Hello Geek!

You are receiving this assignment because we believe in your profile. We are seeking someone who has taken a path different than their counterparts, who has done things that are different. Here at our company, even the CEO codes, and we're seeking the same passion in you.

Does that sound, like we're talking about you? But before you get in - you would need to prove yourself, everyone has to! Your selection, if you can join the league will depend upon this particular assignment of yours.

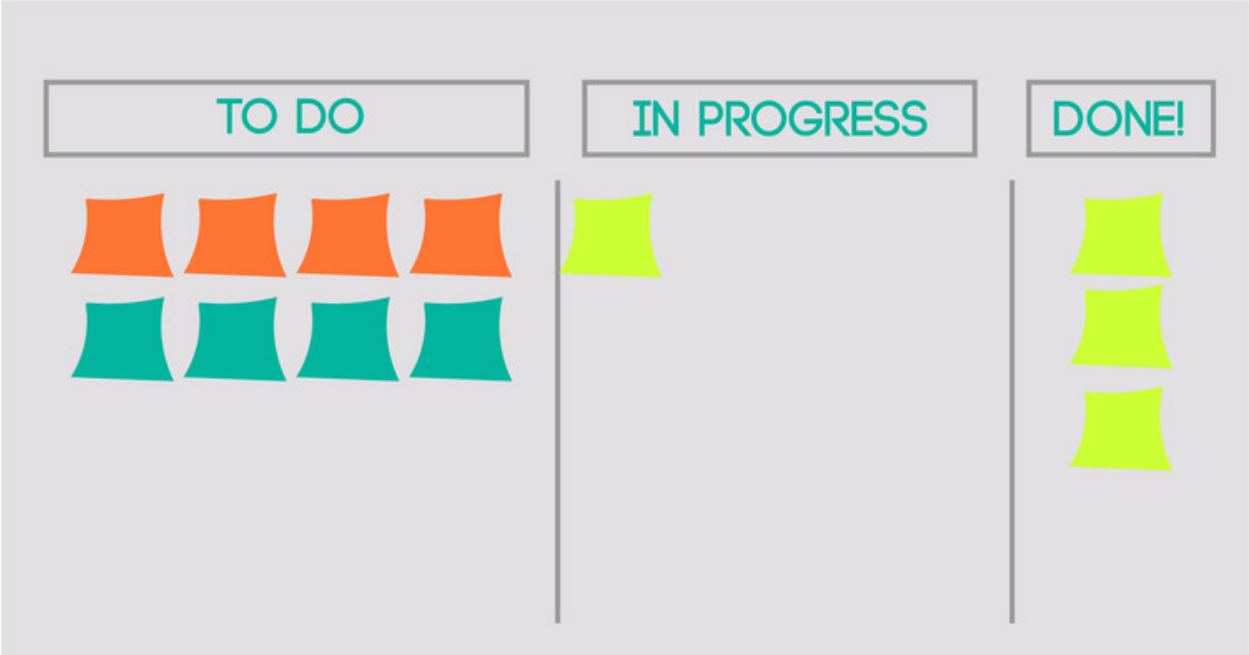
As you've applied for a Web Development Internship, we're sure, you'd be well versed with technologies like - ReactJs, and Node.js.

There are two coding assignments - one on React.js and another on Node.js. PICK ONE you're most comfortable with.

OVER TO THE NEXT PAGE

Frontend Assignment - For Geeks mastered on React!!!

You need to create a Work Board. Similar to the one down below:



What is a Work Board? A work board is a software, which has tracks your work status. In general, there are three categories - Todo, In Progress, and Done.

Detailed Features:

- **Adding a Task:** Right-click on the ToDo pane will open a menu option with - “New Task”. On click of New Task, open a modal (popup) and ask for Title (Text Field) and Description (Text Area) and a button - “Add Task”. On clicking the “Add Task” button, the modal should close and the task should be displayed in the Todo Pane in form of a sticky note.
  *** A task can only be added from the Todo Pane. Right-Clicking in the InProgress and Done should not show anything.*
- **Menu Options on Sticky Notes:** If right-clicked on any sticky notes, it should give open a menu option with the following options -
  - Send to >
    - Todo (disabled if already on Todo)
    - In Progress (disabled if already on In-Progress)
    - Done (disabled if already on Done)
  - Delete
  - Archive
- **Deleting a Task** - Right-click on the Sticky note to be deleted and click on the Delete button. This should ask a question - Are you sure you want to delete the task? On confirmation, this should be deleted.

