this.jobs - sprint 3

Group Members

Ranjeet Mallipeddi (Frontend)
Syama Vangmayi Vydyula (Frontend)
Vishnuvardhan Reddy Jammula (Backend)
Sai Sneha Paruchuri (Backend)

Github repository link: https://github.com/flash29/this.jobs

Outline

this.jobs is a platform where people can build their profile, connect with other users who share similar interests in careers and find/ apply or post new jobs

Demo

Complete demo is here

Backend demo can be found here

Frontend demo can be found here

Technical stack, their pre-requisites and how to setup and run both frontend and backend can be found at this wiki

Backend accomplishments

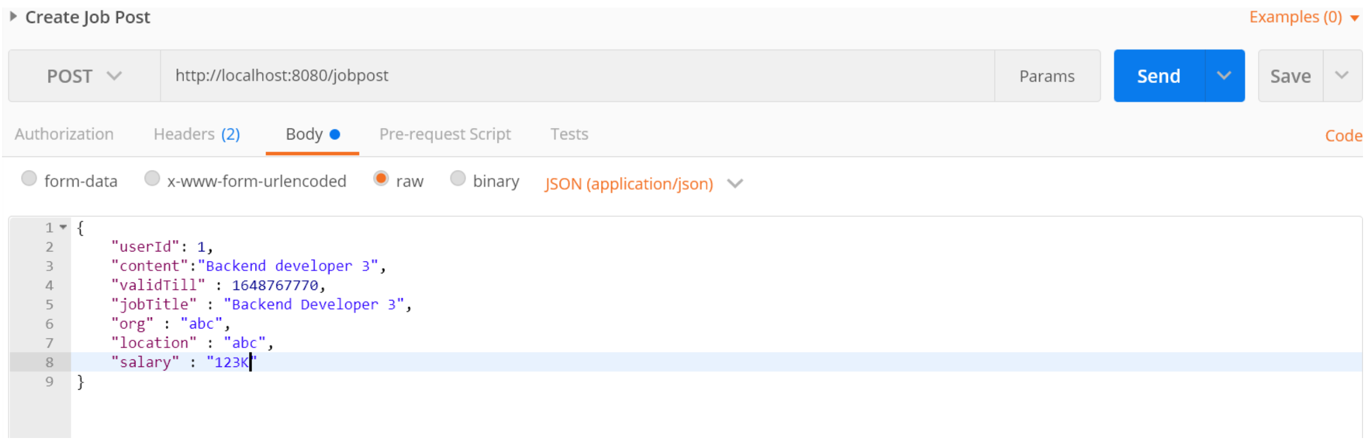
- Created REST API's to upload resume in user profile, create, update and delete jobs as a recruiter and apply to jobs as user, few other APIs to display the list of existing jobs, applied jobs and the users who applied to the jobs posted by logged in user. Api's accept json as data input and produces json responses
- Create Job takes basic details like job title, organization, salary, deadlines and location. All the details can be updated.
- Users can view all the posted jobs and apply to them. Multiple applications are not allowed and recruiters will not accept the applications post the mentioned deadline.
- Defined the data models for jobs and applications. GORM is used to automigrate the model schema to SQLite tables.
- All the data is persisted and fetched from SQLite tables related to the application.
- Unitests are created for all the APIs in the appropriate controller files.
- More about REST api's documentation can be found at this wiki

REST API

Create a Job

URL: `<base_url>/jobpost`

Request Method: POST



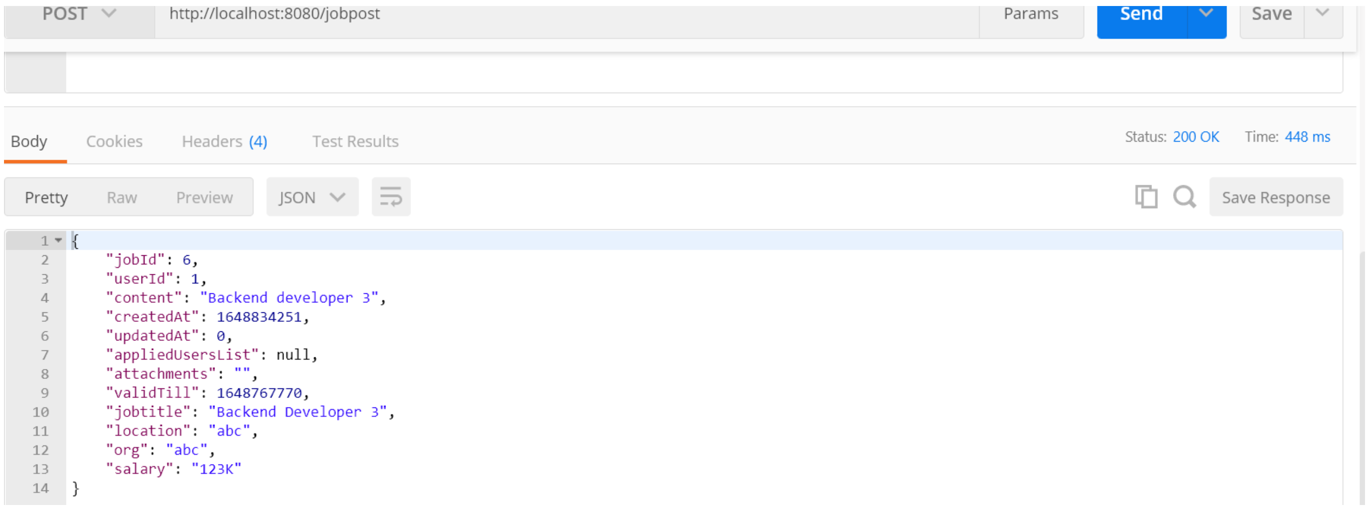
http://localhost:8080/jobpost

```
{
    "userId": 1,
    "content":"Backend developer 3",
    "validTill" : 1648767770,
    "jobTitle" : "Backend Developer 3",
    "org" : "abc",
    "location" : "abc",
    "salary" : "123K"
}
```

Id associated to the job is an auto-incrementing value and is assigned directly in the database. `userId, content, jobTitle, org, validTill, location` and `salary` are required fields to post a job. Response:

Possible Response status : `200, 400`

Example: Response status 200 The job has been created and the response with status 200 shows the newly created job details with id.



http://localhost:8080/jobpost

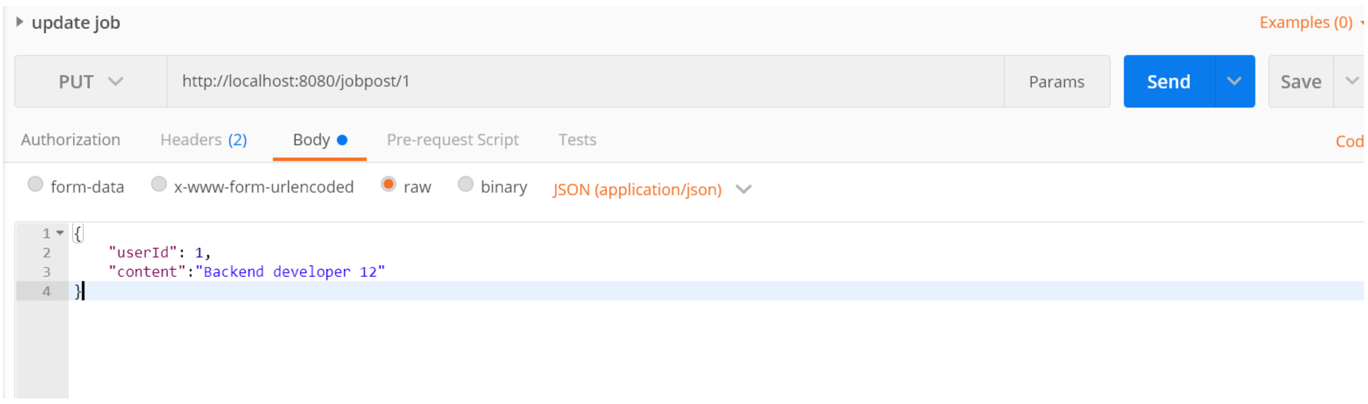
Status: 200 OK Time: 448 ms

```
{
    "jobId": 6,
    "userId": 1,
    "content": "Backend developer 3",
    "createdAt": 1648834251,
    "updatedAt": 0,
    "appliedUsersList": null,
    "attachments": "",
    "validTill": 1648767770,
    "jobtitle": "Backend Developer 3",
    "location": "abc",
    "org": "abc",
    "salary": "123K"
}
```

