- *Think about how to classify each of the types of content on your site. Articles, comments on articles,news items, questions and answers, documents to be approved or rejected with comments, and so on. Each type of content will require a different flavor of data model and moderation/approval/versioning process.*

  Our site's main contents are

1. Meeting:
    a. Moderation: A user need to login to create a meeting. Also only the user that created the meeting can edit or delete it.
    b. Approval: No approval required
    c. Versioning: No versioning required

2. Schedule:
    a. Moderation: Only the user that created a meeting can provide the schedule for that meeting i.e., time, place and date.
    b. Approval: No approval required
    c. Versioning: No versioning required

3. Comment:
    a. Moderation:After successful creating and scheduling a meeting, that user invites other users. These users can only comment on that meeting.
    b. Approval: No approval required
    c. Versioning: No versioning required

- *Design your content data model and discuss the implications of your design choices with potential users of the system. In general, there should be at least one category of user-submitted content with authoring workflow specific to your project and at least one question and answer forum. However, the specifics will be very dependent on your project's requirements. Document the design and your discussions on your site.*

  User submitted content: They are 3 categories of user-submitted contents i.e., meeting, scheduling and comment.

❖ *Specify the workflow for each kind of content on your site and discuss the implications of your design choices with potential users of the system. Document the workflow design and your discussions on your site.*

Workflow

In our workflow we have 3 contents types.

- Meeting: A meeting is created by a user which can be either a faculty or a student. A meeting includes schedule and a comment option.
- Scheduling: Every meeting includes a schedule like the time, place and date. A user has to login to their profile in order to create, edit or delete a schedule. Moreover that schedule can only be edited or deleted by that particular user.
- Comment: Every meeting includes a comment option. When a user creates a meeting and then invites other users to inform them of that meeting, then those users can only comment. They can also edit or delete the comments.

A student/faculty creates a profile and then logs in using that identification. Thereafter then can create a meeting and invite other people. A user logged into their profile can also see the various meetings that they have created as well as reschedule or delete that meeting altogether. After everything is completed they can logout.

❖ *Design your versioning system. What is the versioning system for each type of content? Do you need a complete history of every version, or just a current version with the author of the last version? Make sure you think carefully about the needs of the user community. Document the versioning design on your site.*

- Meeting: Once a meeting is created, it will be referred by a unique id. Only the current changes are kept track of.
- Scheduling: When an old/existing schedule is edited , only the new/updated schedule is kept track of, instead of the old one.
- Comment: The timestamp of the comment will be the versioning system.

Blending versioned and non-versioned parts of the application

There are parts that we do not want to version. At the user level, we have user-managed data, through a Content Management System (texts). Such parts should be separate from a versioning system as the data will be of very large volume. Versioning is problematic at the very least, and in most cases is simply not possible.

We resolve the situation by placing configuration files and data folders in a list of ignored parts of the project. In Git, such a setting has the file content marked .gitignore, where specific files can be specified and rules can be set for ignoring folders parametrically.

❖ *Write UATs for a \skeletal" implementation of the workflow for at least one type of content on your site then get the UATs to pass. Users should be able to get something up on your server and see it published.*

Feature: Meeting
In order to organize a meeting, I want to create a schedule for the meeting

Scenario: Create a schedule
A user should be able to create a schedule

Given I am logged in
And I am in the meetings page
Then I should visit the create meeting page
Then I should see a form for meeting to fill
When I submit the create meeting form
Then I should see the details of my meeting

```
Given (/^I am logged in$/) do
  @user = FactoryBot.create :user
end

And(/^I am in the meetings page$/) do
  visit '/meetings'
end

Then(/^I should visit the create meeting page$/) do
  visit '/meetings/new'
  @meeting = FactoryBot.create :meeting
end

Then("I should see a form for meeting to fill") do
  expect(page).to have_selector('form')
end
```

```
When("I submit the create meeting form") do
   fill_in 'Title', with: @meeting.title
   fill_in 'Note', with: @meeting.note
   fill_in 'Location', with: @meeting.location
end

Then("I should see the details of my meeting") do
  visit '/meetings'
end
```