# FILESTAGE
# [Technical Challenges]

## 1.Due dates

- ❖ Requirement :

  A feature that specifies that todo/task will be completed on a particular date.
  This feature reminds the user that this are the task that need to completed on today or till the date he has specified.

  This will help us to get(filter) todos by today's date.

- ❖ Approach :
  - ➢ **Frontend**
    - ■ We will add one **date picker field(calendar)** so we can add the due date on a todo.
    - ■ Users can select the due date at the time of creating todo.
    - ■ We will show the date from today to future dates and in the calendar and the previous dates will be disabled.
  - ➢ **Backend:**
    - ■ We will add one more field in todos document with name due_date.
  - ➢ **Security:**
    - ■ We will add the checks or conditions in add todo api that user can't be able to add previous date in due date if he will do, api responds that ` please select valid date`
    - ■ For frontend and backend secure connection there is one option that we can use. We have to use one static secured generated key in backend and frontend We have to pass key from frontend to backend and have to verify it in backend
    - ■ We can modify our cors in backend for validate that particular domain can only will have access for this backend apis

- ❖ Estimation: 1.5 - 2 hours

## 2. Supporting a large number of tasks

- ❖ Requirement :

  To reduce load time, by dividing a large amount of data retrieval from the backend server into smaller chunks.
  In this feature we need to implement **pagination** in which we can get a specific amount of data from the server.

❖ Approach

➢ **Frontend:**

- For Implementation of pagination we will use `InfinitScroll` module on frontend app.
- Which will fire/hit the backend server for next page data/record, automatically upon reaching to the last element (Vertically) on DOM or browser screen.
- This module will help to provide an instance to fire the backend request and we can get a specific amount of data from the server.
- In backend we have to add the limit and skip property on get todos api controller.

  **Limit:** This property specifies how much data we want to get.
  **Skip:** This property specifies how much data we have to skip.

❖ Estimation: 2 - 3 hours

## 3. Reordering tasks

❖ Requirement :

Users will be able to re-order the todos with the help of drag and drop to any specific index.

❖ Approach :

➢ **Frontend:**

- For managing re-ordering the todos list we will use the react-dnd module
- In this module we will add the functionality that the user can drag the todo and drop the todo to shuffle the order.

❖ Estimation: 2 hours

---

## ❖ Refactoring and changes:

### ➢ Frontend :

- Improve file structure.
- Created separate folder for our components
- Create utils.js file for managing our api calls so we can use our api to make it easily accessible to any module/component.
- Make separate styles.js file for our components style

### ➢ Backend:

- Create controller.js for managing our api logics.
- Create logs.js fire for our logging configuration.

### ➢ Security:

- I have used the wiston module for implementing logging functionality .
- I have tested **XSS attack** cases. We have avoided directly rendering html strings.
- I have handled the exceptions on api calls and handled in the logger as well.

### ➢ Bug:

- Completed field is not updating in the backend although the api is working fine. **(**Their is change in backend where we are updating todos their was wrong collection name is used we have fixed with right one**)**