Update Job

URL: `<base_url>/jobposts/<job_id>`

Request Method: PUT



http://localhost:8080/jobpost/1

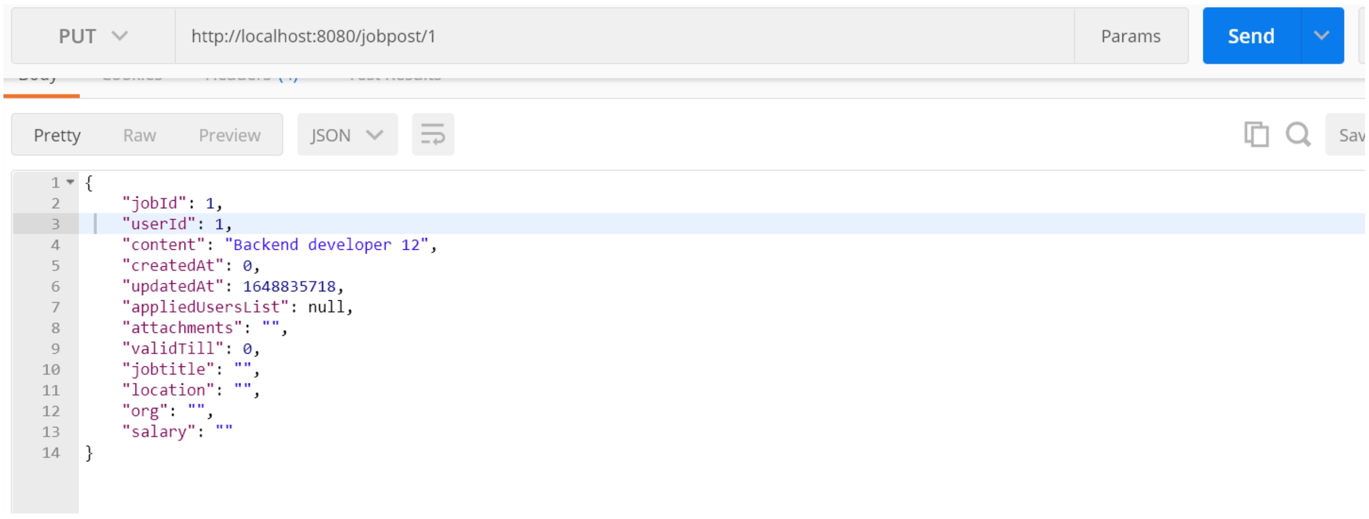
```
{
    "userId": 1,
    "content":"Backend developer 12"
}
```

The `jobId` which is sent as URL parameter is required field, remaining fields which are to be updated should be sent over payload. Response:

Possible Response status : `200, 400`

Example:

Response status : `200`



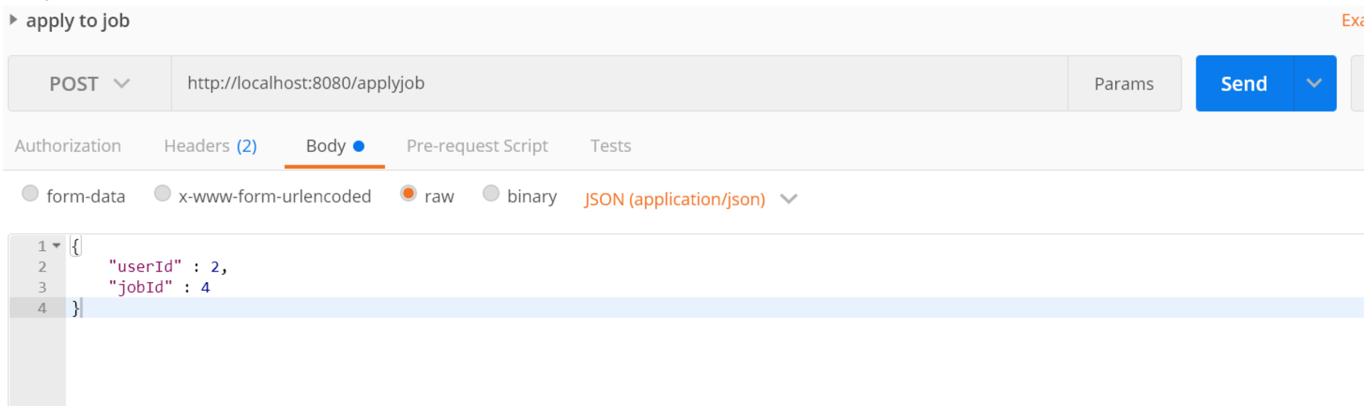
http://localhost:8080/jobpost/1

```
{
    "jobId": 1,
    "userId": 1,
    "content": "Backend developer 12",
    "createdAt": 0,
    "updatedAt": 1648835718,
    "appliedUsersList": null,
    "attachments": "",
    "validTill": 0,
    "jobtitle": "",
    "location": "",
    "org": "",
    "salary": ""
}
```

Apply to a Job

URL: `<base_url>/applyjob`

Request Method: `POST`



http://localhost:8080/applyjob

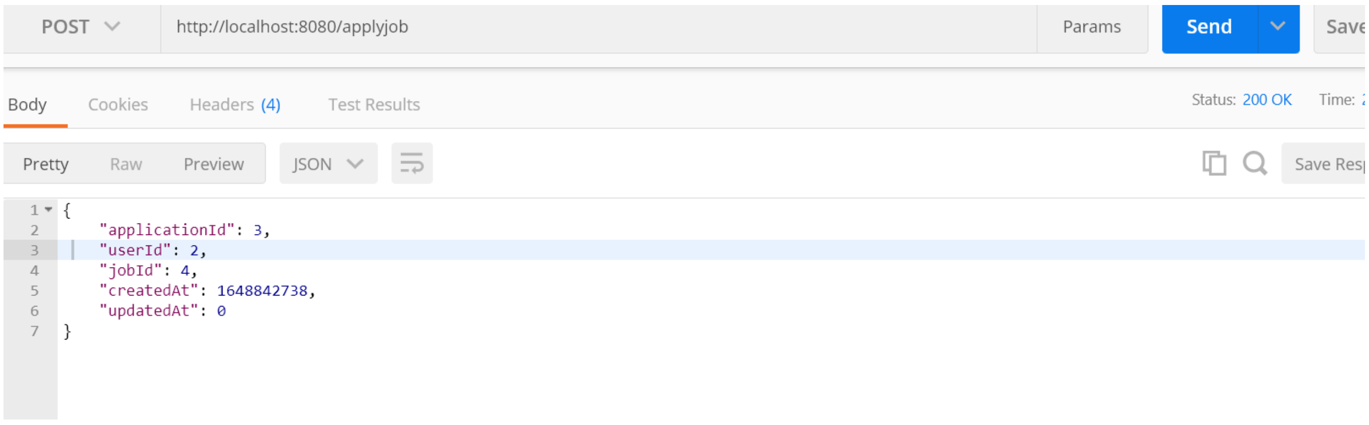
```
{
    "userId" : 2,
    "jobId" : 4
}
```

Applying `userId` and the `jobId` to which they are applying are required fields. Possible Response status: `200, 400`

