)
    if minutes < 5:
        return 1
    else:
        return math.ceil(minutes/5)
```

3 **Data function**

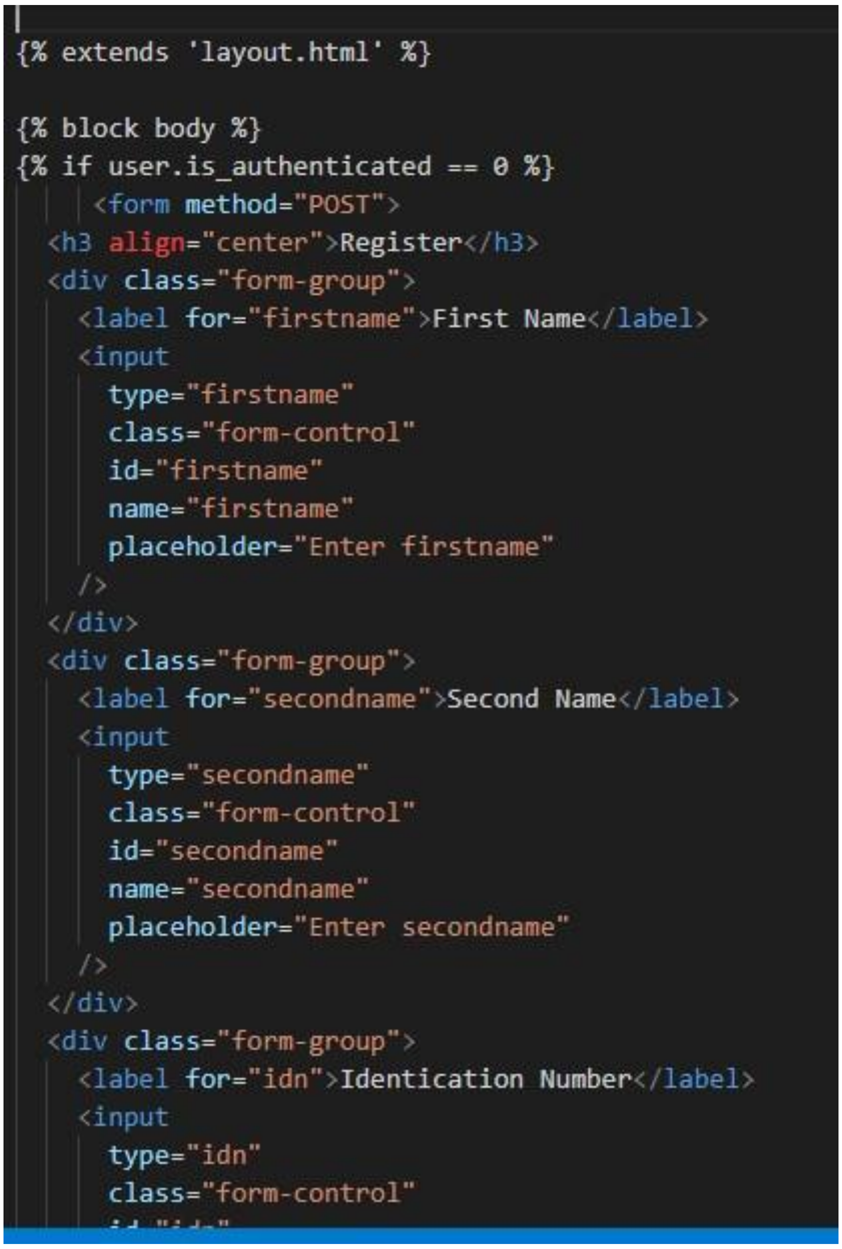


```python
def planting_procedures():
    procedures = [
        {
            'id': 1,
            'maintitle':'Planting Potatoes',
            'description':'The following information concerns the procedures of planting potatoes for farmers acros
            'weeks':[
                {
                    'week':1,
                    'title':'Cut Potatoes before planting',
                    'activities': [
                        'Place seed potatoes in light and 60-70 F temps',
                        'Cut larger seed potatoes in half before sowing',
                        'Each 2 square needs 1-2 eyes or buds'
                    ]
                },
                {
                    'week':2,
                    'title':'Land Preparation',
                    'activities':
                        [
                            'For pure stand potato the spacing is 75 cm between rows and 30 cm between tubers.'

