**Coffee Project**

**Overall Documentation**

**1. Code structure & Functionalities**

**1.1 Application Life Cycle**

Welcome and login

**User input**

Userinput page

(1) Build Excel Model

(2) Replace data with user input data

(3) Recalculate

**User view pages**

Bar chart for the calculated result

Go back or

5 mins No activity

**1.2 Excel Modeling**

FlexcelSum.cs

Each of the sheets in the excel file has c# code related to it. The calculations are done in all of them and then the output is obtained from one of the sheets.

This c# file handles all the Excel model building procedures. The application uses FlexCel to convert the excel file into C# code (hardcode all the information of the Excel file into C# code). When user inputs all his values to the input page and click submit, the server executes the following steps:

1. Create an in-memory database storing all the information about the target Excel file by running the C# converted code and store it as an object. This object can be seen as a “virtual Excel file”. The object has all the default value of each cell (The values stored in the Excel file when it’s converted into C# code before)
2. Take all the user input values and replace all the values of the corresponding cells in this object. Now the object stores the updated values.
3. The object does an internal recalculate procedure to update all other cells that are related to the changed cells (Excel formulas, etc.)
4. The server returns back the pages that the user wants to view (Each of the pages uses the updated values stored in this object by grabbing data from it and generate the dynamic website)

The flow goes as the following –

User logs in and then we take him/her to the input page where all the input boxes are populated with 0 if the person is logging in for the first time or if the person doesn’t have any inputs saved.

The Person can enter the input values and then press the submit button.

After pressing the submit button the output gets calculated and the inputs and the outputs get saved.

When the person logs back in again, the input gets populated with the latest input which the person had entered and then if there is output, the app will go straight to the chart page.

Deployment to the server in Brazil:

Through Visual Studio –

Server: 104.41.49.69:8172

Site Name: Default Web Site

User name: AVFCoffeeBrazil\AVFadmin

Password: i#w3G43LzFbQy9$fv^j4

Destination URL: <https://avfcoffee.brazilsouth.cloudapp.azure.com>

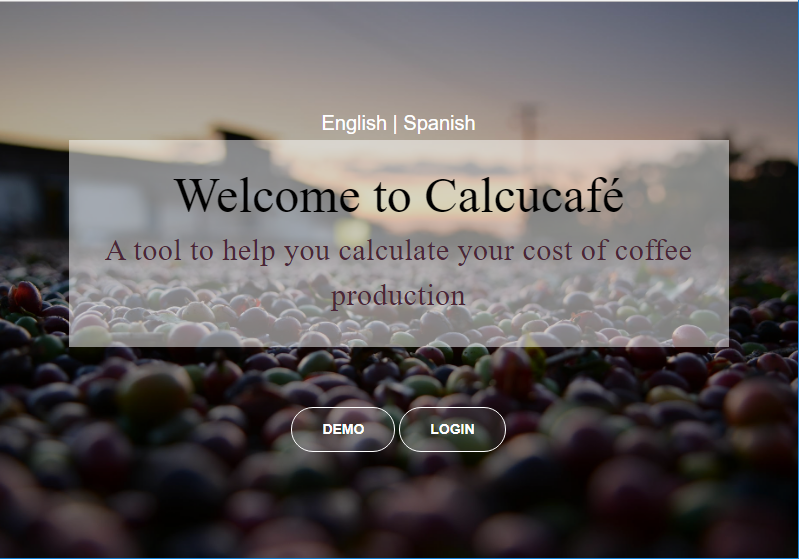
Make sure that there are no unused DLLs present while you are deploying to the server, otherwise there will be problems in deploying. Microsoft.entityframework should be deployed. Also, make sure to use Microsoft Core 2 runtime SDK in the server and also core 2 preview package in the application.

