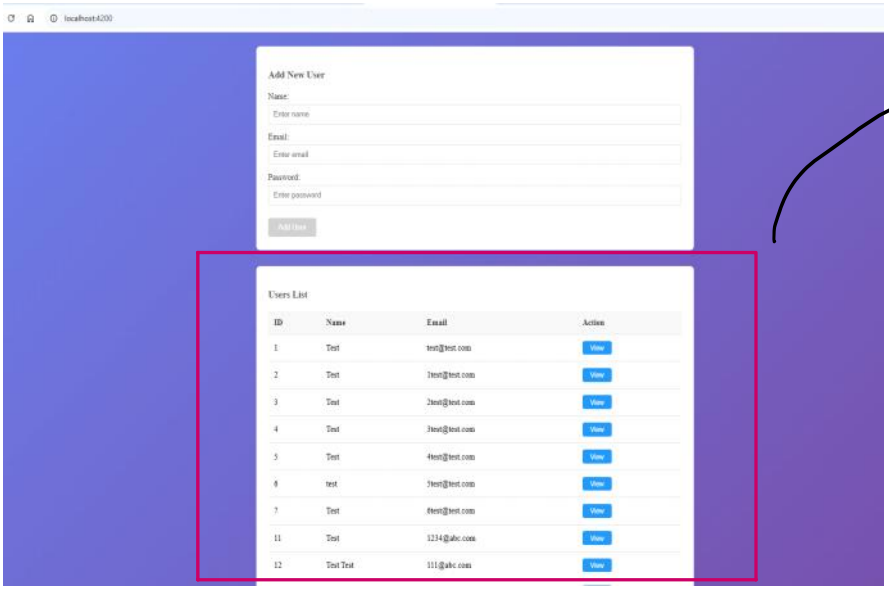
Project_Day_2

29 January 2026 00:50

On Second day of project we will see how two features can be built.

1st feature is Pagination : Here the scenario is list of users which is created is getting large and there is limit to show these users over UI, as you can't populate all the users over a same page.



As you can see loading 100k users can cause issues here.
To overcome it Pagination will be used.

Implementation brief

1. Pageable interface will be used to get the size you require and you can set up page, size, sorting and sort by as per your requirement.

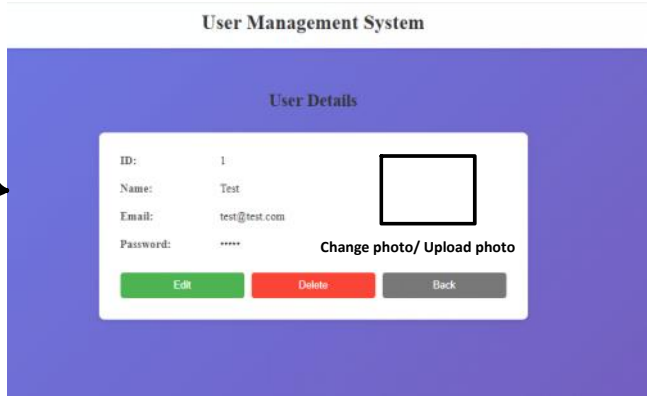
```
@GetMapping
public ResponseEntity<Page<User>>
getAllUsers(@RequestParam(defaultValue = "0") int page,
                                @RequestParam(defaultValue = "3") int size,
                                @RequestParam(defaultValue = "id") String
sortBy,
                                @RequestParam(defaultValue = "asc") String
sortDir) {

    Sort sort = sortDir.equalsIgnoreCase("desc")
            ? Sort.by(sortBy).descending()
            : Sort.by(sortBy).ascending();

    Pageable pageable = PageRequest.of(page, size, sort);
    return ResponseEntity.ok(userService.getAllUsers(pageable));
}
```

2nd feature

Here in this feature it is asked to upload the photo of user and save it. There are multiple ways to do it and you can save that in database also.
Solution designed here will be using AWS S3 bucket to get the photos loaded there. As of now the page looks like below.

Current

Expected



Implementation brief

1. Create S3 bucket in AWS and give it appropriate permission policy.
2. Add dependency in pom.xml and application.properties
3. Change frontend and backend code.

Both these features are implemented for backend and frontend and in git you can visit both below branches in git in **photoupload** branch.
Backend : https://github.com/neerajbalodi/user-management-backend.git
Frontend : https://github.com/neerajbalodi/user-management-frontend.git