                        ]
                },
                {
                    'week':3,
                    'title':'Planting',
                    'activities':
                        [
                            'Plant each cut-side-down potato every 12-15 inches, with 3 feet between rows.',
                            'Apply Diammonium phosphate (DAP-18% N and 46% P2O5) at a rate of 500kg/ha (or 200kg/ac
```



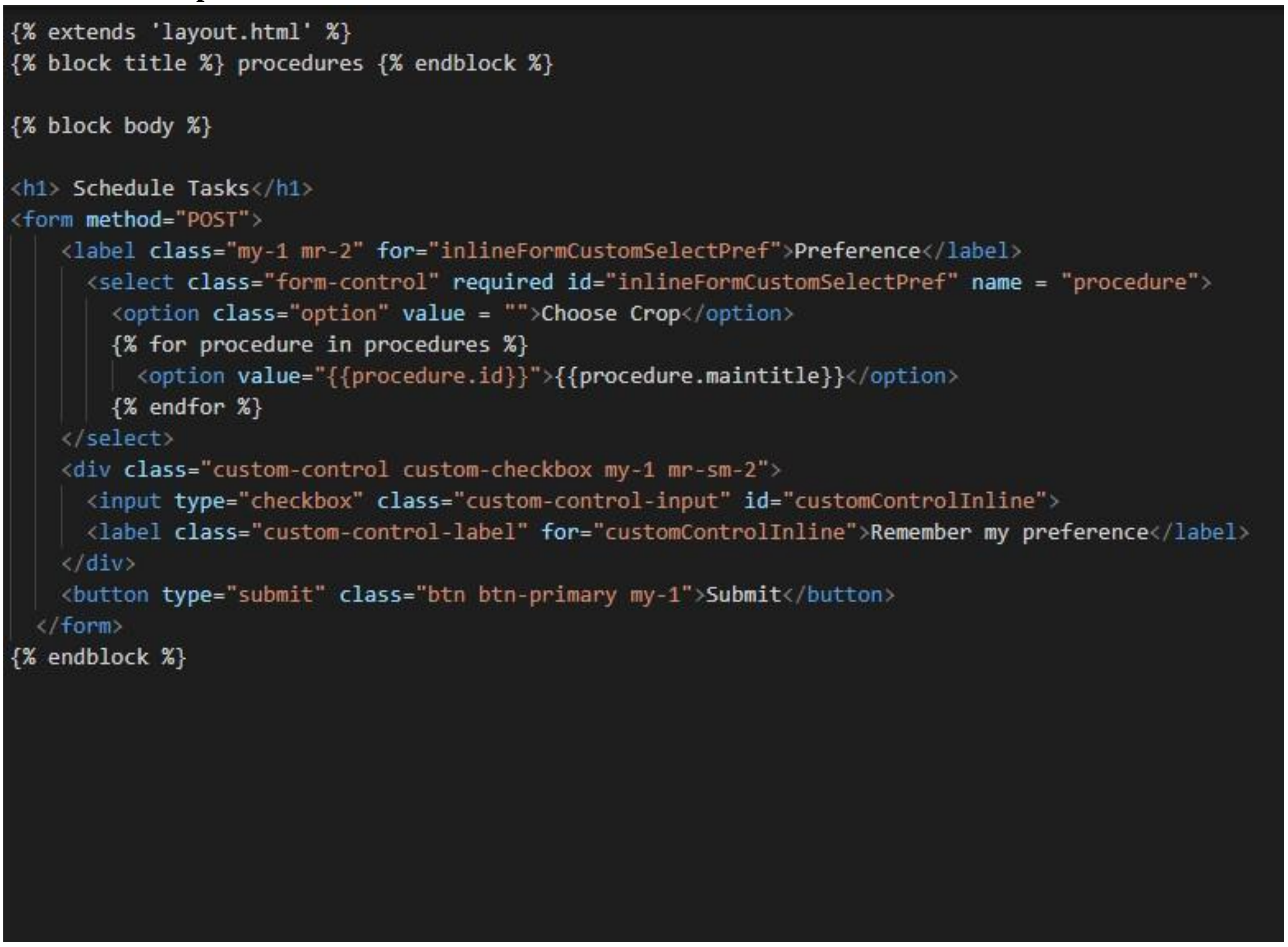
```
                },
                {
                    'week':5,
                    'title':'Fertilizer',
                    'activities':
                        [
                            'Soil analysis to determine fertiliser kind and rate; Soil fertility and tree age deter
                        ]
                },
                {
                    'week':6,
                    'title':'Irrigation',
                    'activities':
                        [
                            'Irrigation'
                        ]
                },
                {
                    'week':7,
                    'title':'Mulching and weeding',
                    'activities':
                        [
                            'Undertake mulching to conserve moisture and to add organic matter to the soil.'
                        ]
                }
            ]
        }
    ]

