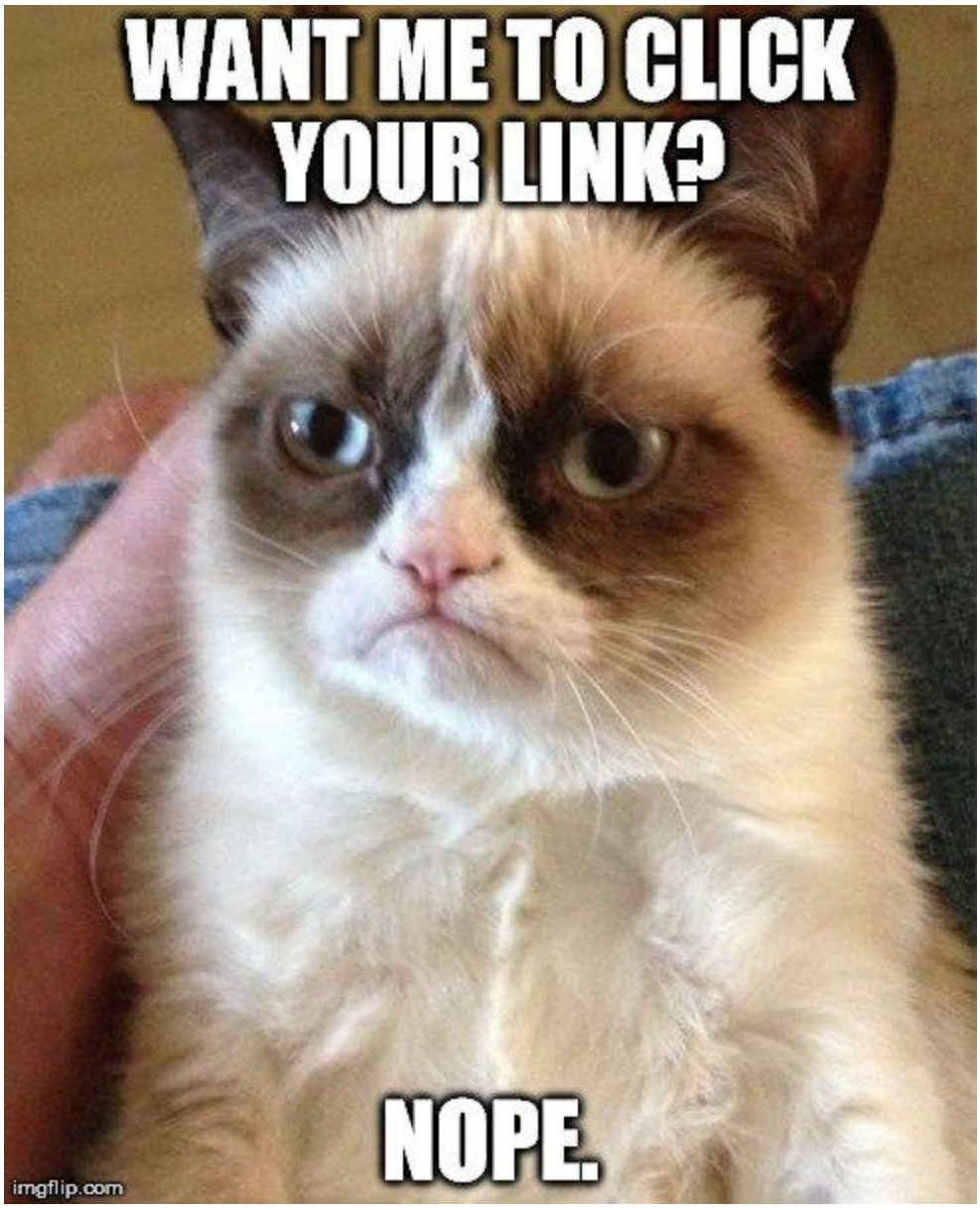
Playground Requirements



Bootcamp app features

Ground Rules

1. **Know that this exercise is not meant to judge you in any way. It is meant for you to learn something fantastic and something really REALLY IMPORTANT!**
2. **Please be strict to the deadline, as we will have to test your features. No extensions will be provided.**
3. **Please take this exercise EXTREMELY SERIOUSLY! Top leadership, including Guru & the HR would be actively looking the same.**
4. It would be best if you do the exercise on your own without seeking help from the senior Tekion engineers. It would lead to your best growth.
5. It's a SECURITY BOOTCAMP. Keep that in mind!
6. Read through the requirements doc VERY CAREFULLY! There are **Notes** mentioned in some places. Please do not ignore them
7. All the below requirements are given by the business to your product manager. Feel free to raise questions around the same if you think that the requirement may not be a good idea.
8. USE PLAIN HTML AND JS if you want to keep it simple. We are not looking for any fancy frontend frameworks if you do not want to use them.
9. For your respective feature you should be able to demo with at least 2 different users
10. Don't get stuck at one point, come and ask TekioKnights (@ Abhinav Chourasia @ Abhishek Bundela @ Ankit Kumar @ Budde Prashanth @ Priyal @ Yogeshwaran N C) on **the channel bootcamp-2023-bangalore** Teams channel. You can ask us AN YTHING AND EVERYTHING in regards to this exercise.
11. *We encourage you to think in a secure perspective about everything, so please come and ask us or rather discuss with us about your thinking if you want to.*
12. Beautiful UI is not a hard requirement for us.
13. Feel free to change the schema of tables as required.
14. Do not change the base app unless you think is is absolutely needed to be changed. In that case, please make the changes to your version of the base app and let us know what changes you did and why
15. **Group and Repository Rules:**
    a. You have been assigned a group number.
    b. Each group has 20 members, subdivided into subgroups of 4.
    c. Some features may be bigger than the others. Feel free to have bigger features built by more than one subgroup.
    d. Some features may also be related to each other but may not be in the same order. Don't bother about the order in which the features have been mentioned in this page. Feel free to mix and match as you deem fit.
    e. Each subgroup **MUST** build at least one feature.
    f. The readme of your repo should mention which subgroup created which feature/s
    g. Each group should check out a new branch expressing the correct name for the branch

    and develop respective features.
    h. The branch name should be group name and nothing else
    i. Each subgroup within a given group should create their own respective subgroup branches, naming it like `group1-subgroup1`