- **Search -** On the top of the page, there should be a search bar (text field). On typing anything in the search bar there should be a search operation performed on the sticky notes based on the title. And only the notes, with a matching title, should be displayed in each of the panes - Todo, In Progress and Done.
- **Moving the Sticky Notes Between the Pane** - Right-click on any of the Sticky notes, and there should be an option for Move To… this should open the 2nd level of menu options - with the panes to send the sticky note to, as discussed under *"Menu Options on Sticky notes".*

Bonus

- **Archiving:** Any sticky note can be archived. On archiving a sticky note it should not be displayed on any of the panes but should remain in the data list. Imagine it to be hidden. The purpose of archiving could be - *"We just created it, but don't need this for now, and maybe would need it in the future."*
- **View archived tasks:** There should be an option on the top of the page, next to the search bar - Hide/Unhide Archives. If the archive is in the hidden stage, all the archive sticky notes should not be visible. If the archive is marked as unhide, then the archived sticky notes should appear in their respective panes.
- **Search on Archive:** The search should check for archive option before displaying any sticky notes.

How will you be evaluated on this Assignment?

- **Code Structure - 30 points**
  We expect you to keep your code clean and distributed, don't keep all the code in just one to two files, give time and structure your code correctly so that you're able to add new features without changing a lot of code. The code should also be clean. You should make sure that your code has a proper indentation. ***A code without a proper indentation will be rejected.***
- **Add README File - 10 points**
  ReadME file is a must to have. If you don't put a README file, we cannot run your code.
- **Right Data Structure and Optimized Code - 40 points**
  This is typically a Data Structure problem, and hence we look forward to how you manage your data structure, is your code good enough to handle big-sized data sets? Are your loops optimized enough?
- **Look and Feel: CSS - 10 Points**
  Being a web developer, we don't expect you to give a plain and simple-looking website. We will admire it if you can put some styling to your application. Looks matter a lot on the web. You can feel free to use any CSS frameworks.
- **Use of SCSS - 5 points**
  The best practice to write a CSS is using SCSS. If you can attempt to write SCSS, and later explain it during our face-to-face interview, you do have some moves which we look forward to!

- **Use of Webpack - 5 points**
  Frontend development is incomplete without Webpack. If you know webpack, then do apply it else don't waste your time.

How to submit your code -

Push your code on your Git account and send your link to us.

What not to do?

Code will be checked for plagiarism. Please ensure that you are being honest and try your best!

Even if you're able to pass the first check, our next round will be discussing your code and asking you to modify a small feature on call. So please avoid blind copy-paste.

What are the Next Steps?

- Submit your code
- Your code will be checked and verified.
- You will receive an interview invite for 1.5 hours.
- A comprehensive study will be done comparing different parameters with other candidates. And you will be notified.
- If all goes well, you're hired!

BACKEND GEEKS - OVER TO THE NEXT PAGE
FAQs towards the last.

Backend Assignment - For Geeky Backend Coders!

If you're here, that means you're a Backend Geek, and might not want to struggle in building an outstanding web application, but is more interested in building what runs it securely.

You need to write role-based backend APIs for School Management Software. In this school management software, there are three roles and these are the functionalities they need to perform -

- **Admin:** As an admin, a user can
    - Manage Classes - Create, Update, Delete Classes
    - Manage Teachers -
        - Add a Teacher's Account
        - Map Teacher to existing Class
        - Get list of teachers
    - Manage Student -
        - Add Student
        - Map Student to existing class
        - Get List of Students
- **Teacher:** As a teacher, a user can
    - Get list of the students (sorted on name)
    - Create Score Card for a student. A score card will have these feilds
        - Subject Name
        - Date of Exam
        - Date of Score Card (Default to today)
        - Score (out of 100)
        - Comments
    - Get student ranking based on the % of all the subjects. Let's say, Student A has multiple subjects - English, Maths, Science and each of the score is, 70, 40, 65 respectively, the total sum of the score for the student is 70+40+65 = 175/300, hence the student scored 58%. Now there is another student Student B which has two subjects - English, Social Science and has scored 60 and 80 respectively, which gives a total % of 70% (60+80 = 140/200 for two subjects). Hence in the ranking sheet, the Student B comes above Student A.
- **Students:** View their own score card for all the subjects.

