Week 4

Assessment Questions:

1. **Explain the role of DocTypes in ERPNext.**
    - In ERPNext DocTypes are the most basic building block for any application.
    - They describe how the data are stored in the database.
    - They form data model which means how the data is structured in database and view which means how data is represented to the user.
    - It stores meta-data in tables which means it not only stores data but also how each data behaves in context of each other and makes it easy to changes to those meta-data on fly without writing much code.
    - DocType generally represents an Entity in context of database i.e if we wanted to have an entity named Student then we create an DocType called Student.
    - It enables ORM(Object Relational Model) which means it makes it easy to read, write and update the data without writing SQL query

2. What are the key features of a DocType? How does it manage data?

- DocType is the basic building block of the ERPNext application which stores the data and its meta-data.
- Following are the Key Features of the DocType:
  - **Data Structure Definition:-** It defines the how data is structured in the database and it also stores meta-data about different data fields and their behaviour i.e. how each fields will behave with respect to each other.
  - **User Interface Design:**- It also stores the user interface details such as labels, form layouts, placeholders, readability etc. it is also describe how data will be visible to user.
  - **Permissions:**- It also provides provisions of providing permissions for particular DocTypes and fields to be restricted to certain users.
  - **Various Types of DocTypes:**- Support of Various types DocTypes for different occasions **e.g.** an virtual type DocType does not create table entries for its fields as data in fields are fetched from external source such as external files or API.
  - **Support For different Data types and Fields:** It supports different fields that supports different data types that we may required to store and perform tasks upon.
- DocType provides the structured way to store data. When we create a new document based on DocType it data are automatically mapped into corresponding

database table. The fields refers to the corresponding column of database tables.

- It provides data consistency through the use of concept of foreign key.

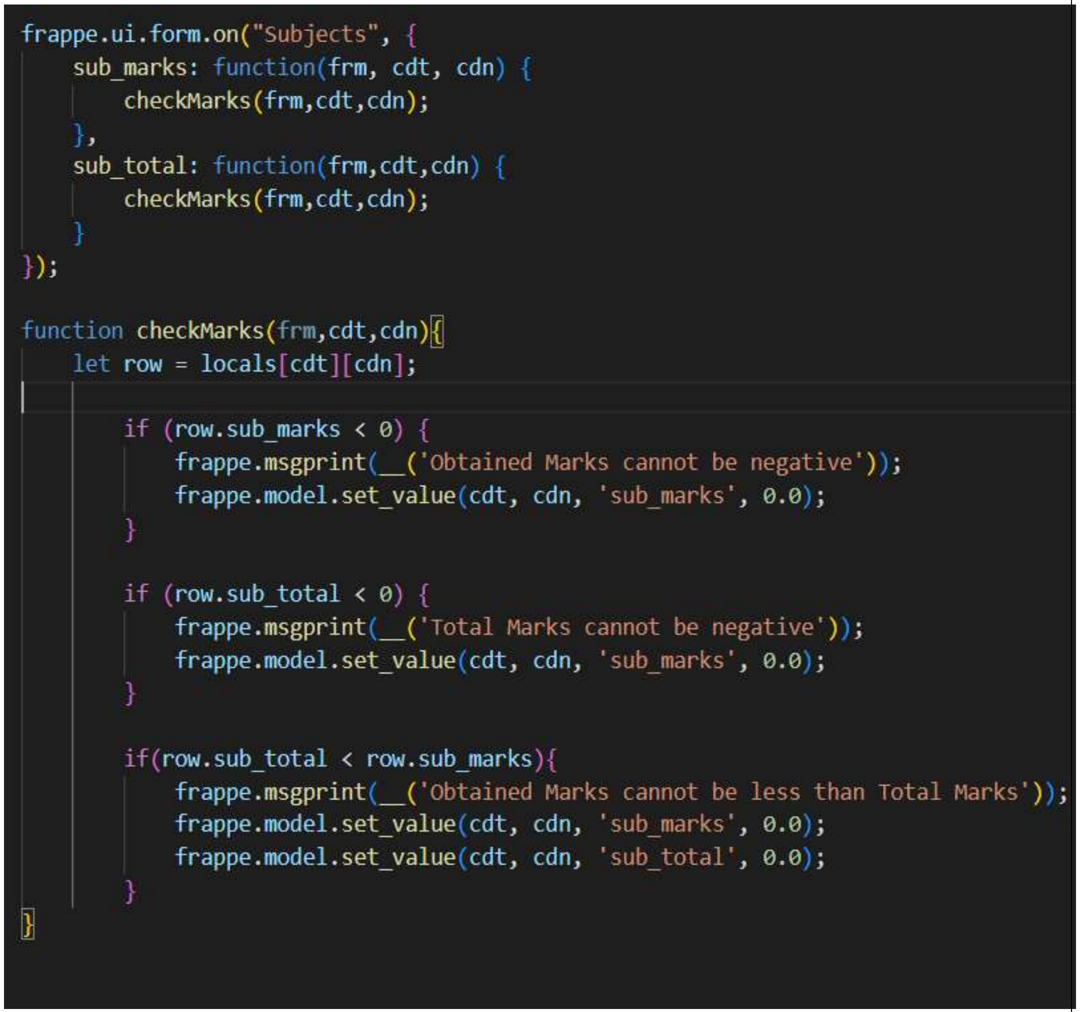
3. Explain how you structured the DocType and the purpose of each field.

