Implementation Documentation:

This document intends to thoroughly explain the implementation of the important modules and functions of the system.

Form Modules:

The main thing we want to do on our website is create a nice way to add and edit blog posts. With forms we will have absolute power over our interface – we can do almost anything we can imagine!

AVI Module Forms:

We use Django built in form creator to create a basic form which will then have to pass our required attributes(which invoke database usage- to be discussed later) to it - ‘from Django import forms’.

### The Django [Form](https://docs.djangoproject.com/en/2.1/ref/forms/api/#django.forms.Form) class[¶](https://docs.djangoproject.com/en/2.1/topics/forms/#the-django-form-class)

At the heart of this system of components is Django’s [**Form**](https://docs.djangoproject.com/en/2.1/ref/forms/api/#django.forms.Form) class. In much the same way that a Django model describes the logical structure of an object, its behavior, and the way its parts are represented to us, a [**Form**](https://docs.djangoproject.com/en/2.1/ref/forms/api/#django.forms.Form) class describes a form and determines how it works and appears.

Django’s role in forms[**¶**](https://docs.djangoproject.com/en/2.1/topics/forms/#django-s-role-in-forms)

Handling forms is a complex business. Consider Django’s admin, where numerous items of data of several different types may need to be prepared for display in a form, rendered as HTML, edited using a convenient interface, returned to the server, validated and cleaned up, and then saved or passed on for further processing.

Django’s form functionality can simplify and automate vast portions of this work, and can also do it more securely than most programmers would be able to do in code they wrote themselves.

Django handles three distinct parts of the work involved in forms:

* preparing and restructuring data to make it ready for rendering
* creating HTML forms for the data
* receiving and processing submitted forms and data from the client

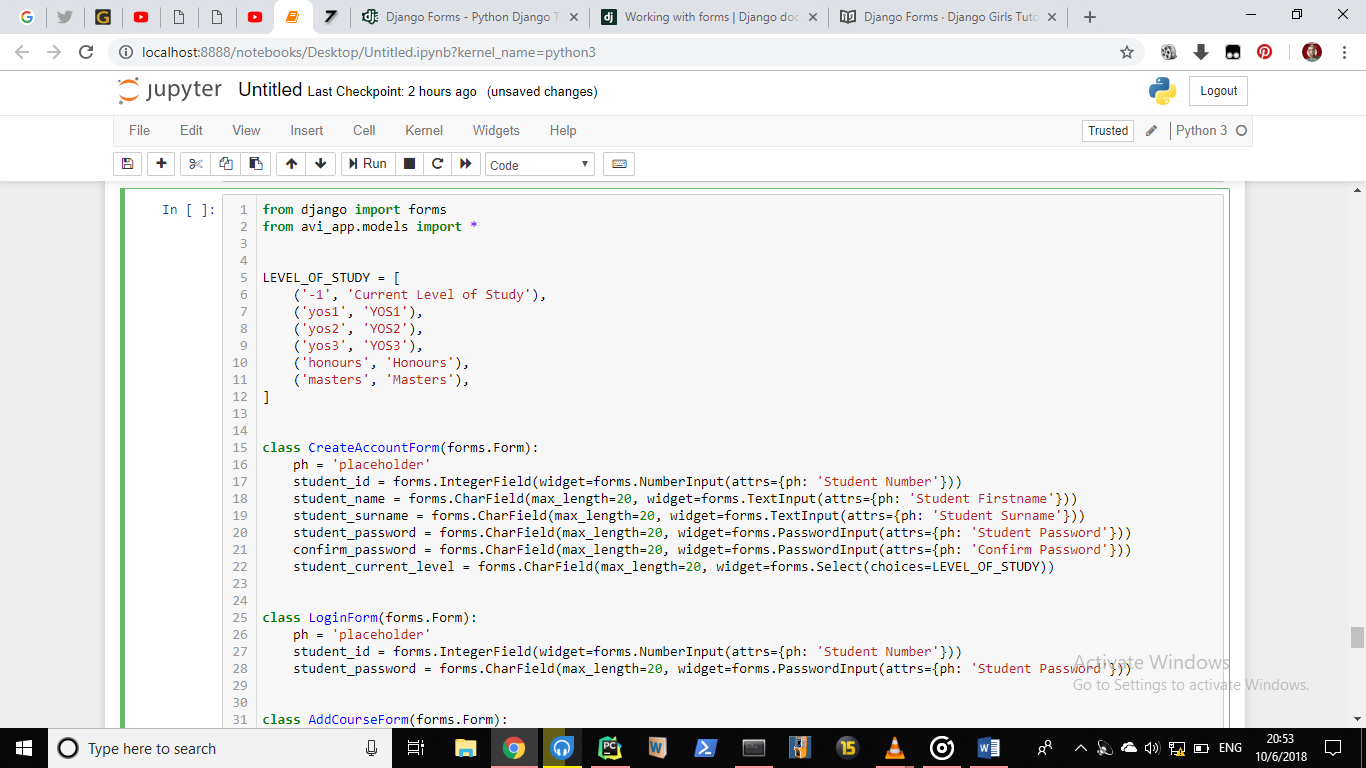
It is *possible* to write code that does all of this manually, but Django can take care of it all for you.