Message format: `json`

Example

Code: 200 OK

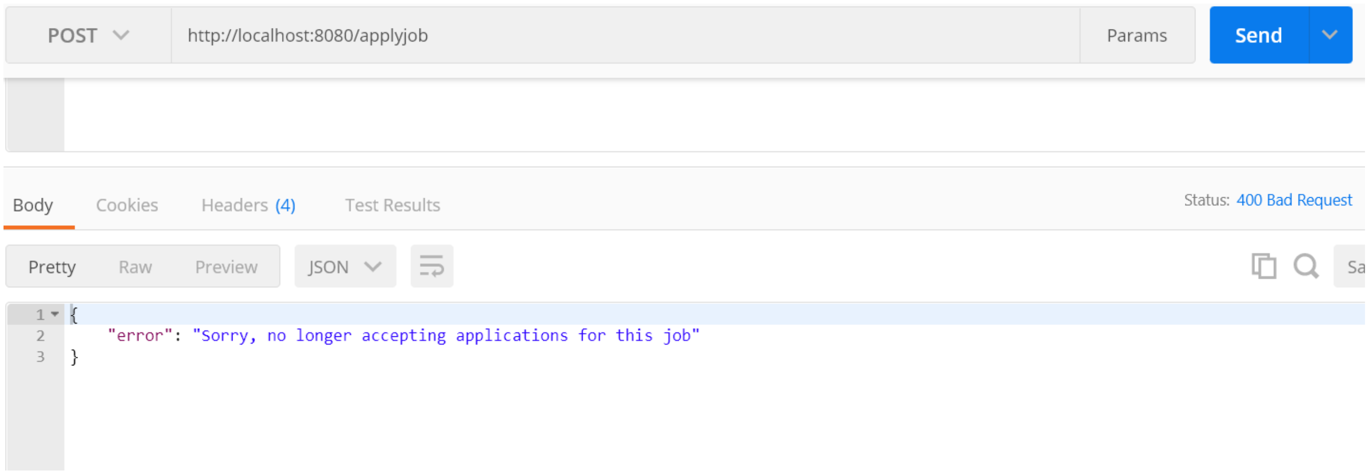


http://localhost:8080/applyjob

```
{
    "applicationId": 3,
    "userId": 2,
    "jobId": 4,
    "createdAt": 1648842738,
    "updatedAt": 0
}
```

If the user tries to apply after the deadline, the following error message is sent.

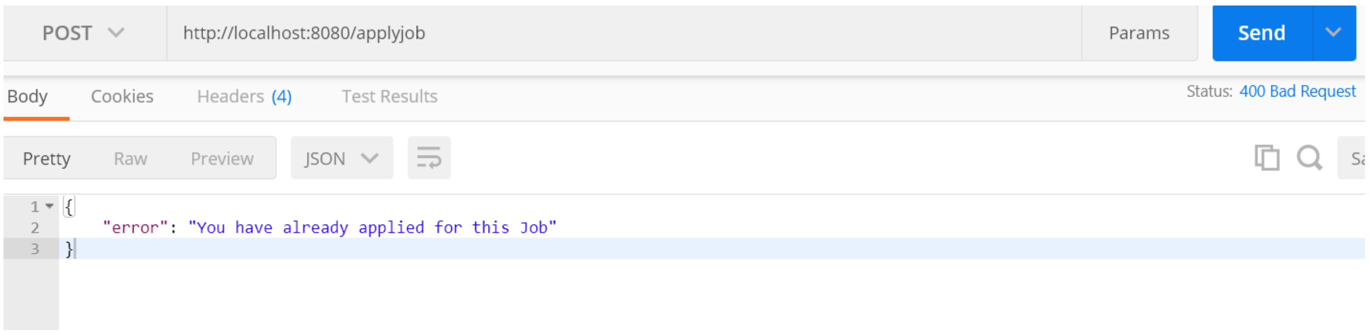
Code: 400 Bad Request



http://localhost:8080/applyjob

```
{
    "error": "Sorry, no longer accepting applications for this job"
}
```

In case of duplicate appliction the error message is as below



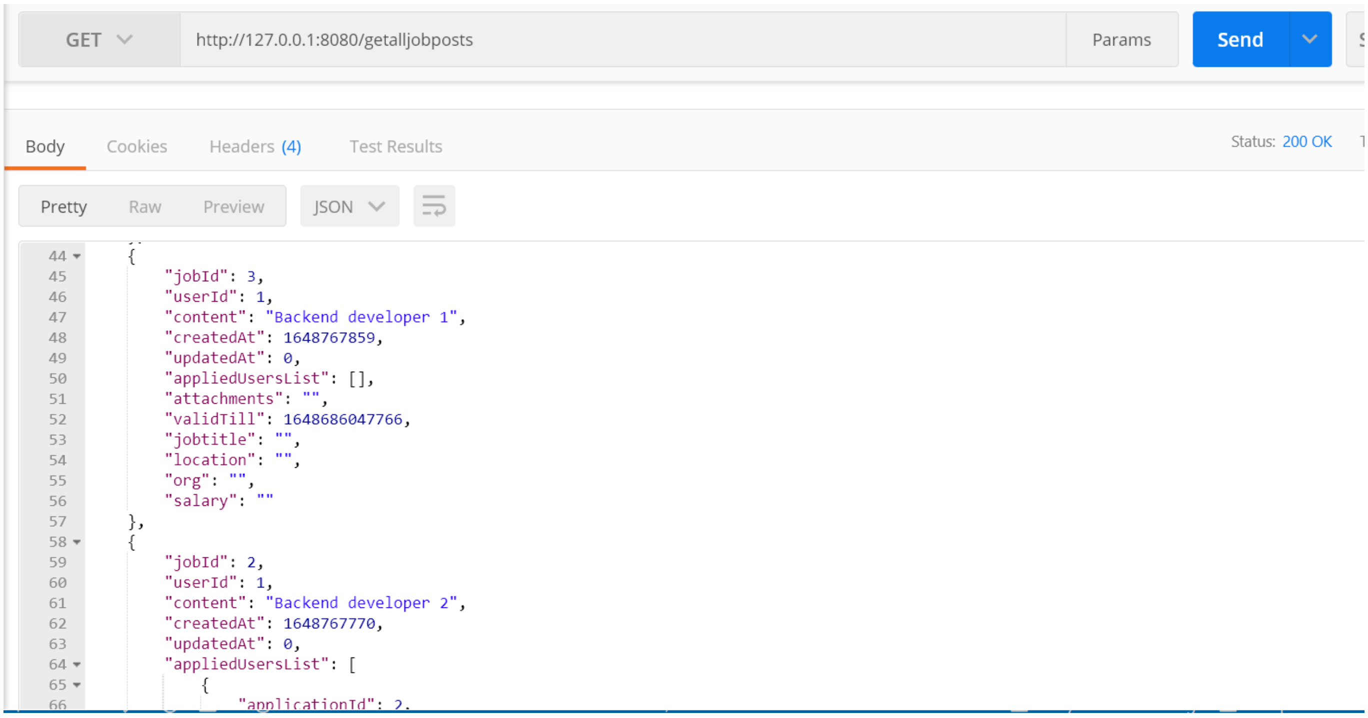
http://localhost:8080/applyjob

```
{
    "error": "You have already applied for this Job"
}
```

Retrieve Jobs

URL: `<base_url>/getalljobs`

Request Method: `GET`



http://127.0.0.1:8080/getalljobposts

```
    {
        "jobId": 3,
        "userId": 1,
        "content": "Backend developer 1",
        "createdAt": 1648767859,
        "updatedAt": 0,
        "appliedUsersList": [],
        "attachments": "",
        "validTill": 1648686047766,
        "jobtitle": "",
        "location": "",
        "org": "",
        "salary": ""
    },
    {
        "jobId": 2,
        "userId": 1,
        "content": "Backend developer 2",
        "createdAt": 1648767770,
        "updatedAt": 0,
        "appliedUsersList": [
            {
                "applicationId": 2,
```

