A10dance

Basic Idea:

This is what i want m webapplication to do in particular:

1. Admin

\* The admin is the key user here. the process starts by adding an excel file of a particular class with the UID and passwords of the students.

\* This excel file of the class along withthe class names and uid and password of students will be inserted into the database as user.

\*Usingf the excel sheet, it logs into the accounts of students and does the following:

1. Find name and other details of the students and insert to database.

2. goes to the attenednce detrails opf each students of that class, and fetch all the details.

3. the details of each students should be sotred speerately, as well as the common details of that class's attendance should be stored seperately from each of the students details.

4. Thus we will get the total number of hours of each sub ject conducted in that particular semester of that class, and each students will have theirown attendance details (number of hours of subject they have single handedly missed).

5. By using these details, we can easily calculate the attendance [percentage of each students. and disp[lay it ito them.

From the students POV, they can simply login to my website usuingg the uid and password of ther actual RSMS, and can view their attendance percentage, filter according to dates and whatsoever. All these detials are inside the database.

But from the admin pov, it is where the work comes in. The admin is the person who adds the ecel file. and keeps track of the number of classses added to the system.

From thee server POV, the scraping of attendance should be done regularly(once in 24 hours) to regularly update the attendance.

THis is the basic idea of what should be done in my webapplication. there are more things to do but this is the overall idea.

Admin/Opening functionality

In my current default admin page, this is how itr looks:(8 models)

ACADEMIA

Branchs

Courses

ACCOUNTS

Users

ATTENDANCE

Branch hours detailss

Percentage detailss

Student attendances

AUTHENTICATION AND AUTHORIZATION

Groups

STUDENTS

Studentss

In this in the Branch model, when i click the add branch option(default), it shows feild like joining year, passout year, Branch name and division. The admin should add a branch. This is how it works:

1. ALong with these 4 details, the admin should upload an excel file containing the uid and password of the students in a particular class(branch).

2. when submit is clicked, using the uid and password, login to the rsms of each students, and fethc their name, semester, subjects and attendance details.

3. an entry should be created into the branch table, along with the given details and the excel file.

4. All the detials about the courses like course name, code, semster, should be insrted into the coursr table, along with FK reference to branch.

5. the uid, password, name, emial etc should be entered into the customuser table and the student table(one to one wiht customuser) along with a FK reference to branch for all the students.

6. Now for each student, insert their attendance details into the StudentAttendance table one by one.

7. from the combined details attendance of each students of that particular branch, comppute and insert the overall attendance details of the class an insert into the BranchHoursDetails table.

8. from this overall attendance details, calulate the overall number of hors conducted of each course in that semester.

9. From this details, caluculate the attendance percentage of all the students of that classs and insert it into the PercentageDetails table.

All these should be done when the admin adds a new branch using an excel file

RESPONSE:  
To implement this workflow in the Django admin interface when adding a new branch using an Excel file, you'll need to customize the behavior of the admin forms and utilize Django's features for handling file uploads, form processing, and database interactions. Here's a high-level overview of how you can achieve this:

1. **Customize Admin Forms**: Create a custom form for the Branch model in the admin interface. Override the **save()** method of the form to handle the processing of the Excel file and the creation of related objects.
2. **Handle File Upload**: Use Django's file upload handling to allow the admin to upload the Excel file containing the UID and password of students. You can use a **FileField** in the custom form to handle file uploads.
3. **Excel Parsing**: Implement logic to parse the uploaded Excel file and extract the UID and password of students. You can use libraries like **openpyxl** or **pandas** to read Excel files and extract data.
4. **RSMS Integration**: Use the extracted UID and password to log in to the RSMS of each student and fetch their details, including name, semester, subjects, and attendance details. You'll need to implement web scraping or use APIs provided by the RSMS if available.
5. **Database Operations**: Inside the **save()** method of the custom form, perform database operations to create entries in the Branch, CustomUser, Student, and Course models based on the extracted data from the Excel file and RSMS.
6. **Attendance Calculation**: Calculate the attendance details for each student and the overall attendance details for the branch. Insert these details into the StudentAttendance and BranchHoursDetails models respectively.
7. **Course Details**: Insert the course details extracted from the RSMS into the Course model, associating them with the branch.
8. **Percentage Calculation**: Calculate the attendance percentage for each student based on the attendance details and insert them into the PercentageDetails model.
9. **Feedback and Error Handling**: Provide feedback to the admin about the success or failure of the operation. Handle errors gracefully and provide informative error messages if any step fails.
10. **Testing and Validation**: Thoroughly test the custom form and workflow to ensure that it works as expected. Validate the input data and handle edge cases appropriately.

By following these steps, you can implement the desired workflow in the Django admin interface for adding a new branch using an Excel file while automating the process of fetching student details and calculating attendance statistics.