------------------------------------------------------------------------------------------------------------------------------------------

We link the form attributes to the models (database tables) pre-defined under the models python file of the app. This allows user input options and verification of requests inputs.

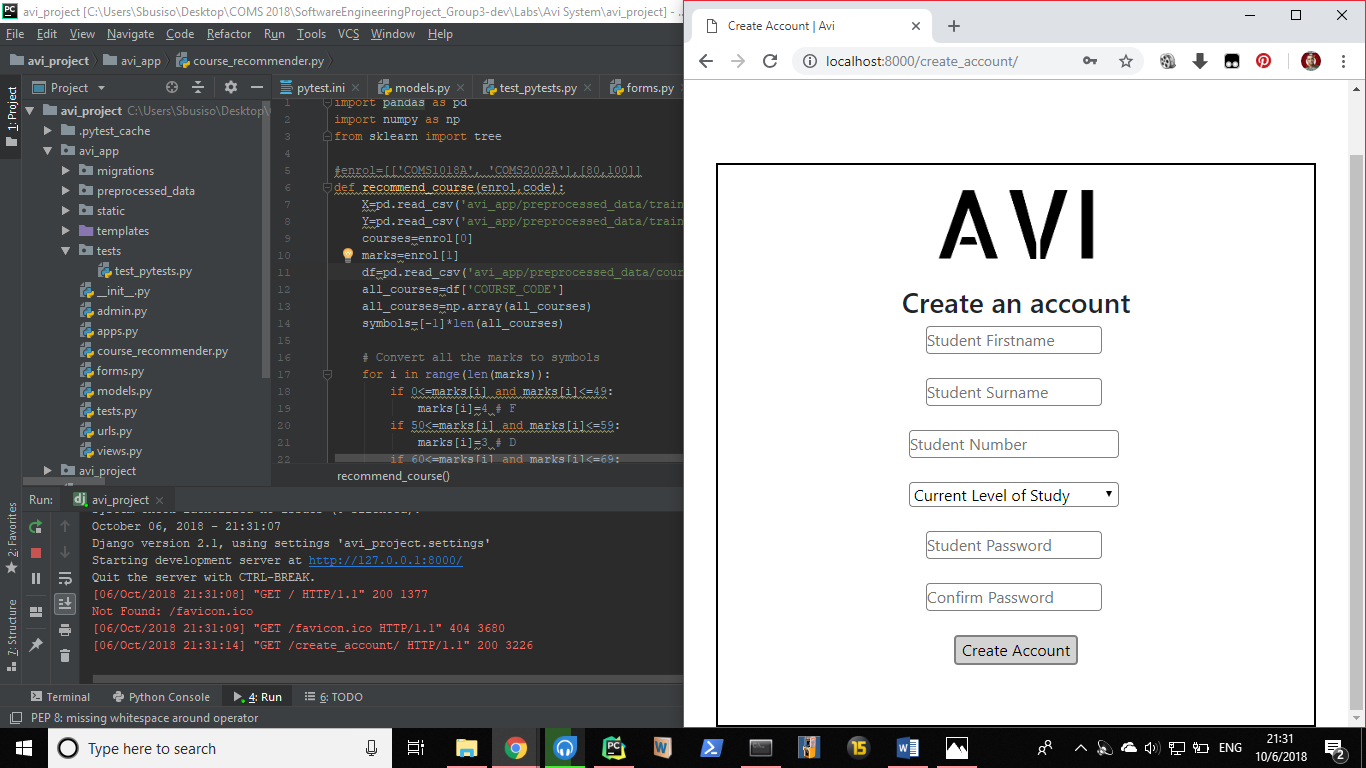
Module Explanation:

The Create User Form: Extracted from “avi\_app/forms.py”



The class ’CreateAccountForm(form.Form)’ instantiates the Student Class attributes therefore enabling them to be used in the form. We specify the type of input we can expect for each attribute which enables us to check whether the user input is valid. This class is used in incorporation with html code and CSS for the createaccount.html file.