    j. All subgroup branches should be eventually merged to the group branch as towards the end of the development, we will deploy your group branch alone and that will be considered your working application.
    k. You will have to demo the full working application (with all the features included) on the remote Ubuntu server provided to you.
    l. Each group will have access to their own dedicated Ubuntu server. This server will be accessible from the office alone. If you're accessing it from your home, you would have to login to the VPN to be able to access it.
    m. You will be doing the local development most probably on your respective Mac machines. But you would be deploying it on the remote Ubuntu server provided to you. There could be changes needed to your setup because of this change. Please ensure you regularly check the setup on the remote machine also to ensure everything is working fine on the remote Ubuntu server as well, instead of hitting last minute hassles.
    n. On Teams you will have the team named `bootcamp-2023-bangalore` created with a `General` channel. All of you can discuss common items on this channel. There will also be individual channels created for each group, like group-1, group-2 etc. These channels are private to each group and will be accessible only to the specific group members alone. For each group we will message the PEM file using which you/your group members will be able to SSH into your respective remote Ubuntu server. **Keep the PEM file of your group strictly confidential. Do NOT share or distribute it with other teammates of yours. If found in violation of this, you may be subject to disciplinary action.**
    o. You will be able to SSH into the Ubuntu server only using the Anyconnect VPN. Follow the instructions at Cisco Any Connect VPN Access - For Mac / Windows --Only For VPN Users to setup Anyconnect VPN on your machines.
    p. The public IP assigned to the remote Ubuntu server assigned to your respective group may change. Hence, one member from each team may be allowed to login to AWS console. We will notify in each group who that person is. This person would be able to see the public IP of the Ubuntu server assigned to his/her respective group.
    q. Repository where the base app can be found and where all your code should be pushed (as per the guidelines mentioned above) https://bitbucket.org/tekion/bootcamp-playground/

Deadlines

Bangalore Dates	Agenda	Notes

05/Feb/23	API doc for desired features should be given	API doc will not be explained on Day 0. Any clarifications can be sought from the security Bootcamp team offline
14/Feb/23 (Tue)	First round of demo to Security assigned SPOC	Bootcampers feature on top of the base app should be 50-60% complete at least and they should be able to demo it
24/Feb/23 (Fri)	Second round of demo to Security assigned SPOC	100% feature demo & all commits to the repo would be denied post the discussion with the SPOC
28/Feb/23 (Tue)	Security team will ensure if the deployed code on the ec2 for each team is working fine or not.	NA
13/Mar/23 - 15/Mar/23	Bootcamp starts, Enjoy, learn, teach	FUN BEGINS!