    return procedures
```

Register form



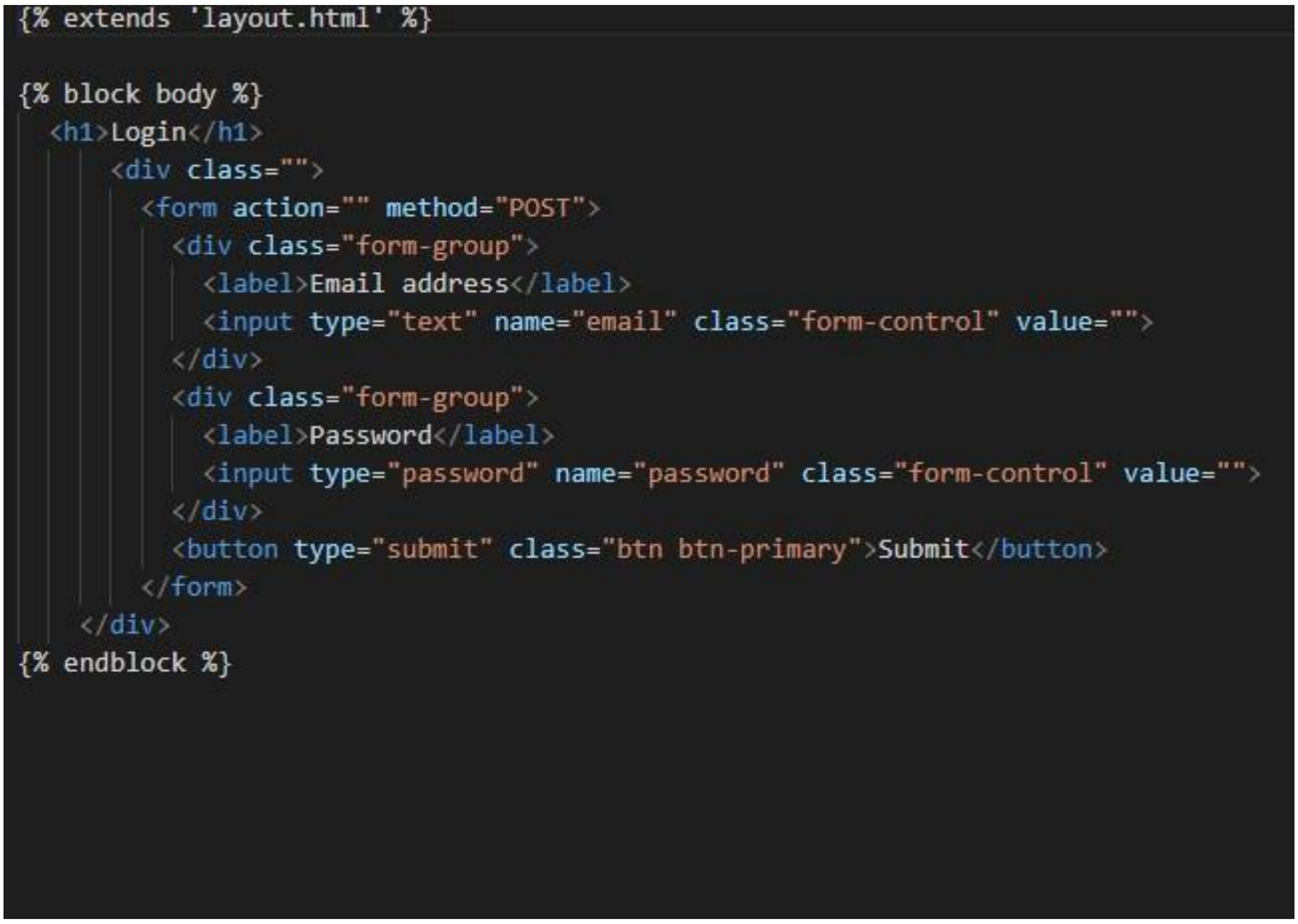
```
{% extends 'layout.html' %}