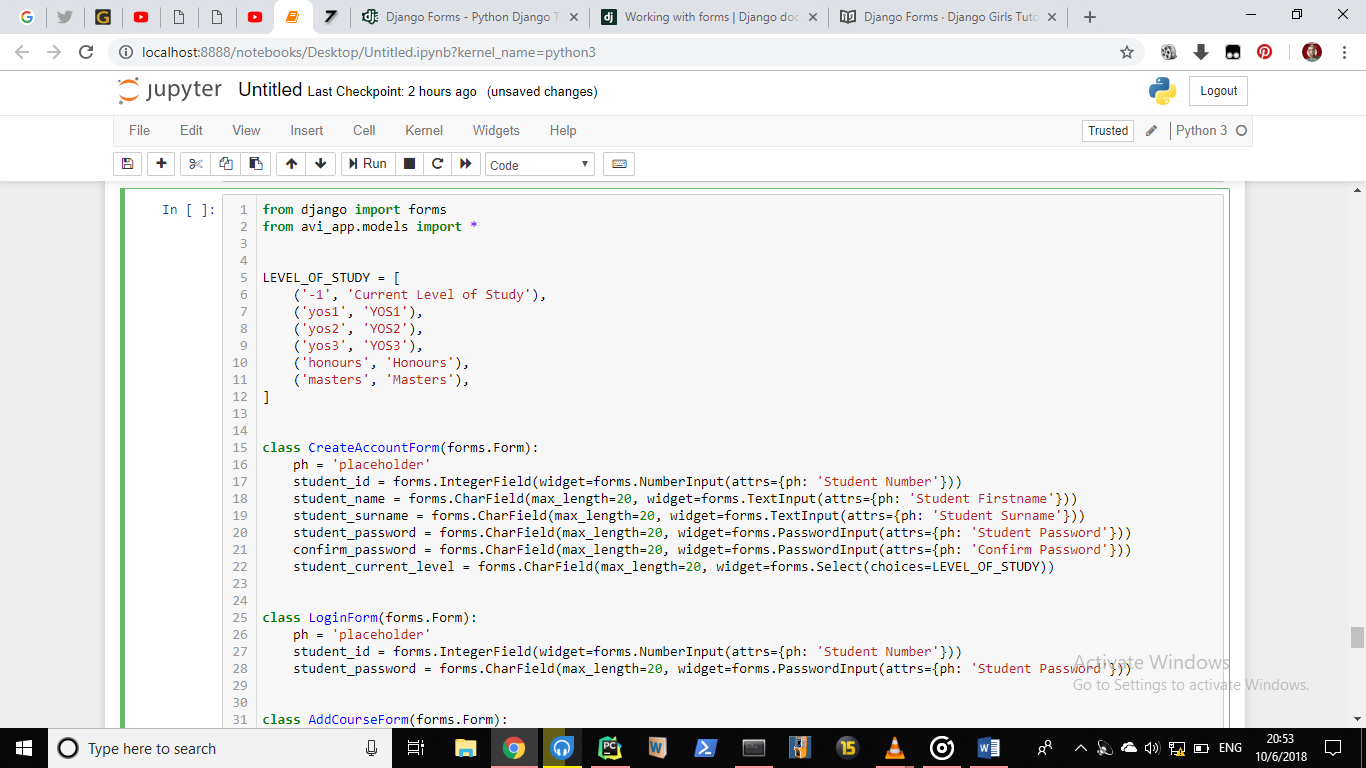
This form allows users to register to use the website app. This is what final form looks like:



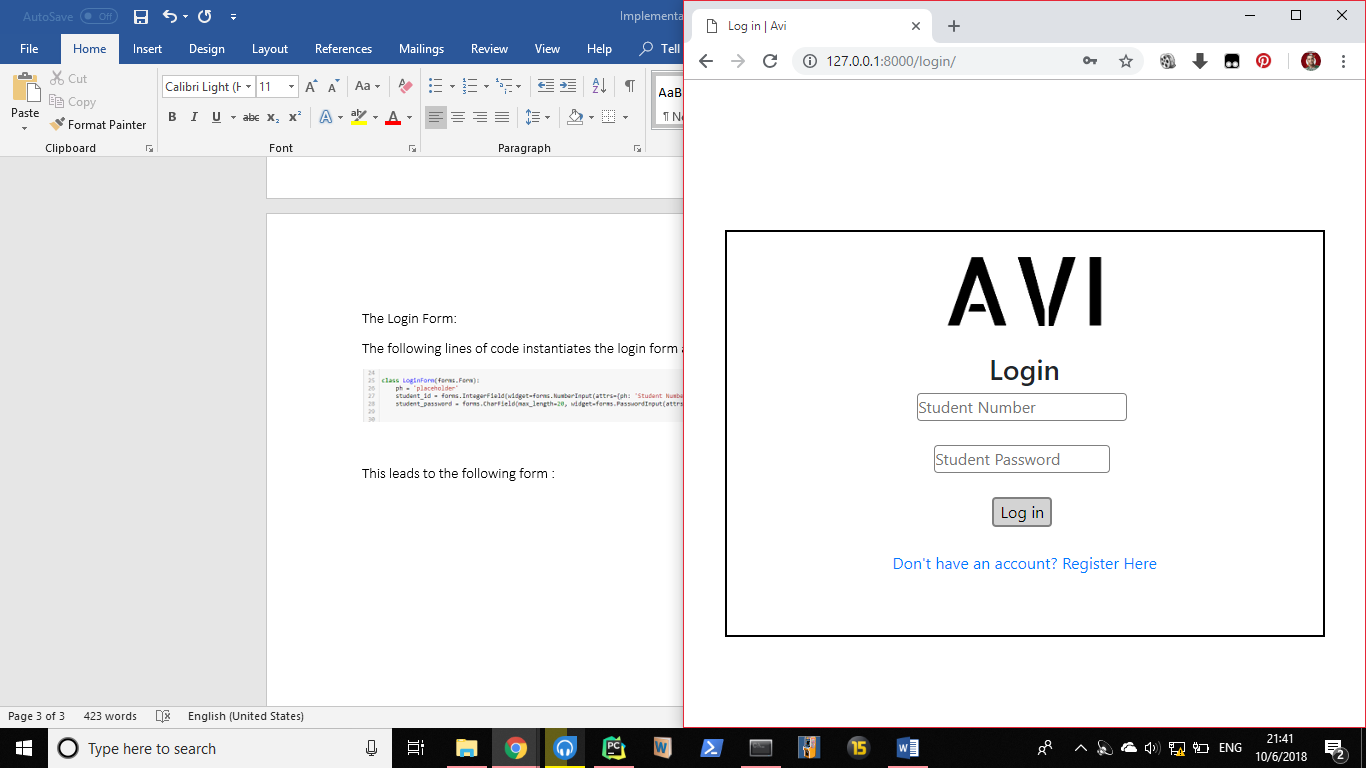
------------------------------------------------------------------------------------------------------------------------------------------

