

# **CLASS-3**

## Sprint 2 Details

MONTH	WEEK	MON	TUE	WED	THUR	FRI	SAT	SUN	
AUG	1	5	6	7	8	9	10	11	
	2	12	13	14	15	16	17	18	Sprint 1
	3	19	20	21	22	23	24	25	
	4	26	27	28	29	30	31	1	Sprint 2
SEPT	5	2	3	4	5	6	7	8	
	Exam 1	9	10	11	12	13	14	15	
	6	16	17	18	19	20	21	22	
	7	23	24	25	26	27	28	29	Sprint 3

- Sprint 1 ends on 26 Aug 2024
- Must commit Sprint 2 by 27 Aug 2024
- Can UAT Sprint 2 by 16 Sep 2024

- SQL Error 1040: Too Many Connection
- Decoding JWT token on client
- Validate JWT token on client?

PBI	NAME	BRIEF DESCRIPTION
15	MATCH-PASSWORD	Check whether the <i>username</i> : <i>password</i> pair matches the stored credential
16	VALIDATE-MATCH-PASSWORD	Extend PBI15. Validate inputs on FE and BE
17	AUTHENTICATE-WITH-JWT-TOKENS	Extend PBI15. BE returns JWT access token when the credential is correct

- Must pass both cypress and postman tests
- password must be blinded
- must use *itbkk\_shared* << test in sprint 2
- BE must not provide any service that return password << no way to test
- [PBI17] should use secret secret/key-pairs << may see on exam
- [PBI17] token is well-formed with specified claims and 30 mins expiration time << test on postman

PBI	NAME	BRIEF DESCRIPTION
18	SIMPLE-AUTHZ	User must authenticate before using the system
19	PERSONAL-BOARD	Each user can create his/her own personal task board

User must authenticate before using the system.

FE (any page other than /login)

1. if **not** authenticated, redirect to login page
2. send access token in the header of every request to BE
3. if BE response 401, reset authentication state and redirect to login page

BE (any endpoint except /login)

1. must received valid token in the header of the request

Scenario	BE Code	FE behavior
authenticated and token is valid	as expected	as expected
<b>not authenticated / no token</b>	401	redirect to /login (use navigation guard)
<b>token is expired</b>	401	reset state and redirect to /login
<b>token is not well-formed JWT</b>	401	as above
<b>token has been tampered with</b>	401	as above

401

## Authentication Failed

Media type

application/json



Example Value | Schema

```
{
  "timestamp": "2024-04-23T08:29:03.133+00:00",
  "status": 401,
  "message": "Username or password is incorrect",
  "instance": "/api/v3/board"
}
```

the message  
depends on scenario

Each user can create his/her own personal task board

## Business/technical conditions

1. New user does not initially own any board.
2. A user can **create** one board and becomes the owner of that board.  
(may extend to own multiple boards in the future)
3. A board must be own by one and only one user.
4. **Custom** statuses (PBI6–8) is specific to each board.  
No two boards share the same custom statuses.
5. User **must** be able to perform actions in PBI1–8.
6. **(OPTIONAL)** User is able to perform actions in PBI10–14.
7. Restrictions of actions on his/her own personal board only will be implemented in PBI20.
8. Use **Nano ID** with length=10 as board-id.  
[Note that the generated id may not be unique, should handle exception from DB]



## FE (/board)


1. set as home page
2. If the user own a personal board, redirect to the user personal board (/board/:id)
3. If the user does not own any board, shows 'Create personal board' button
  1. when clicked, open modal window to specify **board name**  
SPEC: NOT-EMPTY, MAX-LENGTH(120)
  2. The default board name is "\$name personal board"
  3. validates input as in PBI16
  4. when 'Create' is clicked, send request to BE
  5. action:
    - 201: redirect to the new (empty) board
    - 401: as stated in PBI18
    - else: show "There is a problem. Please try again later." and stay on the same page


## BE (/boards)

1. **GET** method:  
returns board-id of the board own by the user specified in access token
2. **POST** method with boardName in request body:  
create a new board for the user specified in access token  
returns:
  - 201 and created board object: successful creation
  - 400: boardName is null, empty, or length > MAX-LENGTH
  - 401: as stated in PBI18

# Sample UI: Board Home (/board)

INT222  
INTEGRATED PROJECT II

 ITB-KK

 ITBKK OLARN

## Board List

itbkk-button-create

Create personal board

No	Name	Action
No personal board		

# Sample UI: Create Board Modal

INT222

INTEGRATED PROJECT II

path: /board/add

The image shows a web application interface with a modal for creating a new board. The modal is titled "New Board" and contains a text input field for the board name, which currently has the placeholder text "ITBKK OLARN personal board". At the bottom of the modal are two buttons: a green "Save" button and a grey "Cancel" button. The modal is overlaid on a background that includes a navigation bar with "ITB-KK" and "ITBKK OLARN" links, and a table with columns "No" and "Action".