{% block body %}
{% if user.is_authenticated == 0 %}
      <form method="POST">
  <h3 align="center">Register</h3>
  <div class="form-group">
    <label for="firstname">First Name</label>
    <input
      type="firstname"
      class="form-control"
      id="firstname"
      name="firstname"
      placeholder="Enter firstname"
    />
  </div>
  <div class="form-group">
    <label for="secondname">Second Name</label>
    <input
      type="secondname"
      class="form-control"
      id="secondname"
      name="secondname"
      placeholder="Enter secondname"
    />
  </div>
  <div class="form-group">
    <label for="idn">Identication Number</label>
    <input
      type="idn"
      class="form-control"
```

Procedures template



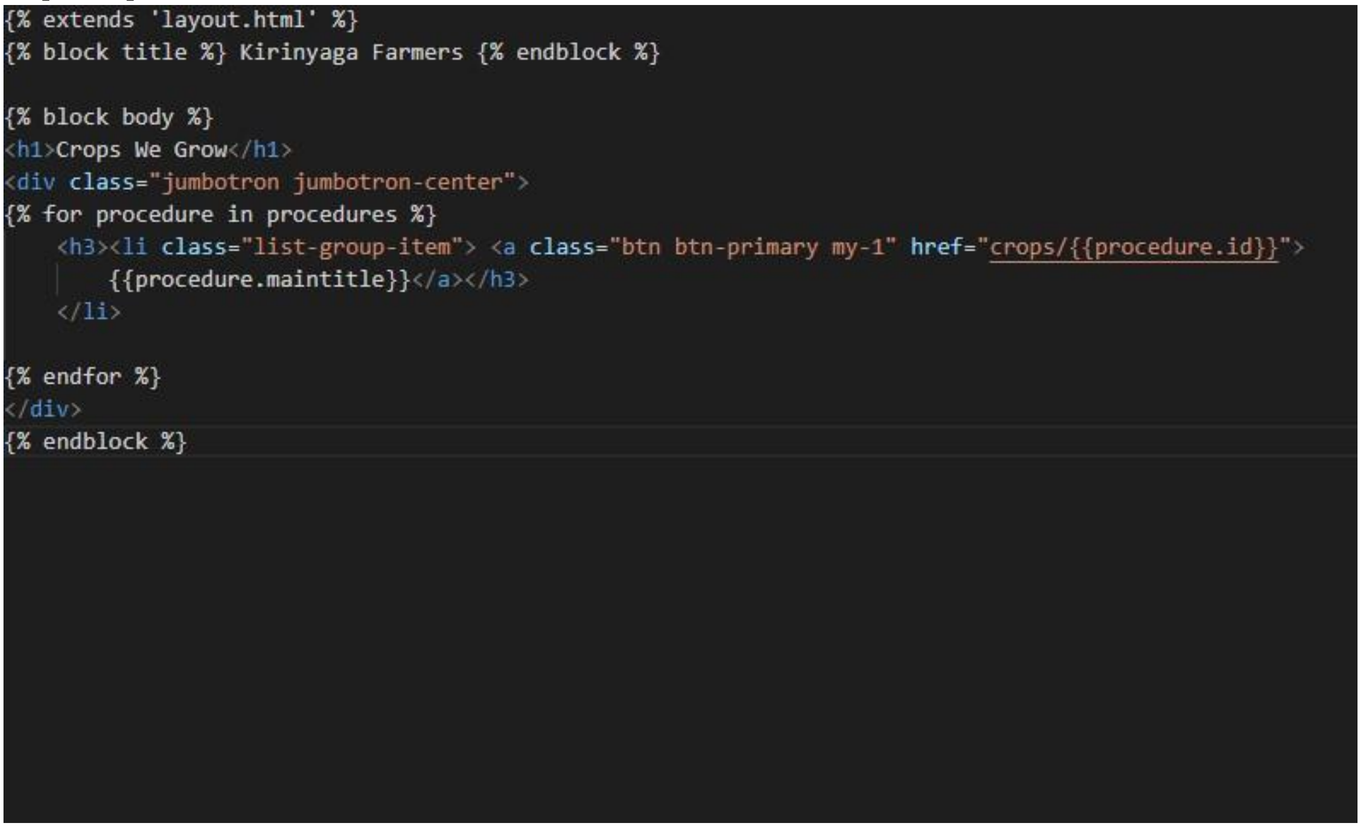
```
{% extends 'layout.html' %}
{% block title %} procedures {% endblock %}