1. Set and get a user profile picture

i). Set Profile pic

Request:

```
POST /profilePic HTTP/1.1
Host: localhost:4567
Content-Type: multipart/form-data; boundary=------------------------
0e9271108a4aba8f

--------------------------0e9271108a4aba8f
Content-Disposition: form-data; filename="pic.png"
Content-Type: application/octet-stream

file bit stream
--------------------------0e9271108a4aba8f--
```

Response:

```
{"success":"true","error":""}
```

i). Get Profile pic

Request:

```
GET /profilePic HTTP/1.1
Host: localhost:4567
```

Response:

```
{"success":"true","error":""}
```

Once the profile picture is set, user should be able to view the uploaded picture on their homepage.

---

2. Edit User Profile

i) edit user profile

```
POST /EditProfile?profile-id=<> HTTP/1.1
Host: localhost:4567
Content-Type: application/json

{"firstName":"Abhishek","lastName":"Bundela"}
```

Response:

```
{"success":"true","error":""}
```

Note: Keep the profile-id small and simple if you want to

ii). Get user profile

```
GET /Profile/<profile-id-of-length-5-goes-here> HTTP/1.1
Host: localhost:4567
```

```
{"success":"true","firstName":"Abhishek", "lastName":"Bundela",
"image_url":"https://test.s3.aws.com/test.png","error":""}
```

---

3. Note sharing with an allowed email

i). Unlock view permission for a given email address.

Request:

```
POST /NotePermission HTTP/1.1
Host: localhost:4567
Content-Type: application/json
Content-Length: 31

{
"noteid":[noteid-1,noteid-2...],
"logged_in_user":"abundela@tekion.com",
"email":["target-user-1-emailid",
"target-user-2-emailid",...],
}
```

Response:

```
{"success":"true","error":""}
```

ii). Lock view permission:

Request:

```
POST /NotePermission HTTP/1.1
Host: localhost:4567
Content-Type: application/json
Content-Length: 31

{
"noteid":[noteid-1,noteid-2...],
"logged_in_user_email":"abundela@tekion.com",
}
```

Response:

```
{"success":"true","error":""}
```

Note:

- Provision SMTP server and credential through a public mail services like https://www.smtp2go.com/ **OR ALTERNATIVELY**
- We have used our own gmail id and password in the base app. In a similar way, if you want to, you could use your own gmail credentials to do the needed

---

4. S3-based logging and debugging: Push the backend application logs in the s3 bucket.

Request:

```
POST /PushLog HTTP/1.1
Host: localhost:4567
Content-Type: application/json
Content-Length: 31

{
"s3_bucket":"https://s3.aws.com/logbucket",
"data": "stringify log"
}
```

Response:

```
{"success":"true","error":""}
```

NOTE: *Your logs should be as verbose as possible to facilitate debugging easily*

---

5. Implement password-based login, reset password, and update password feature (13 member group)

i). Set password for login

Request:

```
POST /SetPassLogin HTTP/1.1
Host: localhost:4567
Content-Type: application/x-www-form-urlencoded
Content-Length: 31

email=abundela@tekion.com&password=test123
```

Response:

```
{"success":"true","error":"error-message/none"}
```

ii). Login through email and password

```
POST /PassLogin HTTP/1.1
Host: localhost:4567
Content-Type: application/x-www-form-urlencoded
Content-Length: 31

email=abundela@tekion.com&password=test123
```

Response:

```
{"success":"true","token":"session token","error":""}
```

iii). Update the password corresponding to an email

Request

```
POST /UpdatePass HTTP/1.1
Host: localhost:4567
Content-Type: application/x-www-form-urlencoded
Content-Length: 31

email=currently-loggedin-users-email&newPass=test123
```

Response:

```
{"success":"true","token":"","error":""}
```

iv). Reset the user password by sending a password reset link to their email.

Request:

```
POST /ResetPass HTTP/1.1
Host: localhost:4567
Content-Type: application/x-www-form-urlencoded
Content-Length: 31

email=abundela@tekion.com
```

Response:

```
{"success":"true","error":""}
```

v). Set the new password through the reset link.

```
POST /ResetLink?token=sometoken HTTP/1.1
Host: localhost:4567
Content-Type: application/x-www-form-urlencoded
Content-Length: 31

password=random12324
```

Response:

```
{"success":"true","error":""}
```

Note:

- Provision SMTP server and credential through a public mail services like https://www.smtp2go.com/ **OR ALTERNATIVELY**
- We have used our own gmail id and password in the base app. In a similar way, if you want to, you could use your own gmail credentials to do the needed

---

6. Rich content editor functionality in the UI: Rich text editor allows to edit, and add content, tables, links, images, and other web components on a webpage with or without writing code.

