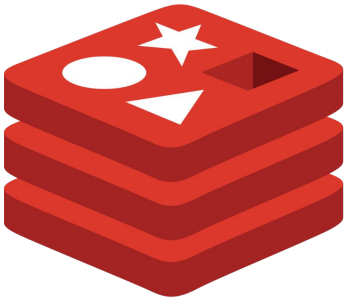
Mini-Presentations - Round 1

Team 7 - Bennett, Dima, Francisco, Ishika, Paula

Project Specifics

- Customized bubble sheets
- Job queuing system for bubble sheet
- Multiple methods of authentication



redis

Framework Choice

Frontend

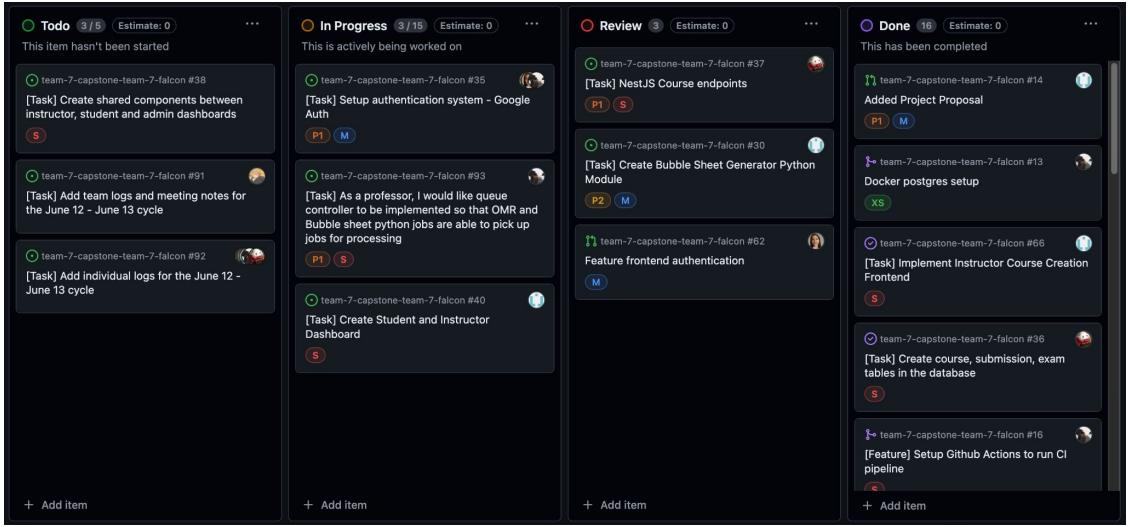
NEXT.JS

Backend

Nest JS

Teamwork strategy

1. Team meeting for cycle planning
2. Update GitHub project board
3. Members select tasks
4. SME assigned to their framework



Testing

Automated Tests → Manual Tests → Pull Request → Reviews

- Integration Tests
- Unit Tests

Testing

Code Blame 39 lines (33 loc) · 863 Bytes

```yaml
name: CI - Tests (Backend)

on:
  pull_request:
    paths:
      - 'backend/**'

jobs:
  run-tests:
    name: Run tests (Backend)
    runs-on: ubuntu-latest
    services:
      postgres:
        image: postgres:latest
        env:
          POSTGRES_DB: test
          POSTGRES_USER: postgres
          POSTGRES_HOST_AUTH_METHOD: trust
          POSTGRES_PASSWORD:
        options: --health-cmd pg_isready --health-interval 10s --health-timeout 5s --health-retries 5
        ports:
          - 5432:5432

    steps:
      - name: Checkout code
        uses: actions/checkout@v4

      - name: Set up NodeJS
        uses: actions/setup-node@v4
        with:
          node-version: '20'

      - name: Install packages
        working-directory: ./backend
        run: npm ci

      - name: Run tests
        working-directory: ./backend
        run: npm run test
```