{% block body %}

<h1> Schedule Tasks</h1>
<form method="POST">
    <label class="my-1 mr-2" for="inlineFormCustomSelectPref">Preference</label>
      <select class="form-control" required id="inlineFormCustomSelectPref" name = "procedure">
        <option class="option" value = "">Choose Crop</option>
        {% for procedure in procedures %}
          <option value="{{procedure.id}}">{{procedure.maintitle}}</option>
        {% endfor %}
    </select>
    <div class="custom-control custom-checkbox my-1 mr-sm-2">
      <input type="checkbox" class="custom-control-input" id="customControlInline">
      <label class="custom-control-label" for="customControlInline">Remember my preference</label>
    </div>
    <button type="submit" class="btn btn-primary my-1">Submit</button>
  </form>
{% endblock %}
```

Login form



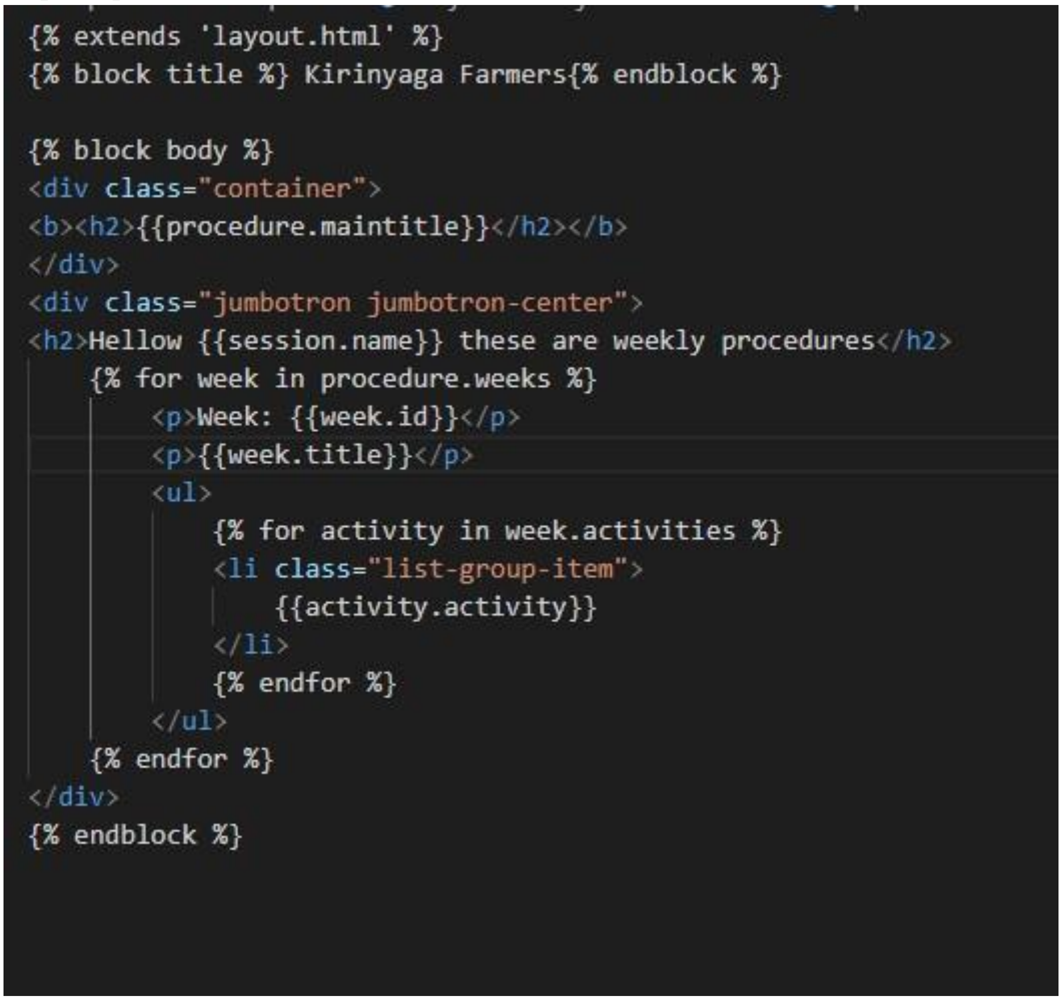
```
{% extends 'layout.html' %}