Reference: https://froala.com/blog/editor/a-beginners-guide-to-rich-text-editors/

7. Send notes on a given email:

Request:

```
POST /EmailNote HTTP/1.1
Host: localhost:4567
Content-Type: application/json
Content-Length: 31

{"email":"abundela@tekion.com", "note_id":1}
```

Response:

```
{"success":"true","error":""}
```

Reference: Provision SMTP server and credential through a public mail services like https://www.smtp2go.com/

8. Secure notes by setting a PIN

Request:

```
POST /SetNotePin HTTP/1.1
Host: localhost:4567
Content-Type: application/json
Content-Length: 31

{"note_id":"1", "pin":"1234"}
```

Response:

```
{"success":"true","error":""}
```

ii). Access pin-protected notes:

```
POST /AccessPinProtectedNotes HTTP/1.1
Host: localhost:4567
Content-Type: application/json
Content-Length: 31

{"note_id":"1", "pin":"1234"}
```

Response:

```
{"success":"true","data":"",error":""}
```

---

9. Import and export notes

i). Import CSV notes

Request:

```
POST /ImportNoteFromCSV HTTP/1.1
Host: localhost:4567
Content-Length: 247
Content-Type: multipart/form-data; boundary=------------------------
0e9271108a4aba8f
Connection: close

--------------------------0e9271108a4aba8f
Content-Disposition: form-data; filename="note.csv"
Content-Type: application/octet-stream

note
this is first note
and this is the second one
--------------------------0e9271108a4aba8f--
```

Response:

```
{"success":"true","data":"",error":""}
```

ii). Export notes in CSV

```
GET /ExportNotes HTTP/1.1
Host: localhost:4567
```

Response:

```
HTTP/1.1 200 OK
Content-Type: text/csv
Content-Disposition: attachment; filename="note.csv"

note
this is first note
and this is the second one
```

10. Import notes from a web link: A user can paste the CSV link in the UI and the backend will make an api call to fetch the notes in the logged-in user's account.

```
POST /ImportNotesFromLink HTTP/1.1
Host: localhost:4567
Content-Type: application/json

{"weblink":"https://anysite.com/mynote.csv"}
```

Response:

```
{"success":"true","data":"",error":""}
```

11. Attach a file in a note, generate and store the compressed file version on the server

Add attachment:

```
POST /AddAttachement HTTP/1.1
Host: localhost:4567
Content-Length: 247
Content-Type: multipart/form-data; boundary=------------------------
0e9271108a4aba8f
Connection: close

--------------------------0e9271108a4aba8f
Content-Disposition: form-data; name="note_id"

1
--------------------------0e9271108a4aba8f
Content-Disposition: form-data; filename="note.csv"
Content-Type: application/octet-stream

note
this is first note
and this is the second one
--------------------------0e9271108a4aba8f--
```

Response:

```
{"success":"true","data":"",error":""}
```

12. Publish the note to the public and allow users to upvote and downvote the published note

i). Make note public

Request:

```
POST /PublishNote HTTP/1.1
Host: localhost:4567
Content-Type: application/json

{"note_id":"1"}
```

Response:

```
{"success":"true","data":"",error":""}
```

ii). Upvote the published note

Request:

```
POST /UpvoteNote HTTP/1.1
Host: localhost:4567
Content-Type: application/json

{"note_id":"1"}
```

Response:

```
{"success":"true","data":"{"count":1}",error":""}
```

iii). Downvote the published note

Request:

```
POST /DownVote HTTP/1.1
Host: localhost:4567
Content-Type: application/json

{"note_id":"1"}
```

Response:

```
{"success":"true","data":"{"count":1}",error":""}
```

13. Give bonus likes to a public note

For every 9 unique upvotes on a public note, one upvote should be given as a bonus upvote by the system. For every 10 upvotes of a public note 1 USD earnings is credited to the author of the note. User should be able to see their total USD earnings on their profile.

14. Persist notes draft but not across logged in sessions

User should be able to save their in draft notes temporarily. But once they logout, the saved drafts should not be available anymore. The saved draft should be available only within a given logged in session.