**1.3 Websites and Files**

**(1) User pages HTML:**

* Welcome page

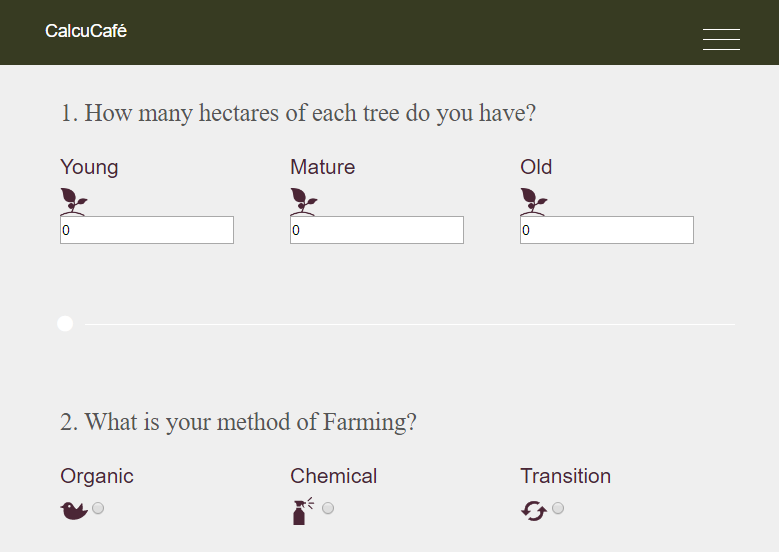
The user can wish to watch the demo and go to the sign in page or the user can go to the login page directly.

