

# Introduction

Hi, my name is Nouha and my project to called the To Do List Tracker (name subject to change)



# To Do List Tracker: Concept

## User Story:

Given that I am always busy and doing so many things in the day, I would like to list the things I tend to forget on a to do list tracker. This application allows me be more organised and visibly see the things I need to complete.



# Sprint plan

- From my concept I created a sprint board with several Epics: frontend, Backend, Testing, Documentation - in a way of joining the user stories together.
- I made several user stories and tasks of what I need to do to create this applications on the biases of the Epics I have created.
- Each user story and task I created were assigned a priority, storypoint and label.
- Once completing the issues i needed to complete my application, I started the Sprint.
- Throughout my project i was able to track my progress and things i had to do and complete.

The image shows a Jira sprint board and the details of a specific issue. On the left, the 'Epic' sidebar lists four categories: Backend, Frontend, Documentation, and Testing, each with a progress bar. The main board displays a list of issues, each with a key (e.g., BAEP-13), a description, a label (FRONTEND or BACKEND), and a story point value. The issue BAEP-13 is highlighted. To the right, the 'Details' panel for BAEP-13 is shown, including the title 'As a developer, I want a input section, so that I can add my to-do-list in', the description 'Add a description...', and the activity section with a comment input field. The right sidebar shows the issue's metadata: Assignee (Unassigned), Priority (High), Labels (ShouldHave), Sprint (BAEP Sprint 1), and Story point estimate (7).

**Epic** X

Issues without epic

- > Backend
- > Frontend
- > Documentation
- > Testing

+ Create Epic

**BAEP-13** As a developer, I want a input section... FRONTEND 7

**BAEP-32** As a developer, i want a axios.js file, ... FRONTEND 10

**BAEP-34** As a developer, I want a style sheet fil... FRONTEND 6

**BAEP-30** As a developer, I want a html page, s... FRONTEND 5

☒ **BAEP-17** Design delete, save, and update butt... FRONTEND 5

**BAEP-20** As a developer, I want an exceptions p... BACKEND 7

**BAEP-22** As a developer, I want a prod and test ... BACKEND 7

☒ **BAEP-29** test if the database is connected to My... BACKEND 4

**BAEP-20** As a developer, i want to connect to th... BACKEND 3

**BAEP-9** As a developer, i want to create a servic... BACKEND 8

**BAEP-25** As a developer, i want to create a gith... BACKEND 3

**BAEP-10** As a developer, I want to create a repo... BACKEND 1

**Frontend / BAEP-13**

**As a developer, I want a input section, so that I can add my to-do-list in**

Attach Add a child issue Link issue ...

**Description**

Add a description...

**Activity**

Show: All **Comments** History Oldest first ↕

**NR** Add a comment...

Pro tip: press **M** to comment

**Done** Done

**Details**

Assignee  
Unassigned

Priority  
High

Labels  
ShouldHave

Sprint  
BAEP Sprint 1


Story point estimate  
7

# Consultant Journey

## Technologies Used:

- SpringBoot for API
- HTML, CSS, Vanilla Javascript for font end
- MySql for database

## What have I learnt:

- Someone who is new to Java, I was able to Learn to use SpringBoot to create my API and to deal with HTTP responses.
  - Instead of using fetch() method in JS, I was able to interact with my API endpoints with AXIOS. This was something new for me which required me to research a lot about during my project.
- 

# CI:

## How did you approach version control?

- This was my first proper project that i decided to use Gitbash throughout to commit my work. In previous projects I have used other versions controls to commit, push and pull my work.
- Instead of relying on the version controls I would usually use, I decided to commit and push all my work through Git bash. I was a real refresher and something i wasn't used to which made it slightly difficult. And due to this, it made me slightly cautious of what I was committing in case there were some mistakes.
- This meant I had to look up a few git commands online !

```
writing objects: 100% (10/10), 870 bytes | 435.00 KiB/s, done.
total 10 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
to github.com:nrakeeb/To-do-List.git
   a23f193..11c0660 feature-intergration-schema -> feature-intergration-schema

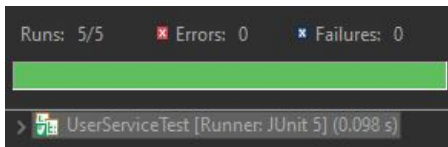
buha@DESKTOP-6QGAT0A MINGW64 ~/eclipse-workspace/ToDoList (feature-intergration-schema)
git add .
warning: CRLF will be replaced by LF in src/test/java/com/qa/ToDoList/service/UserServiceTest.java.
the file will have its original line endings in your working directory

buha@DESKTOP-6QGAT0A MINGW64 ~/eclipse-workspace/ToDoList (feature-intergration-schema)
git commit -m "NR: crated tests for UserService"
feature-intergration-schema 6e66e43] NR: crated tests for UserService
1 file changed, 74 insertions(+)

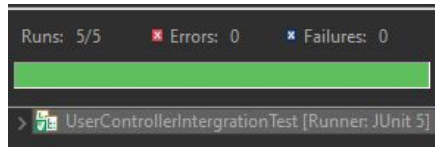
buha@DESKTOP-6QGAT0A MINGW64 ~/eclipse-workspace/ToDoList (feature-intergration-schema)
git push
Enumerating objects: 19, done.
Counting objects: 100% (19/19), done.
Delta compression using up to 4 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (10/10), 1.26 KiB | 643.00 KiB/s, done.
total 10 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
to github.com:nrakeeb/To-do-List.git
   11c0660..6e66e43 feature-intergration-schema -> feature-intergration-schema
```

# Testing:

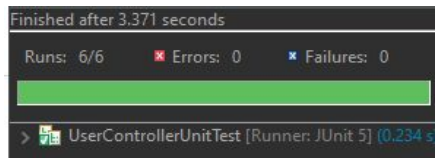
- I was able to create two class: User Controller Integration Test and User Controller Unit Test. Both of these were successful of completing the test.