The Login Form : Extracted from “avi\_app/forms.py”

The following lines of code instantiates the login form attributes. This form allows user to be able to login to the website app.



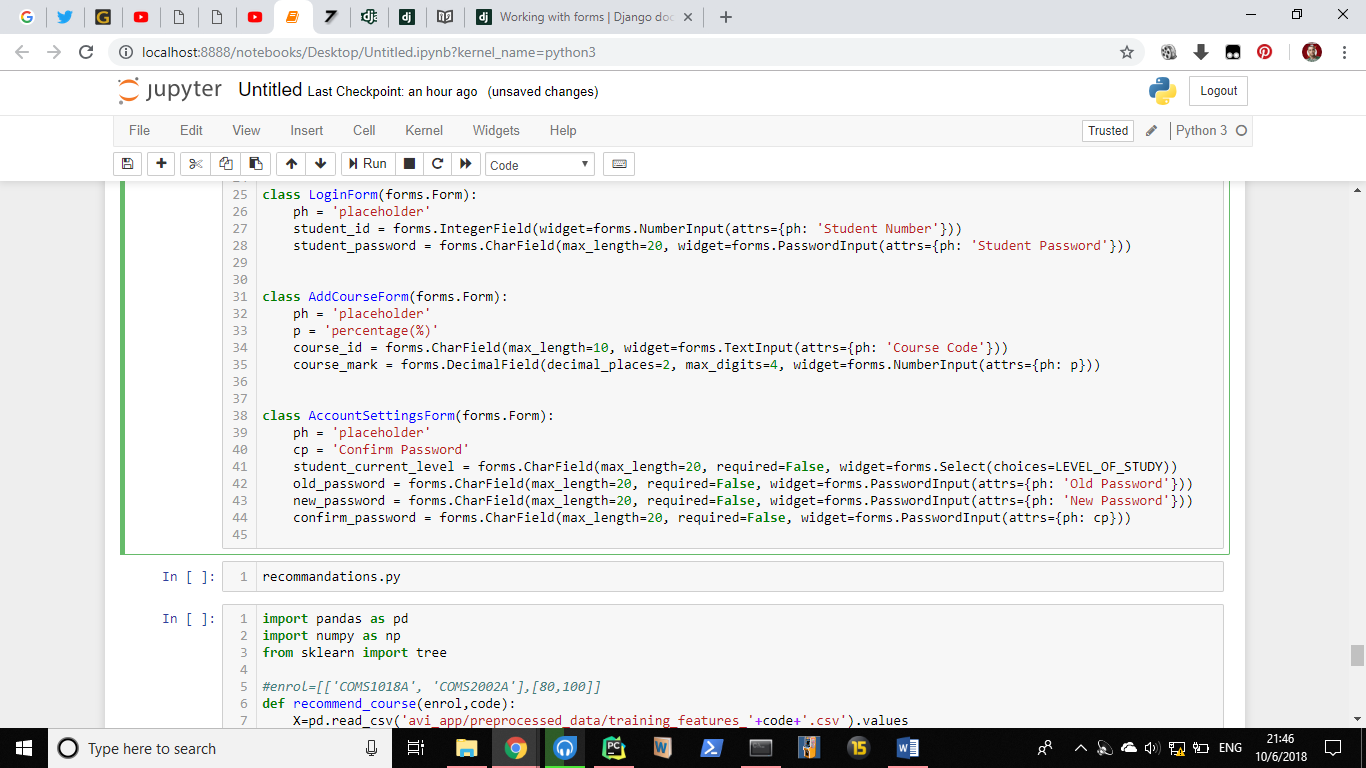
This along with the html and CSS code leads to the following form being created:



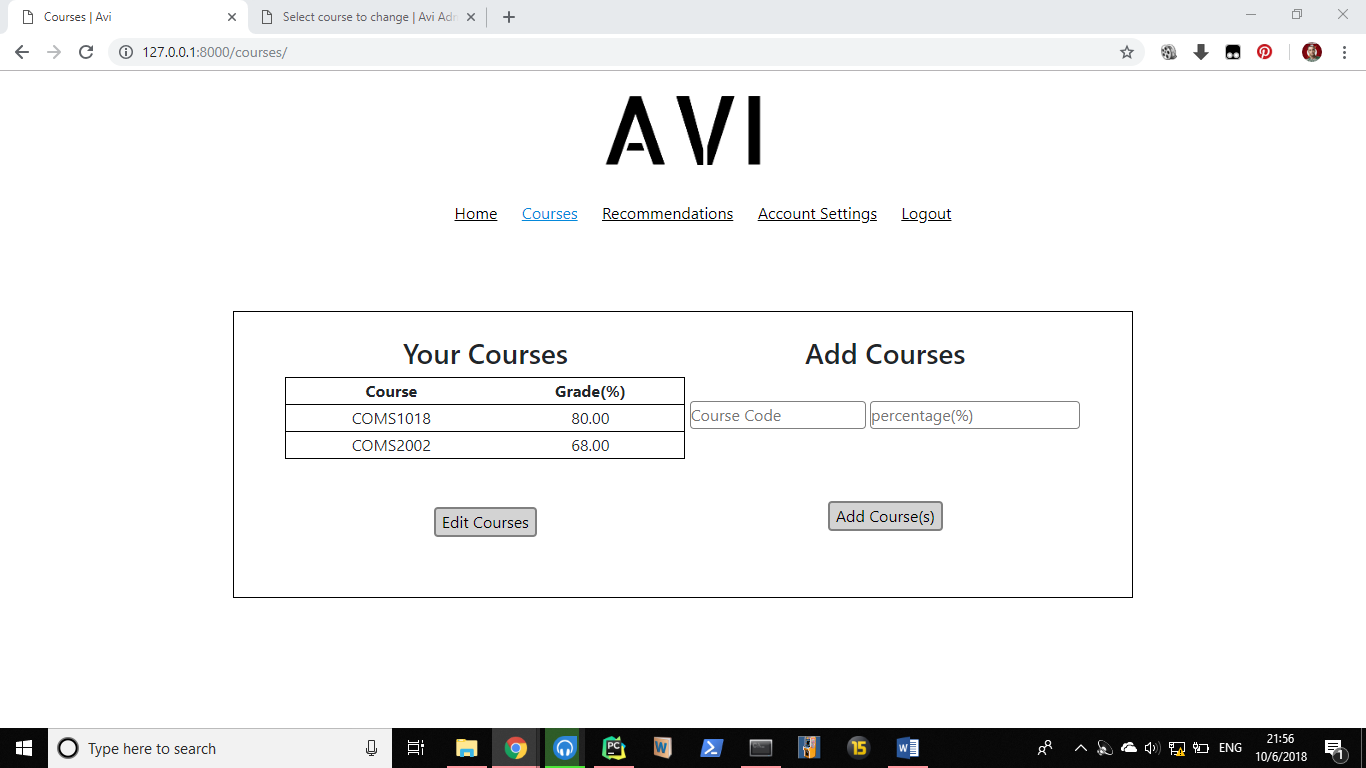
------------------------------------------------------------------------------------------------------------------------------------------

Add Courses Form : Extracted from “avi\_app/forms.py”

This form allows users to be add courses to their profile.