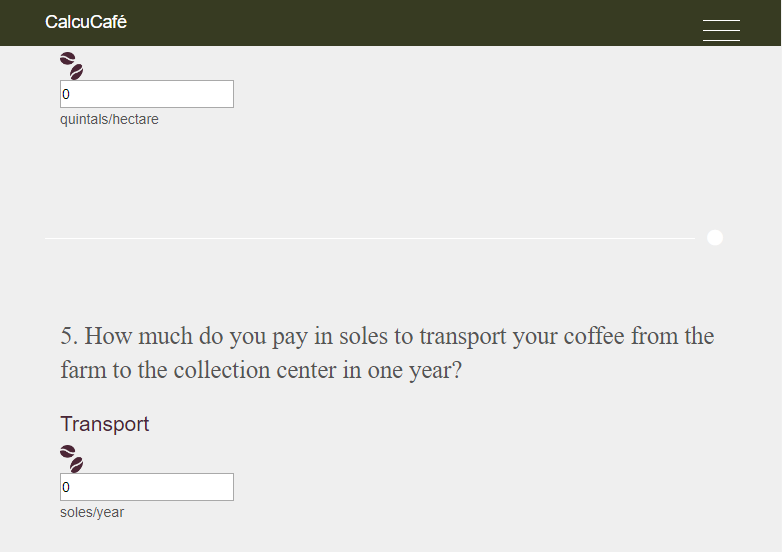


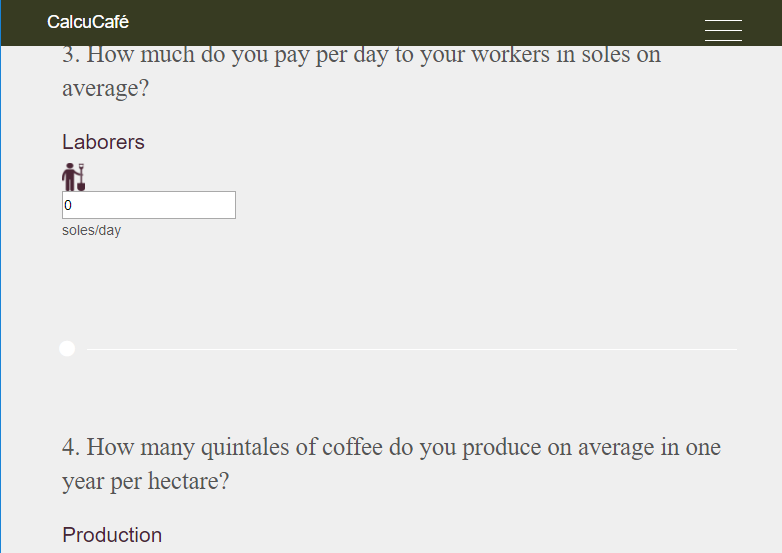
* UserInputPage

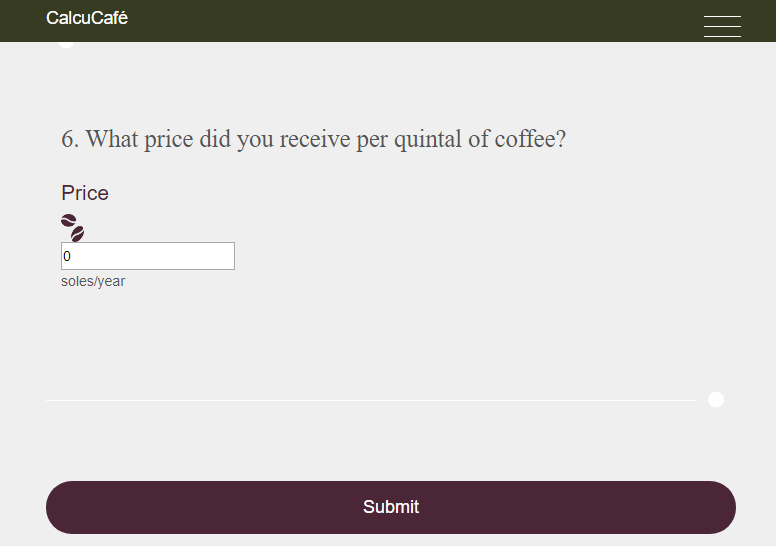
These pages are for the user to provide their input and by default the values are put in as 0s.

…..