- To structure the DocType we use the features of fields that are present in the Doctype.
- We have support of different types fields in DocType which are used for specific types.
- Using this field types in the DocType will structure it.
- Lets Take an example of Academics Doctype in which we wanted to take different subjects name and the marks obtained.
- So we choose two field lets say with Subject Name and Marks Obtained label. Now we will make the type of subject as link so that we can entered the predefined subject name from another doctype called subject and make Marks Obtained field as Float as it may takes marks in floating point representation. And can add Constraints to the fields as if Subject name is entered we must add Marks Obtained by using Mandatory depends on field in Marks obtained field.
- This was just one small example we can do mush more with playing the field properties.

4. Describe the difference between Client Script and Server Script.

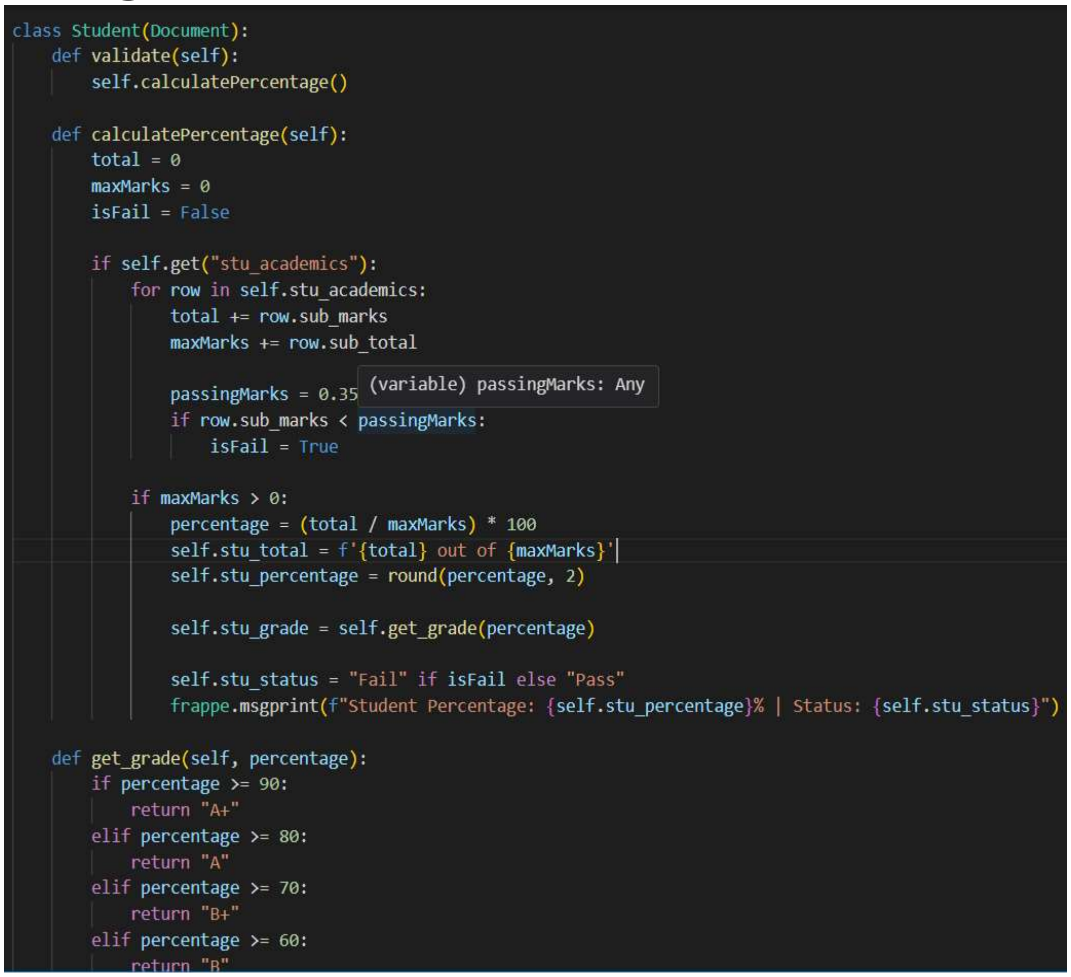
- Both this scripts are used to extend the functionality of the DocTypes but differs significantly in their execution:
  - **Client Script:-** Its primary use is to providing immediate support to user interface and immediate feedback. With this real-time validations are possible as they support field events for each fields in DocType. It does the task on client side rather than server side so it decreases the load on server and are also fast as it is done locally. It is generally used to improve user experience. It also supports database events. It implements those database events using the concepts of asynchronous programming in Java Script.
  - **Server Script:-** It will run the scripts on Frappe server. Generally used to provide core business logic to our application. Complex calculations, Database operations, Data Validation etc are the work done by server scripts. It provides security and data integrity. It leverages servers resources for complex operations to provide optimal performance.