Possible Response status: `200, 404`

Message format: `json`

Example

Code: 200 OK

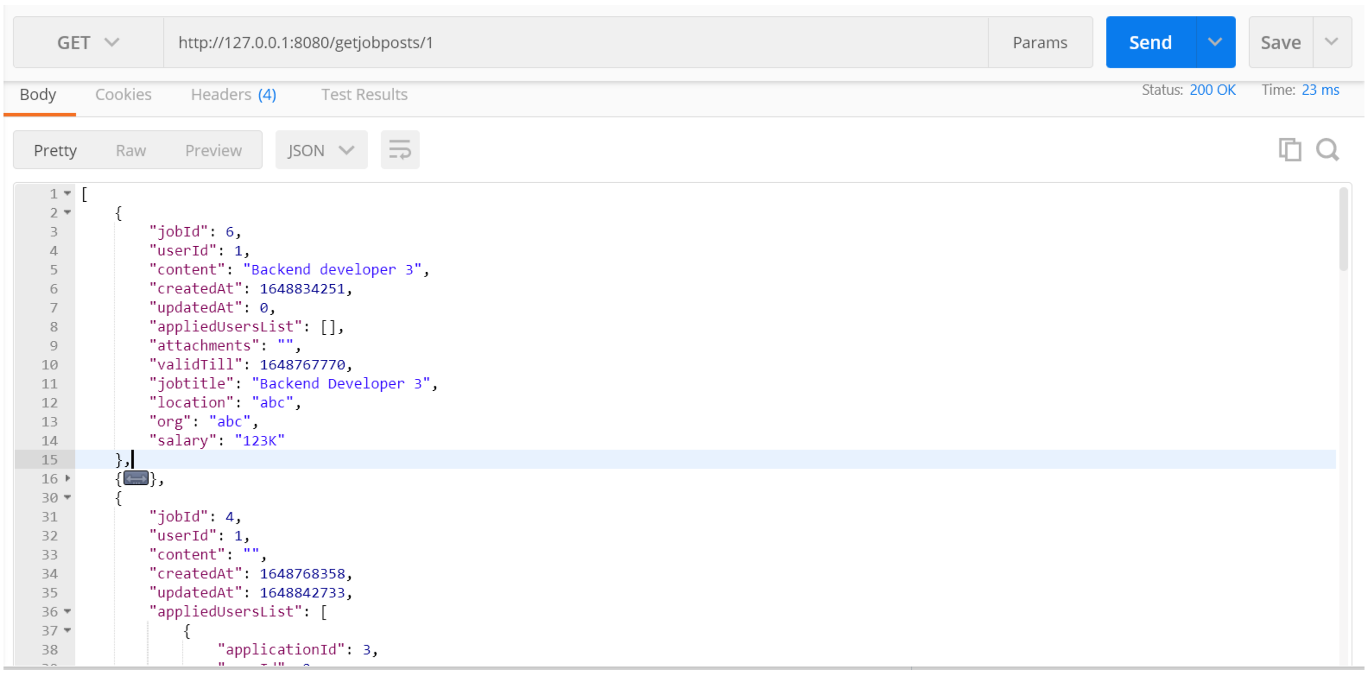
```
[
    {
        "jobId": 4,
        "userId": 1,
        "content": "Job posting 1",
        "createdAt": 1648753603,
        "updatedAt": 0,
        "appliedUsersList": [],
        "attachments": "",
        "validTill" : 1648767770,
        "jobTitle" : "Backend Developer 3",
        "org" : "abc",
        "location" : "abc",
        "salary" : "123K"
    }
]
```

Retrieve Jobs posted by Recruiter

List of jobs along with the Applicant Ids are retrieved.

URL: `<base_url>/getjobs/<user_id>`

Request Method: `GET`


GET http://127.0.0.1:8080/getjobposts/1

Status: 200 OK Time: 23 ms

```
[
    {
        "jobId": 6,
        "userId": 1,
        "content": "Backend developer 3",
        "createdAt": 1648834251,
        "updatedAt": 0,
        "appliedUsersList": [],
        "attachments": "",
        "validTill": 1648767770,
        "jobtitle": "Backend Developer 3",
        "location": "abc",
        "org": "abc",
        "salary": "123K"
    },
    {},
    {
        "jobId": 4,
        "userId": 1,
        "content": "",
        "createdAt": 1648768358,
        "updatedAt": 1648842733,
        "appliedUsersList": [
            {
                "applicationId": 3,
```

Possible Response status: `200, 404`

Message format: `json`

Example

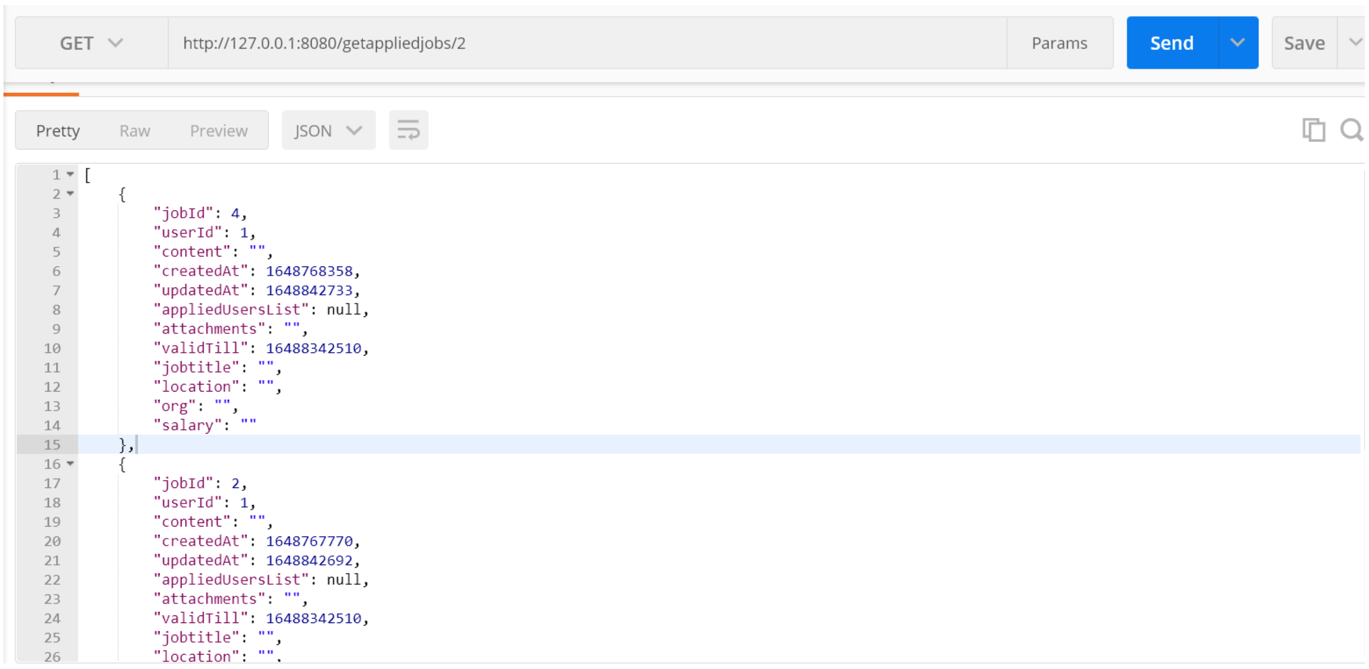
Code: 200 OK

```
[
    {
        "jobId": 4,
        "userId": 1,
        "content": "Job posting 1",
        "createdAt": 1648753603,
        "updatedAt": 0,
        "appliedUsersList":  [
                                {
                                "applicationId": 2,
                                "userId": 2,
                                "jobId": 2,
                                "createdAt": 1648767785,
                                "updatedAt": 0
                                }
                            ],
        "attachments": "",
        "validTill" : 1648767770,
        "jobTitle" : "Backend Developer 3",
        "org" : "abc",
        "location" : "abc",
        "salary" : "123K"
    }
]
```