You can use any database (MySQL, Postgres, MongoDB, etc…) of your choice. You need to write APIs for -

The flow should start with Login - Pass email id and password and get login token.
*** for admin you should add a data for login details manually in the database.*
*** for teacher and student the admin should create login credentials.*

If logged-in as a Admin, one should not able to hit any other APIs. In case, someone calls a different API it should return 401.

Similarly, as a Teacher, only the APIs assigned for the teacher should be allowed. And same for Sudent.

For your ease of thinking here are some of the API contracts, which contains, what API endpoints you should create and what data you should pass, and receive:

POST `/auth/login` - For Login

```
Request Params:
    email_id: STRING
    password: STRING
Response:
    {
        id: NUMBER, //id of the user
        auth_token: STRING, //auth token for further API
    calls
        name: STRING,
        role: STRING,
        email: STRING,
    }
```

POST `/admin/teacher` - Add Teacher

```
Request Header:
    auth_token: STRING
Request Params:
    name: STRING,
    email_id: STRING,
    password: STRING
Response:
    {
        id: NUMBER, //id of the teacher
        message: STRING //teacher added
    }
```

DELETE `/admin/teacher/:teacherId` - delete the teacher with id.

```
Request Header:
    auth_token: STRING // to validate admin access
Response:
    {
        id: NUMBER, //id of the teacher removed.
        message: STRING //teacher removed
    }
```

POST `/admin/class/` - Add a class

```
Request Header:
      auth_token: STRING //to validate admin access
Request Params:
      className: STRING //e.g. 10A, 10B etc…
Response:
      classId: NUMBER //id of the created class
      message: Sting //message like: class created.
```

POST `/admin/teacher/:teacherId/class/:classId` - Map teacher to class

```
Request Header:
      auth_token: STRING //to validate admin access
Response:
      message: Sting //message like: teacher added
```

Rest of the APIs we leave it to you, to predict and build it.

Bonus

- Add a swagger file to share contract
- Add a postman file to test your code.

How will you be evaluated on this Assignment?

- **Code Structure - 30 points**
  We expect you to keep your code clean and distributed, don't keep all the code in just one to two files, give time and structure your code correctly so that you're able to add new features without changing a lot of code. The code should also be clean. You should make sure that your code has a proper indentation. ***A code without a proper indentation will be rejected.***
- **Add README File - 10 points**
  ReadME file is a must to have. If you don't put a README file, we cannot run your code.
- **Right Database Structure - 30 points**
  We expect that you would be using a database of your own choice. Hence make sure that your database structure is clean.
- **Proper use HTTP Status Code - 20 Points**
  We expect you to properly use HTTP Status codes, this adds confidence to your NodeJs APIs.
- **Correct Endpoints - 10 points**
  We expect you to properly structure your API endpoints, read our example of endpoints, and think around that.

How to submit your code -

Push your code on your Git account and send your link to us.

What not to do?

Code will be checked for plagiarism. Please ensure that you are being honest and try your best!

Even if you're able to pass the first check, our next round will be discussing your code and asking you to modify a small feature on call. So please avoid blind copy-paste.

What are the Next Steps?

- Submit your code
- Your code will be checked and verified.
- You will receive an interview invite for 1.5 hours.
- A comprehensive study will be done comparing different parameters with other candidates. And you will be notified.
- If all goes well, you're hired!

TURN AROUND FOR FAQs.

FAQ

What are your chances of getting hired?

We receive more than 1000 applications from different portals. We are looking for the top 1% of candidates from that pool.

What is the timeline for the process?

This process will take 2 weeks of time. We want people to join from 1st July 2022. Hence we will close this by that time.

What if you have any doubts?

The requirement is a bit detailed, and if you've doubts, it will be actually difficult to answer or get on a call. You can make assumptions because you'll receive a similar set of requirements or even a limited one than this while you're actually working in the organization. You must apply your innovative mind.

Why is this Coding Challenge so difficult?

To be honest, if you're a real developer, or want to be one, then these assignments are not difficult. It should take you less than 2 days. We do understand, as an intern, there could be things which would be out of your knowledge, and we want hustlers, who never gives up! And that is the reason why we are giving this coding challenge to you!