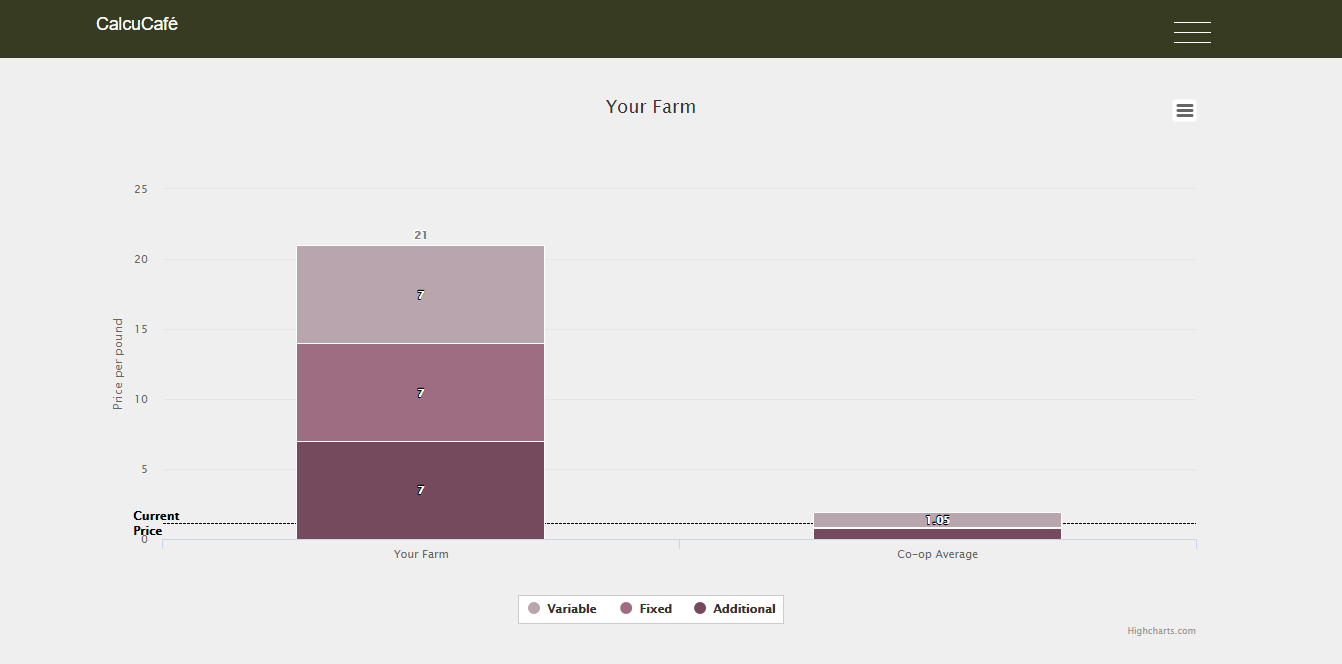








Once they press submit they have the chart created for them which looks like this