Retrieve My Applications

URL: `<base_url>/getappliedjobs/<user_id>`

Request Method: `GET`



http://127.0.0.1:8080/getappliedjobs/2

```
[
    {
        "jobId": 4,
        "userId": 1,
        "content": "",
        "createdAt": 1648768358,
        "updatedAt": 1648842733,
        "appliedUsersList": null,
        "attachments": "",
        "validTill": 16488342510,
        "jobtitle": "",
        "location": "",
        "org": "",
        "salary": ""
    },
    {
        "jobId": 2,
        "userId": 1,
        "content": "",
        "createdAt": 1648767770,
        "updatedAt": 1648842692,
        "appliedUsersList": null,
        "attachments": "",
        "validTill": 16488342510,
        "jobtitle": "",
        "location": "",
```

Possible Response status: `200, 404`

Message format: `json`

Example

Code: 200 OK

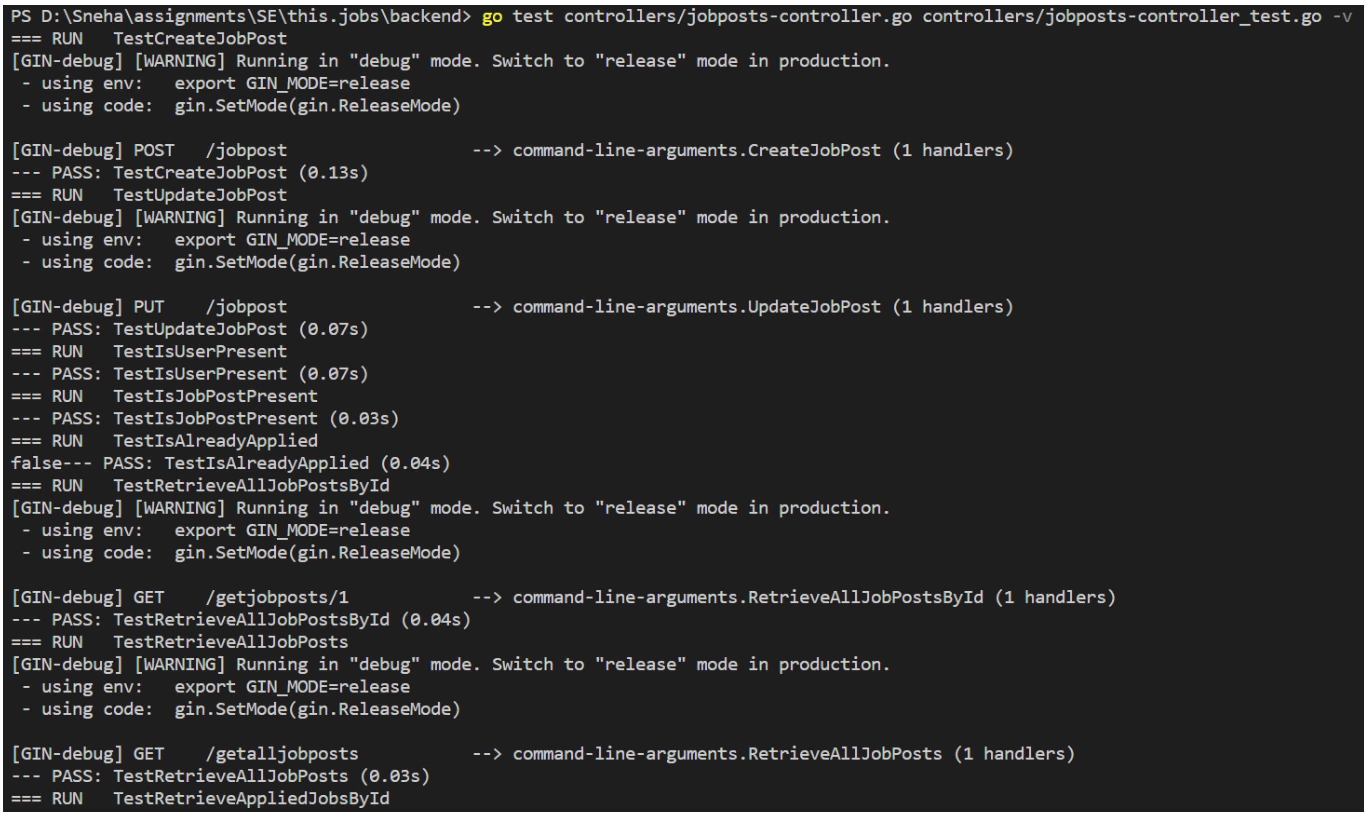
```
[
    {
        "jobId": 4,
        "userId": 1,
        "content": "Job posting 1",
        "createdAt": 1648753603,
        "updatedAt": 0,
        "appliedUsersList": [],
        "attachments": "",
        "validTill" : 1648767770,
        "jobTitle" : "Backend Developer 3",
        "org" : "abc",
        "location" : "abc",
        "salary" : "123K"
    }
]
```

Unit Tests

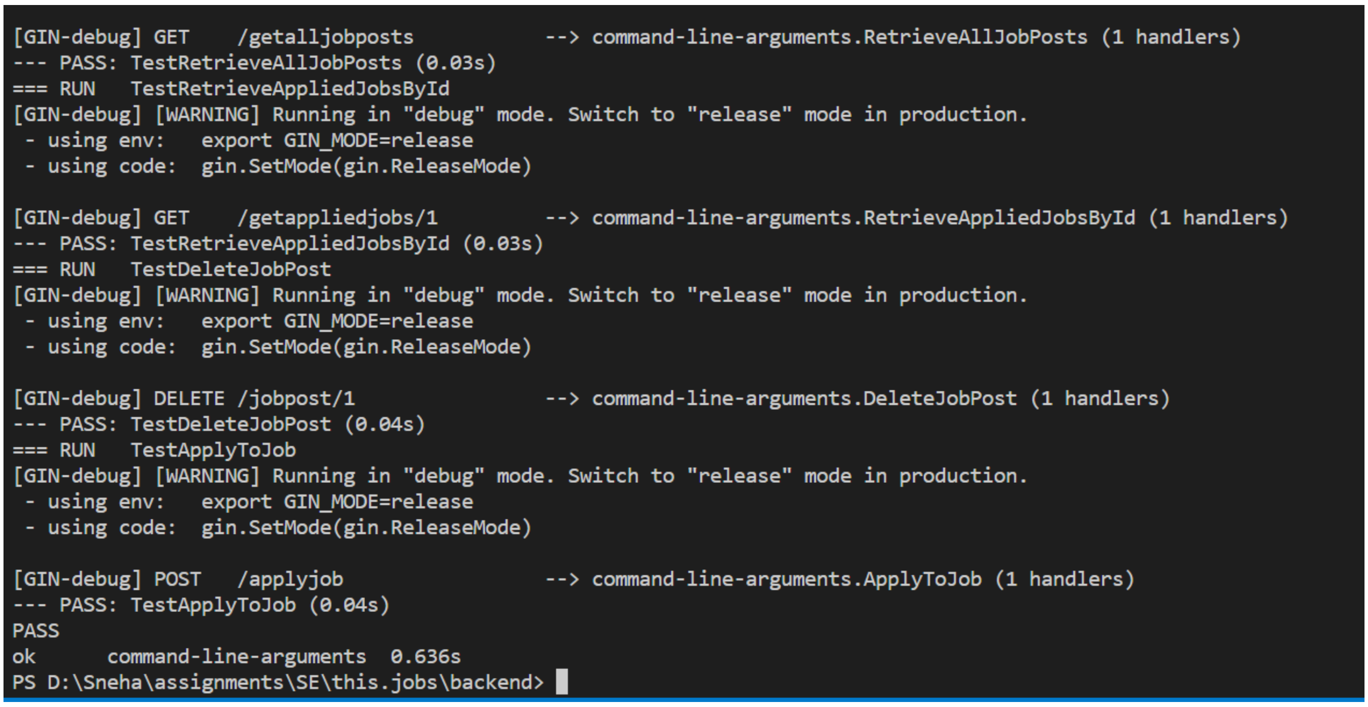
A mock database is created and unit tests are performed on the data from mock DB. The below sections show the unit testing output along with their coverage