{% block body %}
  <h1>Login</h1>
      <div class="">
        <form action="" method="POST">
          <div class="form-group">
            <label>Email address</label>
            <input type="text" name="email" class="form-control" value="">
          </div>
          <div class="form-group">
            <label>Password</label>
            <input type="password" name="password" class="form-control" value="">
          </div>
          <button type="submit" class="btn btn-primary">Submit</button>
        </form>
    </div>
{% endblock %}
```

Crops template



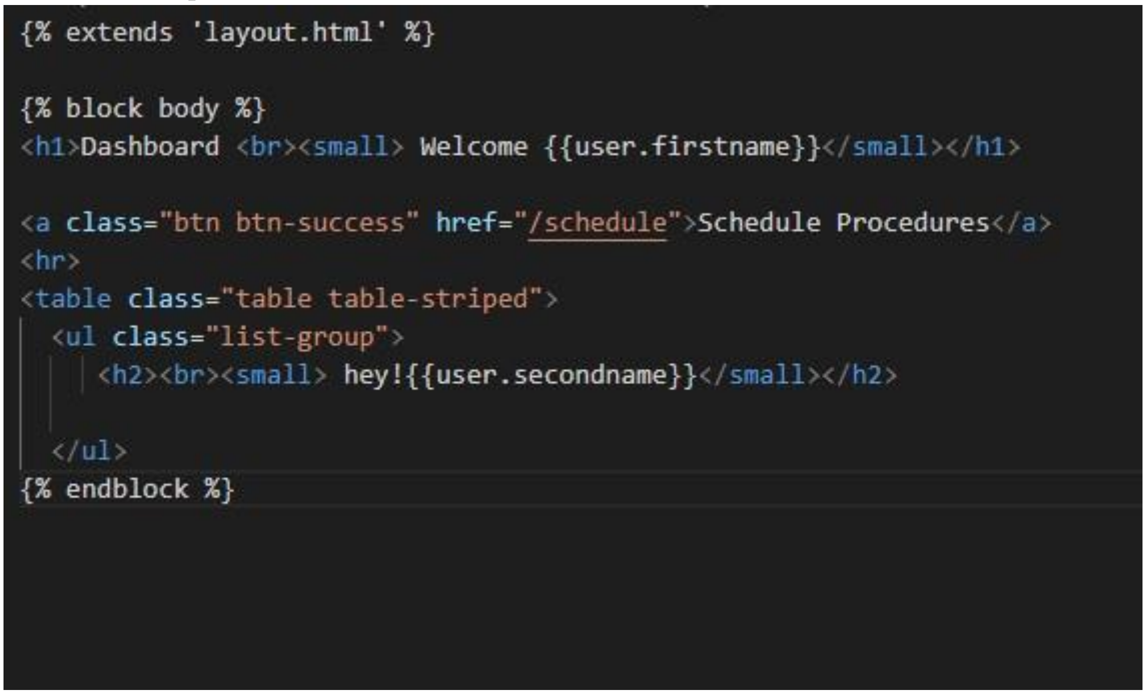
```
{% extends 'layout.html' %}
{% block title %} Kirinyaga Farmers {% endblock %}

{% block body %}
<h1>Crops We Grow</h1>
<div class="jumbotron jumbotron-center">
{% for procedure in procedures %}
    <h3><li class="list-group-item"> <a class="btn btn-primary my-1" href="crops/{{procedure.id}}">
        {{procedure.maintitle}}</a></h3>
    </li>

{% endfor %}
</div>
{% endblock %}
```

Crop template



```
{% extends 'layout.html' %}
{% block title %} Kirinyaga Farmers{% endblock %}

{% block body %}
<div class="container">
<b><h2>{{procedure.maintitle}}</h2></b>
</div>
<div class="jumbotron jumbotron-center">
<h2>Hellow {{session.name}} these are weekly procedures</h2>
    {% for week in procedure.weeks %}
        <p>Week: {{week.id}}</p>
        <p>{{week.title}}</p>
        <ul>
            {% for activity in week.activities %}
            <li class="list-group-item">
                {{activity.activity}}
            </li>
            {% endfor %}
        </ul>
    {% endfor %}
</div>
{% endblock %}
```

Dashboard template



```
{% extends 'layout.html' %}

{% block body %}
<h1>Dashboard <br><small> Welcome {{user.firstname}}</small></h1>

<a class="btn btn-success" href="/schedule">Schedule Procedures</a>
<hr>
<table class="table table-striped">
  <ul class="list-group">
     <h2><br><small> hey!{{user.secondname}}</small></h2>

  </ul>
{% endblock %}
```