Additionally, I also created a User Service Test class that also was successful.



I was able to test 6 crud for User Controller User Unit Test (getAll, getByld, Create, update, delete, delete fail test)



I was able to test 5 crud for User Controller Integration Test (getAll, getByld, Create, update, delete)

## Coverage :

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
▼  ToDoList	36.4 %	318	555	873
>  src/test/java	26.0 %	165	469	634
▼  src/main/java	64.0 %	153	86	239
>  com.qa.ToDoList.domain	32.7 %	32	66	98
>  exceptions	0.0 %	0	9	9
>  com.qa.ToDoList	37.5 %	3	5	8
>  com.qa.ToDoList.Controller	93.0 %	53	4	57
>  com.qa.ToDoList.service	97.0 %	65	2	67

# Demonstration

## My To Do List

Add Task

### Things to do

Wash my car

UPDATECOMPLETE

Go food shopping

UPDATECOMPLETE

Book my holiday

UPDATECOMPLETE

return books to library I

UPDATECOMPLETE



# Sprint review

What was I able to complete?

- I was able to complete most of the tasks on my Sprint board. All the necessary things anyway! I prioritised the API more so than my frontend as tasks first.
- Originally, I wanted to create an interactive To do List where i could add features like inserting prices and dates of when they should have been completed.

What got left behind?

- I created “CanHave” ticket for another html page. This ticket would have required me to create another HTML page that welcomed users to the TO DO List website. This would have been decorative and had button in the center. When button is clicked it would have called a function to get all the previous todos in the database and present the user to another html page where they can add their todo tasks. As this wasn't mvp I did not prioritised with it and would have only dealt with the ticket if i had extra time.





# Sprint retrospective

What went well?

- I was able to meet mvp and created an application that sends CRUD requests and present them on my web page.

What i could have improved on?

- Time management
- Leave extra time for tests or bugs



# Conclusion

- Overall, this was both a challenging and fun project to do. I think I was going to extend this project there would be several things I would like to improve on.
  1. I would like to make this application adjustable to responsive layouts. If given the chance I would redo this project in React. This would have easily allowed me to to develop a responsive layout without the use of media query or css.
  2. I would also had authentication for users to sign into their accounts which had their stored todo lists there.



# Questions

Do you have any questions?



## Risk Assessment

Description	Evaluation	Likelihood	Impact Level	Response	Responsibility	Control Measures
Computer Breaks Down	Unable to complete my project	Low	High	Quickly must notify appropriate staff	Developers	Ensure all my recent work is committed to github to reduce the risk of losing work
Personal/ Health incident	Either delays in completing work or not meet mvp	Medium	Medium	Notify staff and let them be aware of any issues so they can provide advice	Developers	If issues that could arise, always notify your colleague or staff so they are aware of it if anything goes wrong.
Merge conflicts	Unable to pull recent work	Medium	Low	Resolve the merge conflicts on github	Developers	Make sure to pull the recent content before commit anything into branch
Deployed and submit broken version of application	Unable to present finished work	Medium	High	Revert production to most recent stable version	Developers	Automate tests before any deployment & restrict access to production branch
Burnout	Feeling unmotivated to complete my work	High	High	Must take regular breaks and speak to someone for help	Developers	Try to meet MVP. do not try to do other things that may prolong your time. This will reduce burnout

# Images of Endpoints: Post and Put

## Create

POST localhost:8080/Notes/create

Params Authorization Headers (8) **Body** Pre-request Script

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary

```
1 [raw]
2 ... "comments": "buy a present for best friend"
3 ...
4 ...
5 ...
6 [copy]
```

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON

```
1 [raw]
2 "id": 44,
3 "comments": "buy a present for best friend"
4 [copy]
```

## Update

PUT localhost:8080/Notes/update/41

Params Authorization Headers (8) **Body** Pre-request Script Tests S

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL

```
1 [raw]
2 ... "comments": "Go clothes shopping instead"
3 ...
4 ...
5 ...
6 [copy]
```

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON

```
1 [raw]
2 "id": 41,
3 "comments": "Go clothes shopping instead"
4 [copy]
```

# Images of Endpoints: GetAll and GetById

## Get All

GET localhost:8080/Notes/getAll

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   {
3     "id": 37,
4     "comments": "test today"
5   },
6   {
7     "id": 39,
8     "comments": "Take the dog for a walk again"
9   },
10  {
11    "id": 40,
12    "comments": "Wash my car"
13  },
14  {
15    ...
16  }
17 }
```

## Get By Id

GET localhost:8080/Notes/getById/39

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 39,
3   "comments": "Take the dog for a walk again"
4 }
```

# Images of Endpoints: Delete

The screenshot displays a REST client interface with the following components:

- Request Bar:** Method `DELETE` and URL `localhost:8080/Notes/delete/42`. A `Send` button is on the right.
- Tabs:** Params, Authorization, Headers (6), Body, Pre-request Script, Tests, Settings. A `Cookies` tab is also visible on the right.
- Response Section:** Includes tabs for Body, Cookies, Headers (6), and Test Results. The `Body` tab is active, showing a status of `204 No Content`, a time of `28 ms`, and a size of `201 B`. A `Save Response` button is present.
- Body View:** Offers `Pretty`, `Raw`, `Preview`, and `Visualize` options. The `Text` view is selected, displaying the number `1`.

Delete