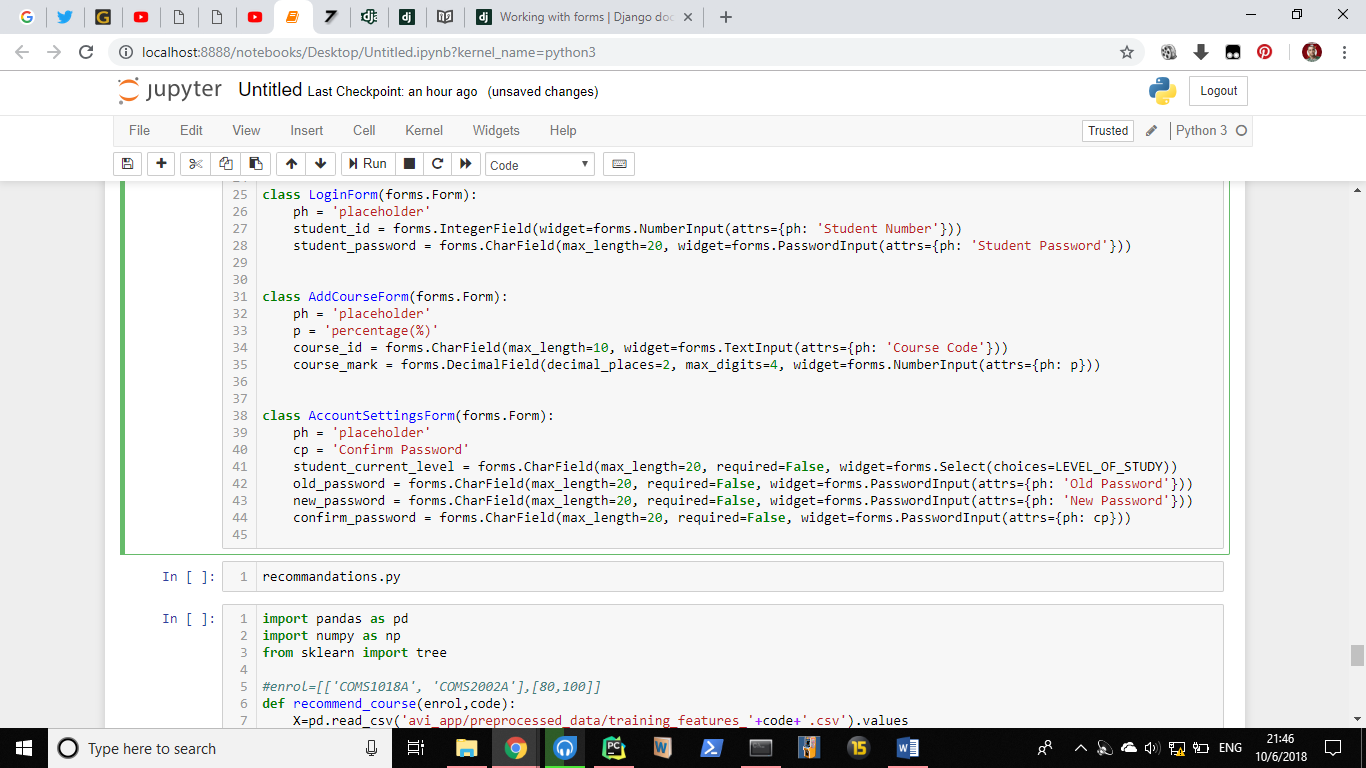


The above code along with the html and CSS code leads to the following form being created:

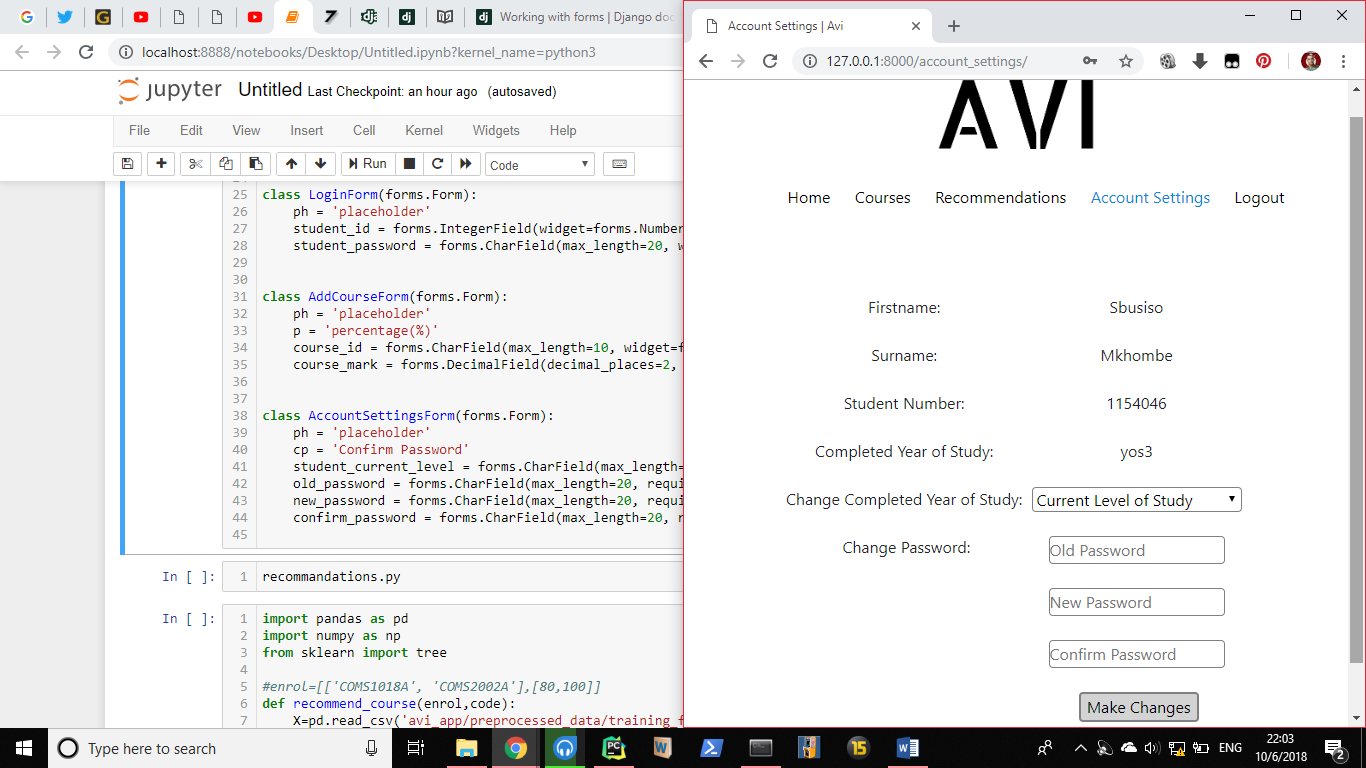


The Account Settings Form: Extracted from “avi\_app/forms.py”

The following lines of code instantiates foreign key attributes. This form allows user to be able to edit their personal details in the website app.



The above code along with the html and CSS code leads to the following form being created:



Models:

Also known as model attributes. A model is the single, definitive source of information about your data. It contains the essential fields and behaviors of the data you’re storing. Generally, each model maps to a single database table.

## Using models[¶](https://docs.djangoproject.com/en/2.1/topics/db/models/#using-models)

Once you have defined your models, you need to tell Django you’re going to use those models. Do this by editing your settings file and changing the [**INSTALLED\_APPS**](https://docs.djangoproject.com/en/2.1/ref/settings/#std:setting-INSTALLED_APPS) setting to add the name of the module/ app that contains your **models.py**.

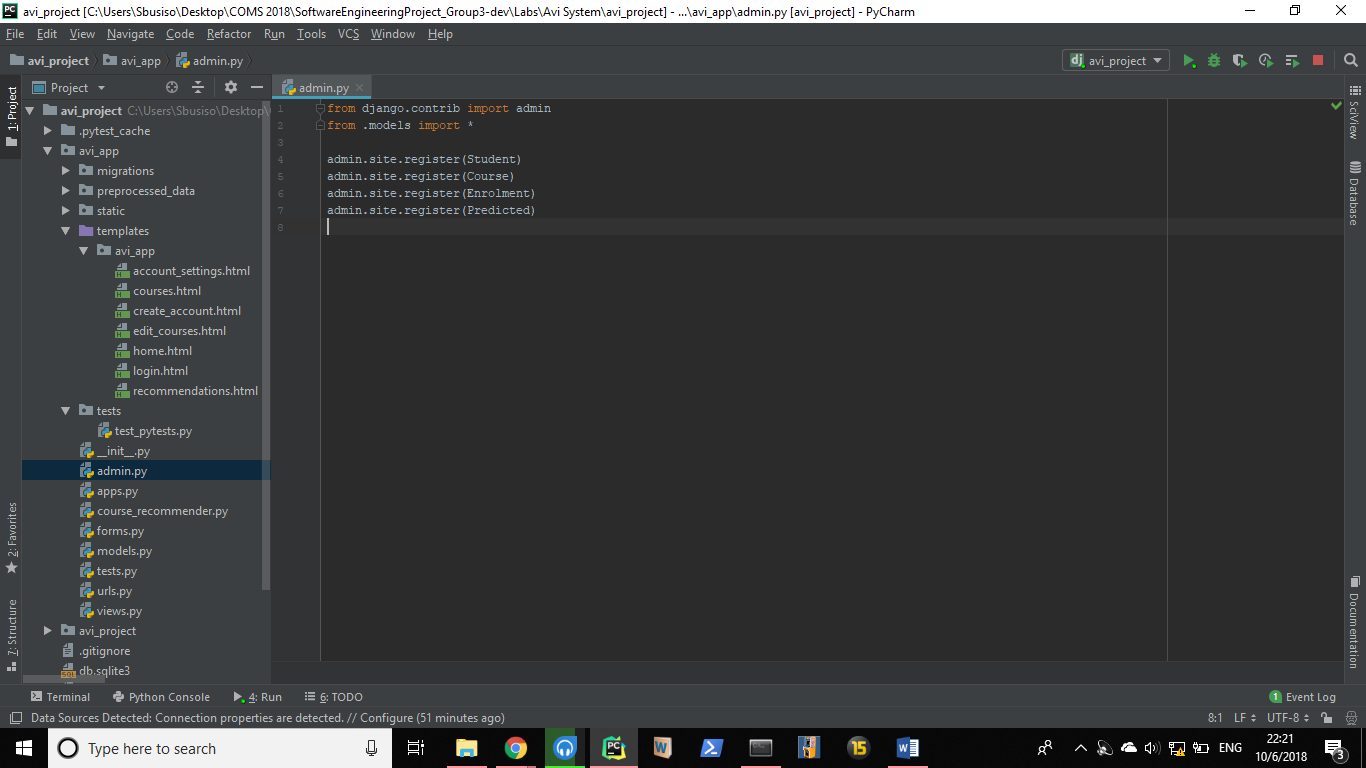
## Fields[¶](https://docs.djangoproject.com/en/2.1/topics/db/models/#fields)