Create, Retrieve and Apply Jobs

Test cases include job creation, updation and deletion with valid and invalid details and also applying to the job.



```
PS D:\Sneha\assignments\SE\this.jobs\backend> go test controllers/jobposts-controller.go controllers/jobposts-controller_test.go -v
=== RUN   TestCreateJobPost
[GIN-debug] [WARNING] Running in "debug" mode. Switch to "release" mode in production.
 - using env:   export GIN_MODE=release
 - using code:  gin.SetMode(gin.ReleaseMode)

[GIN-debug] POST   /jobpost                  --> command-line-arguments.CreateJobPost (1 handlers)
--- PASS: TestCreateJobPost (0.13s)
=== RUN   TestUpdateJobPost
[GIN-debug] [WARNING] Running in "debug" mode. Switch to "release" mode in production.
 - using env:   export GIN_MODE=release
 - using code:  gin.SetMode(gin.ReleaseMode)

[GIN-debug] PUT    /jobpost                  --> command-line-arguments.UpdateJobPost (1 handlers)
--- PASS: TestUpdateJobPost (0.07s)
=== RUN   TestIsUserPresent
--- PASS: TestIsUserPresent (0.07s)
=== RUN   TestIsJobPostPresent
--- PASS: TestIsJobPostPresent (0.03s)
=== RUN   TestIsAlreadyApplied
false--- PASS: TestIsAlreadyApplied (0.04s)
=== RUN   TestRetrieveAllJobPostsById
[GIN-debug] [WARNING] Running in "debug" mode. Switch to "release" mode in production.
 - using env:   export GIN_MODE=release
 - using code:  gin.SetMode(gin.ReleaseMode)

[GIN-debug] GET    /getjobposts/1            --> command-line-arguments.RetrieveAllJobPostsById (1 handlers)
--- PASS: TestRetrieveAllJobPostsById (0.04s)
=== RUN   TestRetrieveAllJobPosts
[GIN-debug] [WARNING] Running in "debug" mode. Switch to "release" mode in production.
 - using env:   export GIN_MODE=release
 - using code:  gin.SetMode(gin.ReleaseMode)

[GIN-debug] GET    /getalljobposts           --> command-line-arguments.RetrieveAllJobPosts (1 handlers)
--- PASS: TestRetrieveAllJobPosts (0.03s)
=== RUN   TestRetrieveAppliedJobsById
```



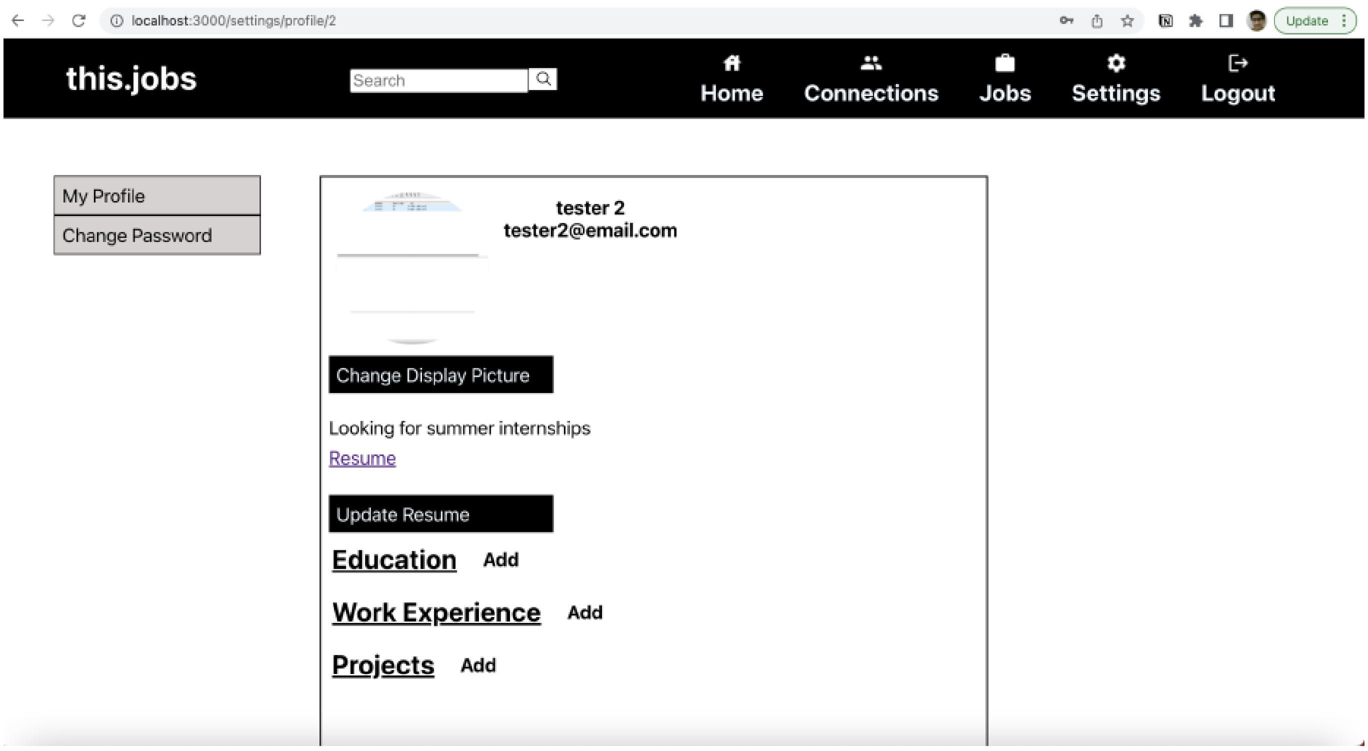
```
[GIN-debug] GET    /getalljobposts           --> command-line-arguments.RetrieveAllJobPosts (1 handlers)
--- PASS: TestRetrieveAllJobPosts (0.03s)
=== RUN   TestRetrieveAppliedJobsById
[GIN-debug] [WARNING] Running in "debug" mode. Switch to "release" mode in production.
 - using env:   export GIN_MODE=release
 - using code:  gin.SetMode(gin.ReleaseMode)

[GIN-debug] GET    /getappliedjobs/1         --> command-line-arguments.RetrieveAppliedJobsById (1 handlers)
--- PASS: TestRetrieveAppliedJobsById (0.03s)
=== RUN   TestDeleteJobPost
[GIN-debug] [WARNING] Running in "debug" mode. Switch to "release" mode in production.
 - using env:   export GIN_MODE=release
 - using code:  gin.SetMode(gin.ReleaseMode)

[GIN-debug] DELETE /jobpost/1                --> command-line-arguments.DeleteJobPost (1 handlers)
--- PASS: TestDeleteJobPost (0.04s)
=== RUN   TestApplyToJob
[GIN-debug] [WARNING] Running in "debug" mode. Switch to "release" mode in production.
 - using env:   export GIN_MODE=release
 - using code:  gin.SetMode(gin.ReleaseMode)

[GIN-debug] POST   /applyjob                 --> command-line-arguments.ApplyToJob (1 handlers)
--- PASS: TestApplyToJob (0.04s)
PASS
ok      command-line-arguments  0.636s
PS D:\Sneha\assignments\SE\this.jobs\backend>
```

Frontend accomplishments

Goals achieved:

> MyProfile:

1. Resume Upload and checking.
2. Other user's profiles are viewable by other users.



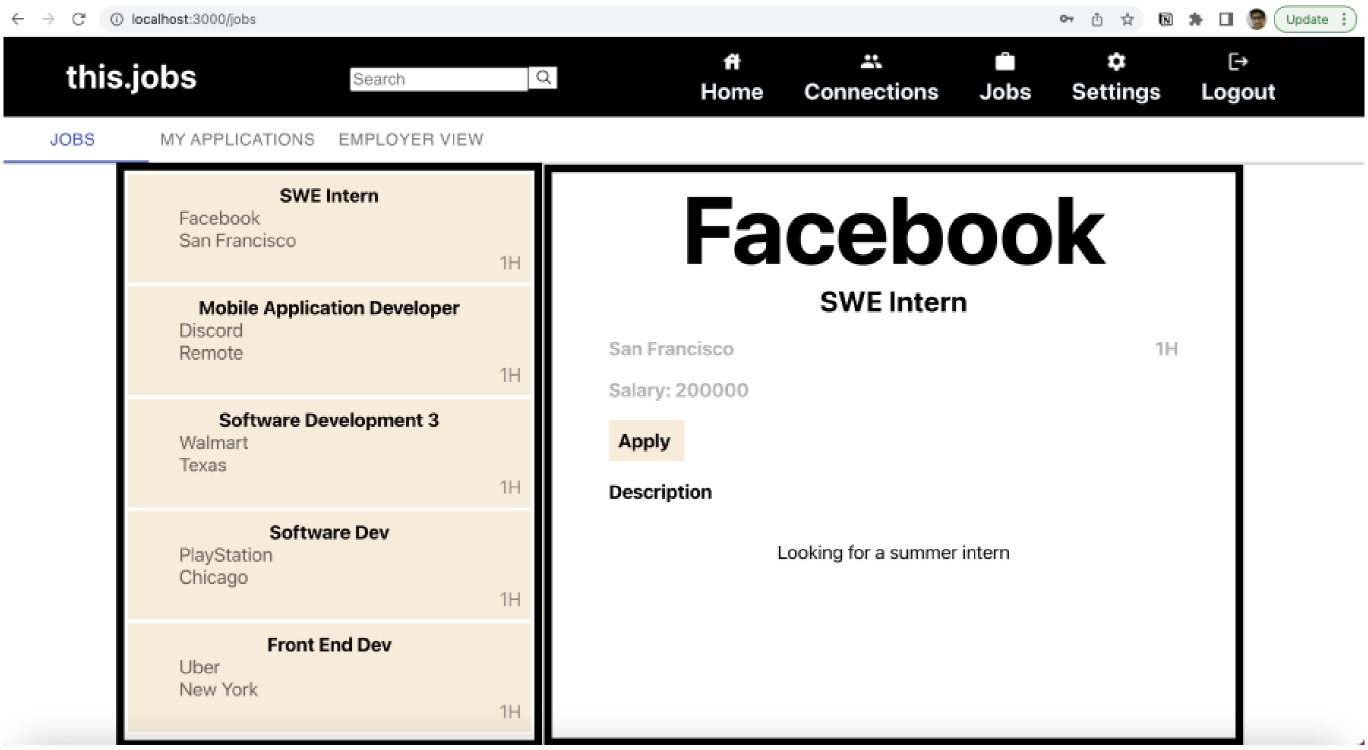
tester 2
tester2@email.com

Looking for summer internships
Resume

Education
Work Experience
Projects

> Jobs: Upon clicking on the jobs icon in the navigation bar, the user is directed to 3 links -

1. Jobs: Where the users can check out all the jobs and apply to whichever they want. They can see the details and descriptions of all the jobs on click.
2. My Applications: Displays all the applications that the user has applied to
3. Employer View: This gives the user the facility to post any jobs that they wish to and the list of all the jobs posted will be displayed in the same page.