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	56.6	41.75	57.42	54.59	
src	0	100	0	0	
app.module.ts	0	100	100	0	1-26
main.ts	0	100	0	0	1-32
src/common	68.96	50	80	68.96	
errors.ts	94.73	100	88.88	94.73	17
helpers.ts	11.11	0	0	11.11	10-19
index.ts	100	100	100	100	
src/decorators	77.77	100	50	75	
roles.decorator.ts	100	100	100	100	
user.decorator.ts	60	100	0	60	12-14
src/enums	100	100	100	100	
auth.enum.ts	100	100	100	100	
user.enum.ts	100	100	100	100	
src/guards	20	0	0	14.81	
auth.guard.ts	33.33	0	0	25	17-49
course-role.guard.ts	0	0	0	0	1-77
system-role.guard.ts	30	0	0	22.22	19-70
src/modules/auth	75.43	28.57	28.57	72.54	
auth.controller.ts	46.15	0	0	41.66	22-25,40,62-77,92-103
auth.module.ts	100	100	100	100	
auth.service.ts	100	100	100	100	
src/modules/auth/pipes	60	0	0	45.45	
auth.pipe.ts	62.5	0	0	50	12-18
code-auth.pipe.ts	57.14	0	0	40	11-14
src/modules/course	38.59	33.33	57.14	39	
course.controller.ts	0	0	0	0	1-87
course.module.ts	0	100	100	0	1-10
course.service.ts	100	100	100	100	
src/modules/course/dto	0	100	100	0	
course-enroll.dto.ts	0	100	100	0	1-7
src/modules/course/entities	100	100	100	100	
course-user.entity.ts	100	100	100	100	
course.entity.ts	100	100	100	100	
src/modules/datasource	0	0	0	0	
ormconfig.ts	0	0	100	0	1-37
typeorm.module.ts	0	100	0	0	1-19
src/modules/exams/entities	100	100	100	100	
exam.entity.ts	100	100	100	100	
submission.entity.ts	100	100	100	100	
src/modules/health	0	100	0	0	
health.controller.ts	0	100	0	0	1-9
health.module.ts	0	100	100	0	1-7
src/modules/queue	0	0	0	0	
queue.controller.ts	0	0	0	0	1-92
queue.module.ts	0	100	100	0	1-21
src/modules/queue/jobs	0	0	0	0	
bubble-sheet-creation.service.ts	0	0	0	0	1-40
src/modules/semesters	36.58	50	50	37.14	
semester.controller.ts	0	0	0	0	1-51
semester.module.ts	0	100	100	0	1-11
semester.service.ts	100	100	100	100	
src/modules/semesters/dto	0	100	100	0	
semester-create.dto.ts	0	100	100	0	1-16
src/modules/semesters/entities	100	100	100	100	
semester.entity.ts	100	100	100	100	
src/modules/user	80	61.76	55.55	78.78	
user.controller.ts	46.15	0	0	43.24	34,46-53,70-77,97-120
user.module.ts	100	100	100	100	
user.service.ts	100	100	100	100	
src/modules/user/dto	100	100	100	100	
user-edit.dto.ts	100	100	100	100	
src/modules/user/entities	100	100	100	100	
employee-user.entity.ts	100	100	100	100	
student-user.entity.ts	100	100	100	100	
user.entity.ts	100	100	100	100	

```
Test Suites: 4 passed, 4 total
Tests:       28 passed, 28 total
Snapshots:   5 passed, 5 total
Time:        8.664 s
```

Testing - Success Criteria


```
/**
 * Create a new semester
 * @param res {Response} - The response object
 * @param body {SemesterCreateDto} - The semester details
 * @returns {Promise<Response>} - The response object
 */
@UseGuards(AuthGuard, SystemRoleGuard)
@Roles(UserRoleEnum.ADMIN)
@Post('create')
async post(
  @Res() res: Response,
  @Body(new ValidationPipe()) body: SemesterCreateDto,
): Promise<Response> {
  try {
    await this.semesterService.createSemester(body);
    return res.status(HttpStatus.CREATED).send({
      message: 'ok',
    });
  } catch (e) {
    if (e instanceof SemesterCreationException) {
      return res.status(HttpStatus.BAD_REQUEST).send({
        message: e.message,
      });
    } else {
      return res.status(HttpStatus.INTERNAL_SERVER_ERROR).send({
        message: e.message,
      });
    }
  }
}
```

How many tests do we need for this class? **6**

What we need to check:

- Is user signed in?
- Does user has ADMIN role?
- Is the request body valid?
- Is the json body returned on success?
- Are all catch statements covered?

Key Features

Authentication System

- Google OAuth - User can sign in using their credentials (email and password) or sign in using Google Account.

Courses:

- Create course - Only the professor can create a course (backend routes are protected)
- Join a course - Student can join a course

Semesters:

- Create a semester - gives administrators the ability to manage courses

And many more...

Q&A