The most important part of a model – and the only required part of a model – is the list of database fields it defines. Fields are specified by class attribute. Examples of fields in our Student Model (Below) would be student\_name, student\_surname and student\_id.

Registering our Models:

Django comes with an admin.py file which allows us to register our model - basically add tables to our database.

We register our tables with the following code:

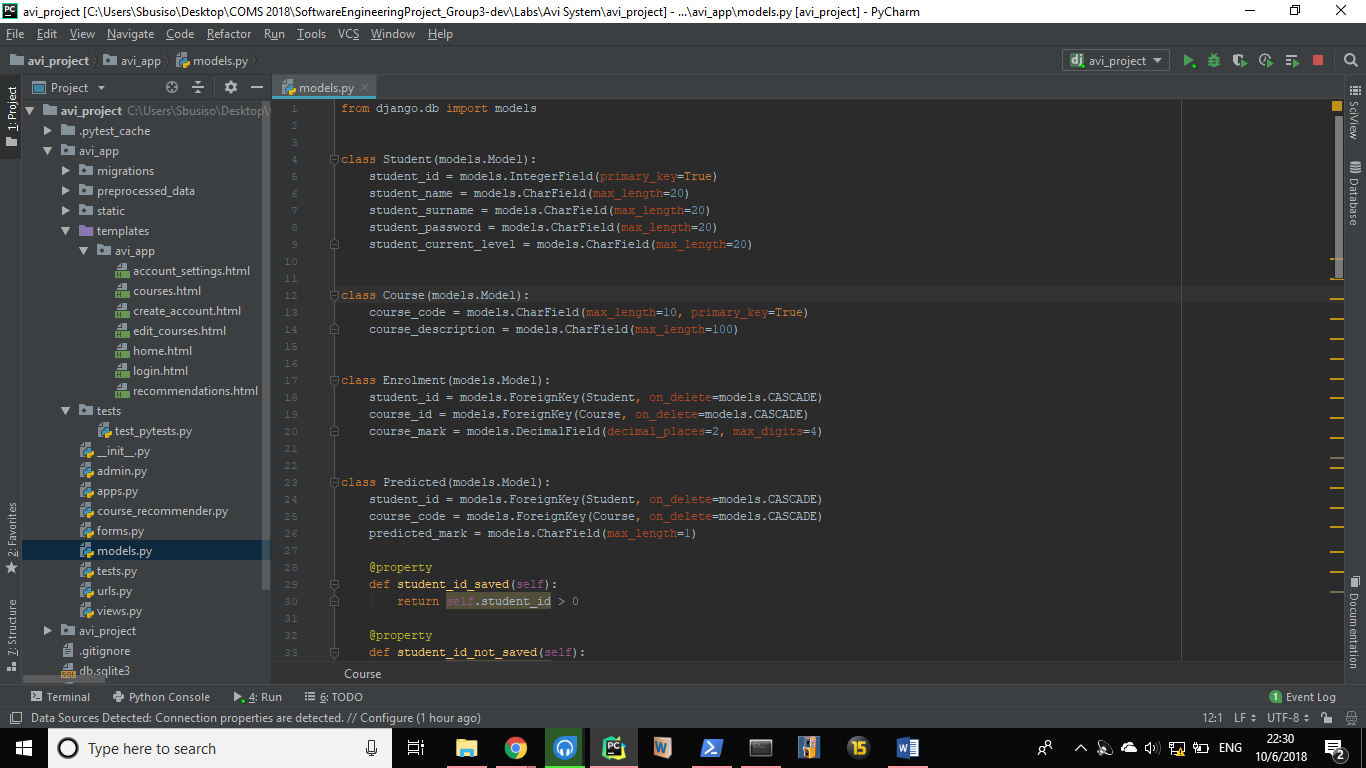


We then use the following commands from our shell to save our database changes:

home\app\_directory:~$ python manage.py makemigrations avi\_app

home\app\_directory:~$ python manage.py migrate

Defining and initializing our models(tables):



The shell migration command leads to these tables being created:

(These can be accessed from <http://127.0.0.1:8000/admin> with the admin as username and “pass1234” as password for admin access)