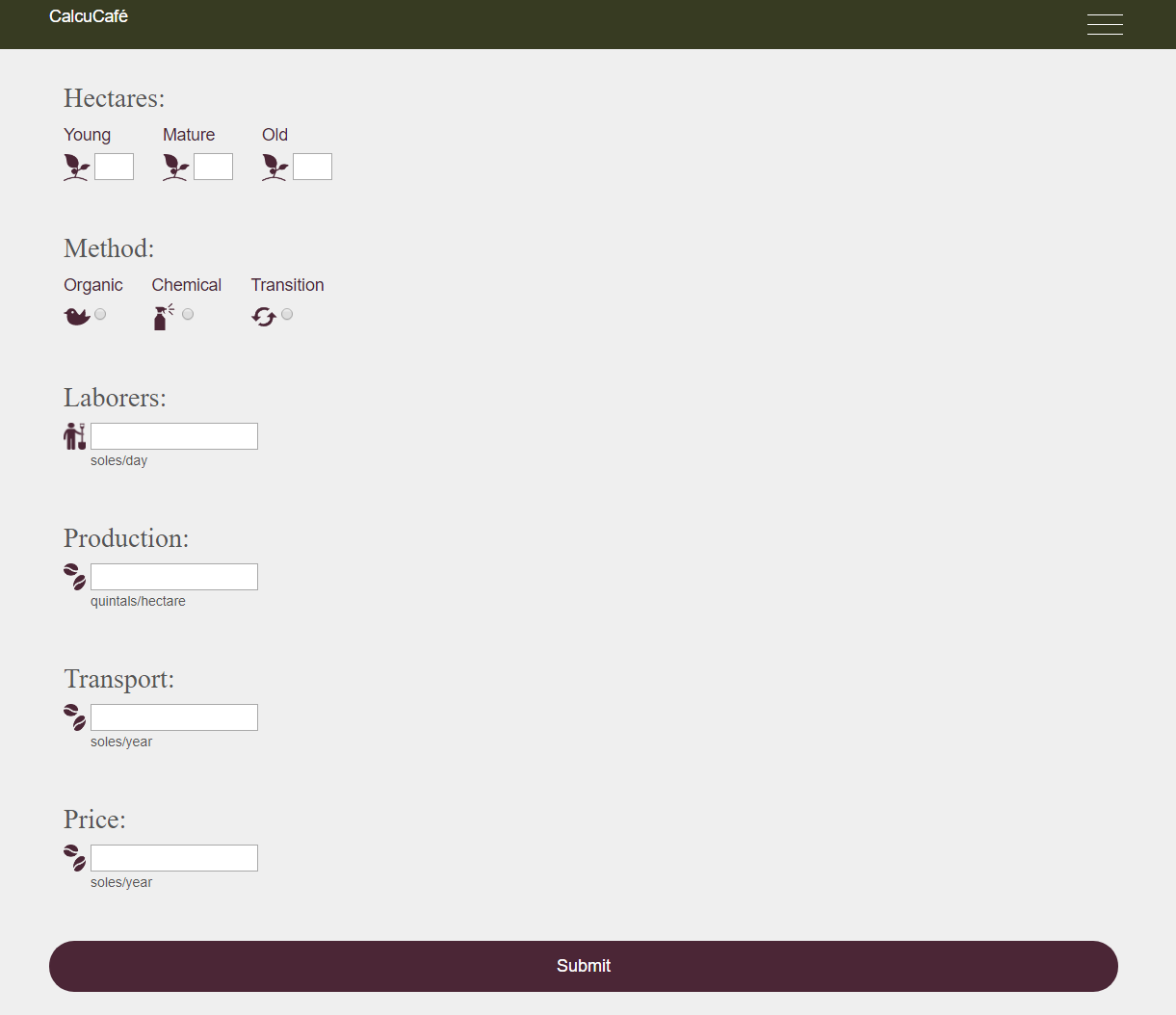


This is calculated from the excel sheet functionalities which was given to us.

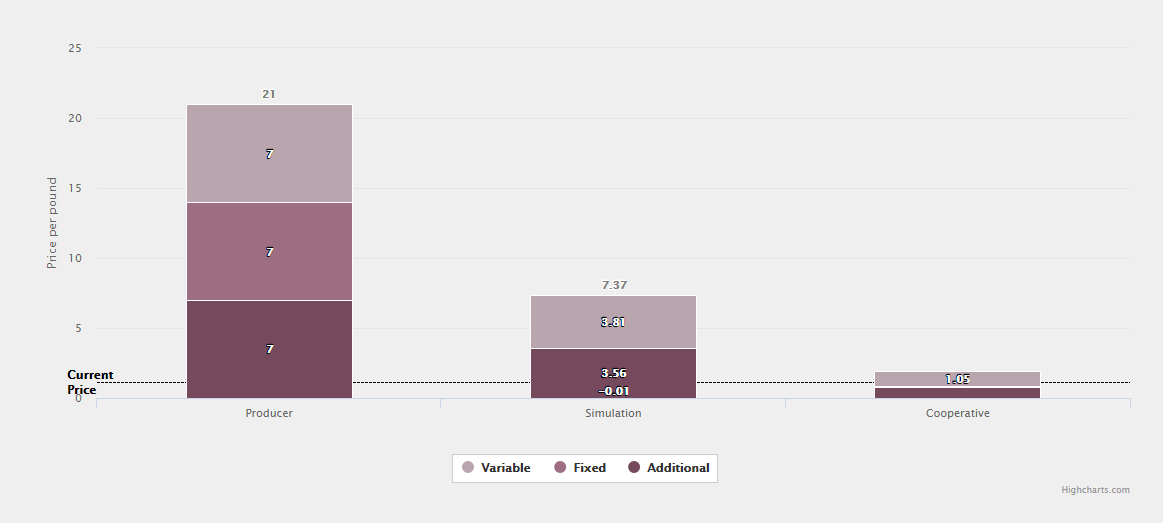
Menu dropdown from the top



* Simulation page



After clicking on the submit button this page will come up.



The producer, simulation and cooperative for the inputs and the simulation.

**2. Recreate Procedure**

When Excel file is updated, first thing is to rebuild the Excel model. i.e convert the new Excel file to C# code again if big changes, or change the converted C# code if small changes.

When convert the Excel file to C# code. Follow the steps:

1. Open FlexCel and give input of your Excel file. Choose to convert to C# code
2. Once the code is converted, check the dropdown menu to see if all sheets are existent
3. Once the code is converted, briefly check one of the sheets to see if the data matches
4. It should be made sure that there are no links to non-existent files in the code otherwise the code will break and it will be hard to detect.

1. Pasted the converted code into the corresponding section (functions) of the FlexcelSum.cs file to replace the old code.
2. Check the user pages backend code to correct linking problems if there is any due to new Excel file moved some cell locations or new cells are generated

The login to the database is as follows –

Normal user:

User id: AVFUser

Password: coffeeuser

Admin user:

User id: AVFadmin

Password: i#w3G43LzFbQy9$fv^j4