5. Check their Client Script to validate the marks entered in the child table.



```
frappe.ui.form.on("Subjects", {
    sub_marks: function(frm, cdt, cdn) {
        checkMarks(frm,cdt,cdn);
    },
    sub_total: function(frm,cdt,cdn) {
        checkMarks(frm,cdt,cdn);
    }
});

function checkMarks(frm,cdt,cdn){
    let row = locals[cdt][cdn];

        if (row.sub_marks < 0) {
            frappe.msgprint(__('Obtained Marks cannot be negative'));
            frappe.model.set_value(cdt, cdn, 'sub_marks', 0.0);
        }

        if (row.sub_total < 0) {
            frappe.msgprint(__('Total Marks cannot be negative'));
            frappe.model.set_value(cdt, cdn, 'sub_marks', 0.0);
        }

        if(row.sub_total < row.sub_marks){
            frappe.msgprint(__('Obtained Marks cannot be less than Total Marks'));
            frappe.model.set_value(cdt, cdn, 'sub_marks', 0.0);
            frappe.model.set_value(cdt, cdn, 'sub_total', 0.0);
        }
}
```

6. Check the Server Script that calculates the percentage of marks.



```python
class Student(Document):
    def validate(self):
        self.calculatePercentage()

    def calculatePercentage(self):
        total = 0
        maxMarks = 0
        isFail = False

        if self.get("stu_academics"):
            for row in self.stu_academics:
                total += row.sub_marks
                maxMarks += row.sub_total

                passingMarks = 0.35
                if row.sub_marks < passingMarks:
                    isFail = True

            if maxMarks > 0:
                percentage = (total / maxMarks) * 100
                self.stu_total = f'{total} out of {maxMarks}'
                self.stu_percentage = round(percentage, 2)

                self.stu_grade = self.get_grade(percentage)

                self.stu_status = "Fail" if isFail else "Pass"
                frappe.msgprint(f"Student Percentage: {self.stu_percentage}% | Status: {self.stu_status}")

    def get_grade(self, percentage):
        if percentage >= 90:
            return "A+"
        elif percentage >= 80:
            return "A"
        elif percentage >= 70:
            return "B+"
        elif percentage >= 60:
            return "B"
```

7. Explain how to trigger a DocType Event in ERPNext.

- In ERPNext and Frappe, you can trigger DocType events using server-side Python scripts or client-side JavaScript scripts. These events allow you to execute custom logic when a record is created, updated, submitted, or deleted.
- Frappe provides hooks to trigger events in hooks.py or directly within a app

- Common Server Side DocType Events:-
    - before_insert
    - after_insert
    - before_save
    - after_save
    - before_submit
    - on_submit
    - before_cancel
    - on_cancel
    - on_trash
- Common Client Side DocType Events:-
    - Refresh
    - Validate
    - Onload
    - before_save
    - after_save
    - before_submit
    - on_submit
    - before_cancel

8. What is the purpose of Print Formats in ERPNext?

- Print Formats in ERPNext allow users to generate customized, professional-looking print documents for invoices, purchase orders, delivery notes, quotations, and other DocTypes.
- They help in branding, compliance, and information clarity when sharing documents with customers, suppliers, or internal teams.

- There are different types of print formats as follows:
    - **Default:** This is an print format that is available by default provided by Frappe.
    - **Custom Print Format using Builder:** By using this we can create custom print format using drag and drop facility.
    - **Custom Print Format using Jinja template:** Using this we can create custom attractive print format using html, css and jinja.

9. How many field types are available in a Doctype? Provide details and explain the purpose of any five of them.

- ERPNext provides more than 35 field types to store and manage different types of data.
- This fields describe how the data is stored, displayed and validated in system.
- Following is the explanations of five field types:
    - **Link Field:** Creates a relationship between two DocTypes, acting as a Foreign Key.
    - **Table Field:** Creates a child table inside a DocType to store multiple records within a single document. Stores structured data, like invoice items with quantity and price.
    - **Currency Field:** Stores monetary values with proper currency formatting.

- **Attach Image Field:** Allows users to upload and store images in a DocType. Stores users profile pictures.
- **Check Field:** Stores a Boolean (Yes/No) value with a checkbox.

10. You have a Link field in your document that references another DocType, which contains an address field. How would you display the value of the address_line1 field from the linked DocType in your print format?

- We can use following jinja code to access address_line1 from address link
  - {% set address_doc = frappe.get_doc("Address", doc.address) %}
  - {{ address_doc.address_line1 }}