SWE Intern
Facebook
San Francisco
1H

Mobile Application Developer
Discord
Remote
1H

Software Development 3
Walmart
Texas
1H

Software Dev
PlayStation
Chicago
1H

Front End Dev
Uber
New York
1H

Facebook

SWE Intern

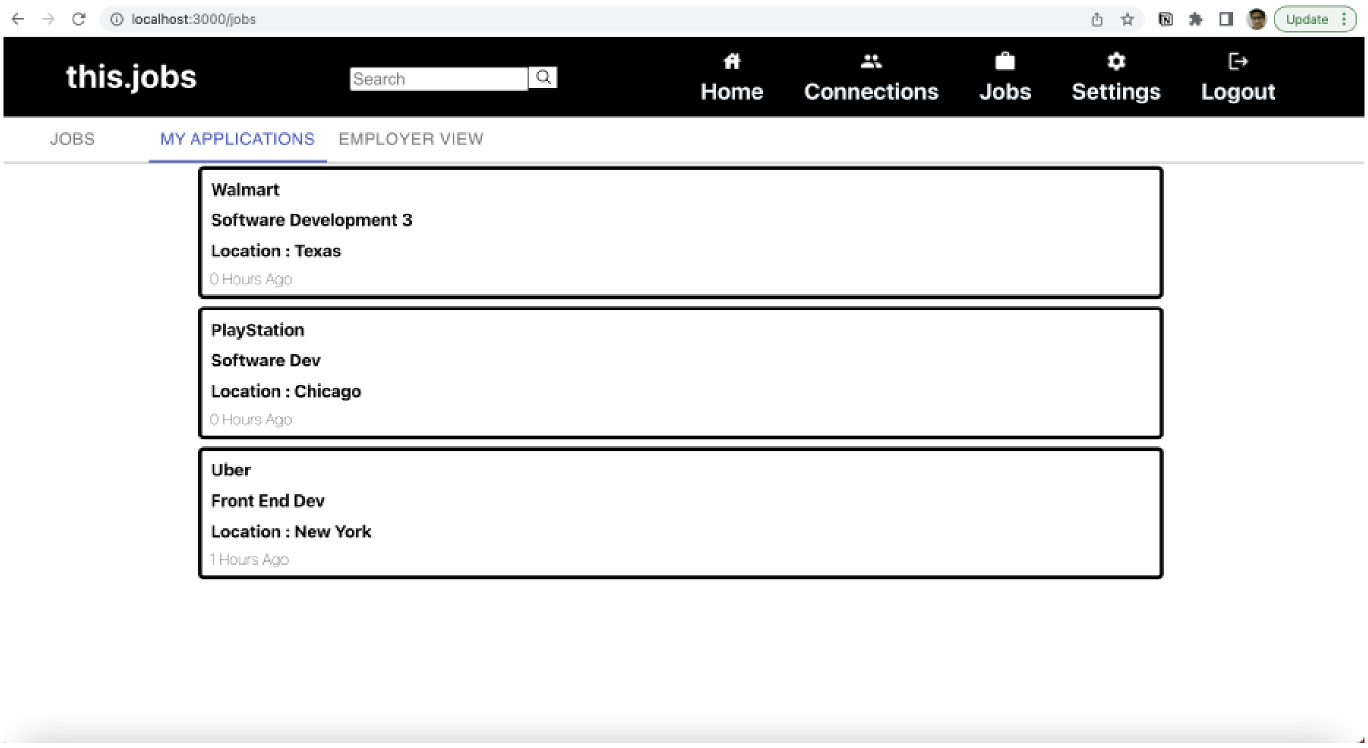
San Francisco 1H

Salary: 200000

Apply

Description

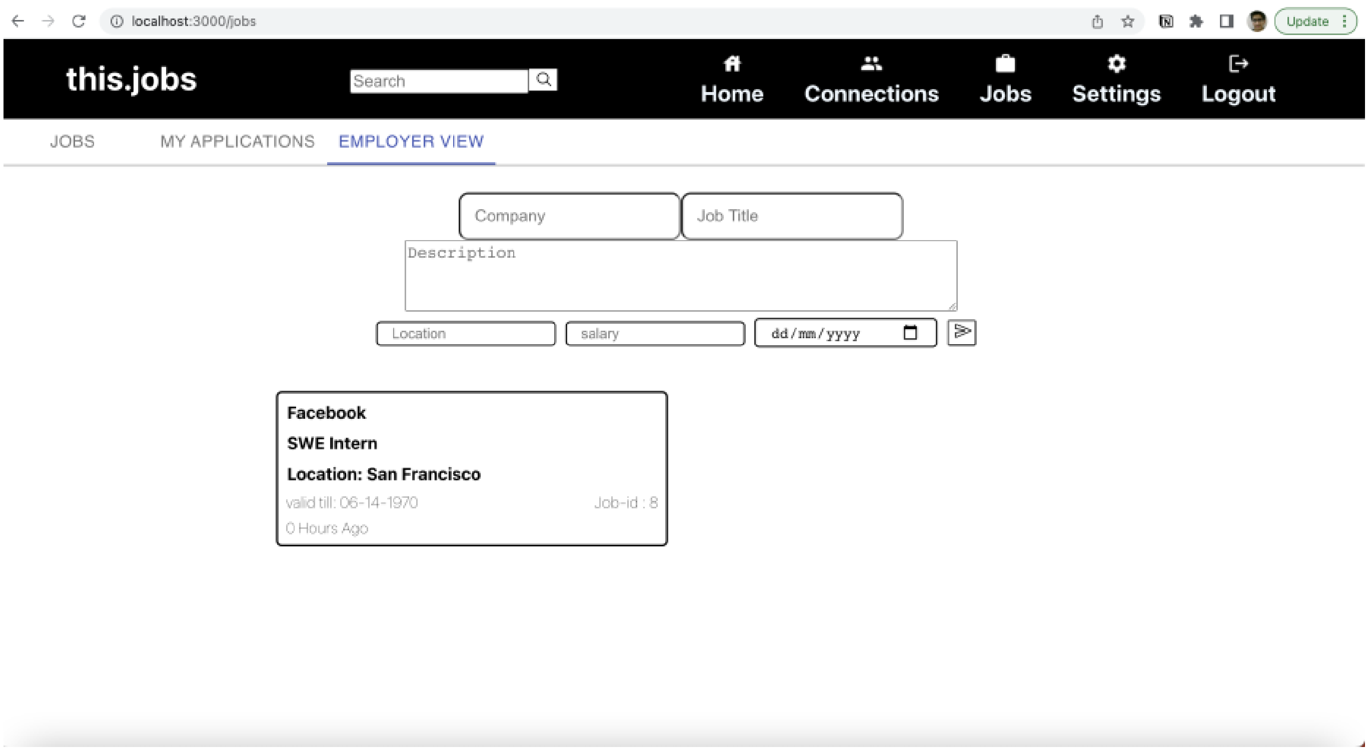
Looking for a summer intern



Walmart
Software Development 3
Location : Texas
0 Hours Ago

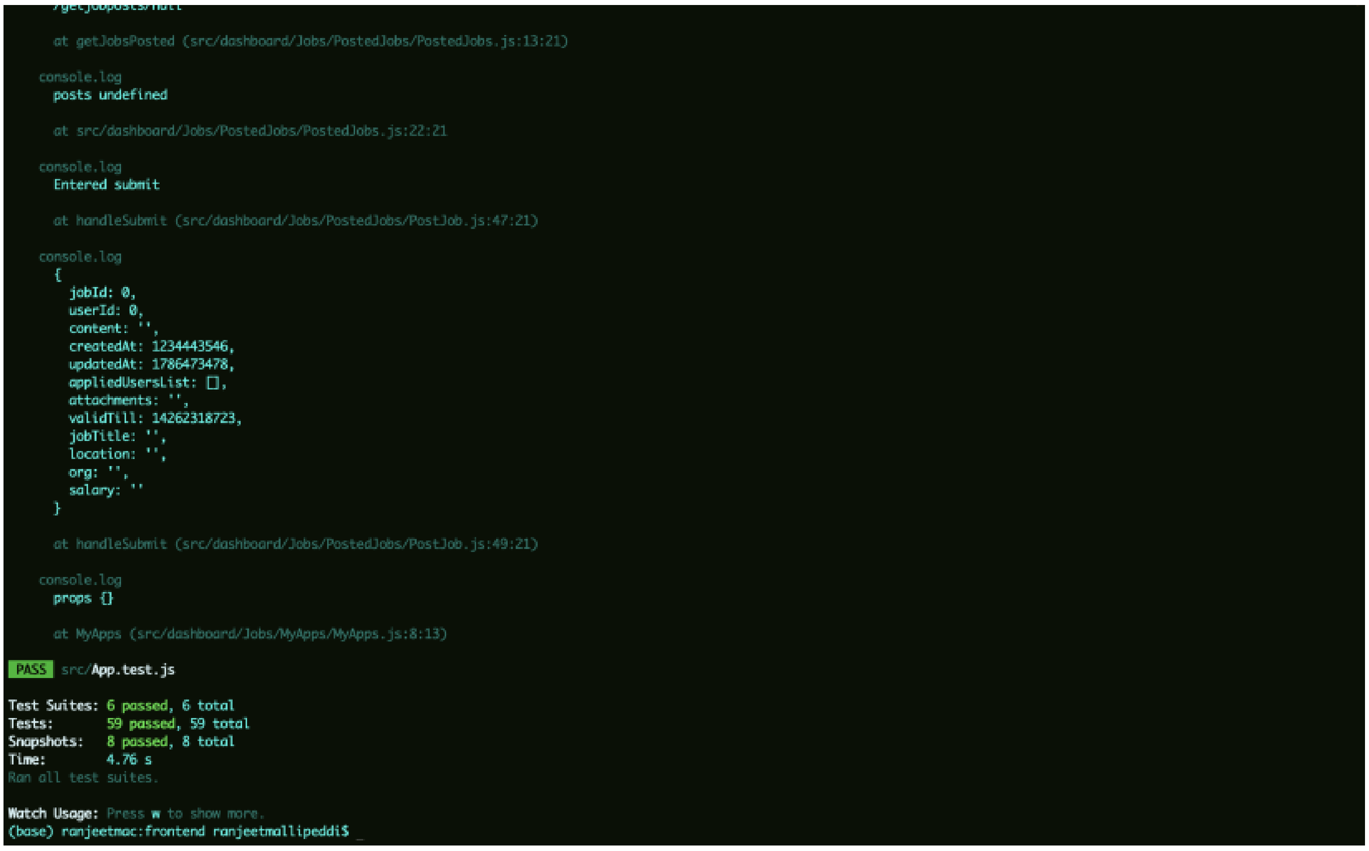
PlayStation
Software Dev
Location : Chicago
0 Hours Ago

Uber
Front End Dev
Location : New York
1 Hours Ago



Facebook
SWE Intern
Location: San Francisco
valid till: 06-14-1970 Job-id : 8
0 Hours Ago

> Unit Testing: For all the created components for this sprint.



```
    /getjobposts/null

    at getJobsPosted (src/dashboard/Jobs/PostedJobs/PostedJobs.js:13:21)

  console.log
    posts undefined

    at src/dashboard/Jobs/PostedJobs/PostedJobs.js:22:21

  console.log
    Entered submit

    at handleSubmit (src/dashboard/Jobs/PostedJobs/PostJob.js:47:21)

  console.log
    {
      jobId: 0,
      userId: 0,
      content: '',
      createdAt: 1234443546,
      updatedAt: 1786473478,
      appliedUsersList: [],
      attachments: '',
      validTill: 14262318723,
      jobTitle: '',
      location: '',
      org: '',
      salary: ''
    }

    at handleSubmit (src/dashboard/Jobs/PostedJobs/PostJob.js:49:21)

  console.log
    props {}

    at MyApps (src/dashboard/Jobs/MyApps/MyApps.js:8:13)

 PASS  src/App.test.js

Test Suites: 6 passed, 6 total
Tests:       59 passed, 59 total
Snapshots:   8 passed, 8 total
Time:        4.76 s
Ran all test suites.

Watch Usage: Press w to show more.
(base) ranjeetmac:frontend ranjeetmallipeddi$ _
```