Labels in the image:


- `itbkk-modal-new`: Points to the modal container.
- `itbkk-button-create`: Points to the "Create personal board" button.
- `itbkk-button-ok`: Points to the "Save" button.
- `itbkk-button-cancel`: Points to the "Cancel" button.
- `itbkk-board-name`: Points to the name input field.

# Sample UI: Board (/board/:id)


itbkk-home => /board

itbkk-fullname


 ITB-KK


 ITBKK OLARN

ITBKK OLARN personal board

Filter by stauts(es) 

Manage Status



	Title	Assignees	Status
---	-------	-----------	--------

itbkk-manage-status

GET

**/v3/boards** Gets all board

POST

**/v3/boards** Create a new board

GET

**/v3/boards/{id}** Gets board by ID

PUT

**/v3/boards/{id}** Update a board with id

DELETE

**/v3/boards/{id}** Delete a board with id

**POST** **/v3/boards** Create a new board

New board must valid

**Parameters**

No parameters

**Request body** **required**

**Example Value** | Schema

```
{  
  "name": "string"  
}
```

**Responses**

Code	Description
201	<div>Successful Operation</div> <div>Media type</div> <div><b>application/hal+json</b> ▼</div> <div>Controls Accept header.</div> <div><b>Example Value</b>   Schema</div> <div><pre>{   "id": "string",   "name": "string",   "owner": {     "oid": 0,     "name": "string"   } }</pre></div>

**GET** **/v3/boards/{id}** Gets board by ID

Board must exist

**Parameters**

Name	Description
<b>id</b> * required string (path)	id of board to be fetch

404 Failed Operation (Resource Not Found)

Media type  
application/json

Example Value | Schema

```
{
  "timestamp": "2024-04-23T08:29:03.133+00:00",
  "status": 404,
  "message": "Board id 'UQ23AXF5S' not found",
  "instance": "/api/v3/board/UQ23AXF5S"
}
```

**Responses**

Code	Description
200	Successful Operation

Media type  
application/hal+json

Controls Accept header.

Example Value | Schema

```
{
  "id": "string",
  "name": "string",
  "owner": {
    "oid": 0,
    "name": "string"
  }
}
```



## FE (other pages)

1. change `/task` path to `/board/:id` to show tasks in `{id}` board
2. change `/task/add` path to `/board/:id/task/add`
3. change `/task/:id/edit` path to `/board/:id/task/:task-id/edit`
4. change `/status` path to `/board/:id/status` to show statuses in `{id}` board
5. update fetch API URLs to new endpoints

## BE (other pages)

1. change `/tasks` to `/boards/{id}/tasks`  
[ALL methods] returns 404 if there is no board with the specified id  
[PUT/DELETE] returns 404 if the specified task is not in the specified board
2. change `/statuses` to `/boards/{id}/statuses`  
[ALL methods] returns 404 if there is no board with the specified id  
[PUT/DELETE] returns 404 if the specified status is not in the specified board

**GET**

**/v3/boards/{id}/tasks** Gets all task of specific board with no pagination and optionals sorting/filtering by status

**POST**

**/v3/boards/{id}/tasks** Create a new task on specific board

**GET**

**/v3/boards/{id}/tasks/{taskId}** Gets task of specific board by task id

**PUT**

**/v3/boards/{id}/tasks/{taskId}** Update a task with specific board id and task id

**DELETE**

**/v3/boards/{id}/tasks/{taskId}** Delete a task with specific board id and task id

- Follows logically from the requirements
- Team should try to come up with test cases first
- Actual test cases will be released later
- (postman) will test with new (unseen) user!

- Secure connection (nginx +optional spring boot)
- Authz personal board (including task/status) – private/public
- Refresh token
- Token storage