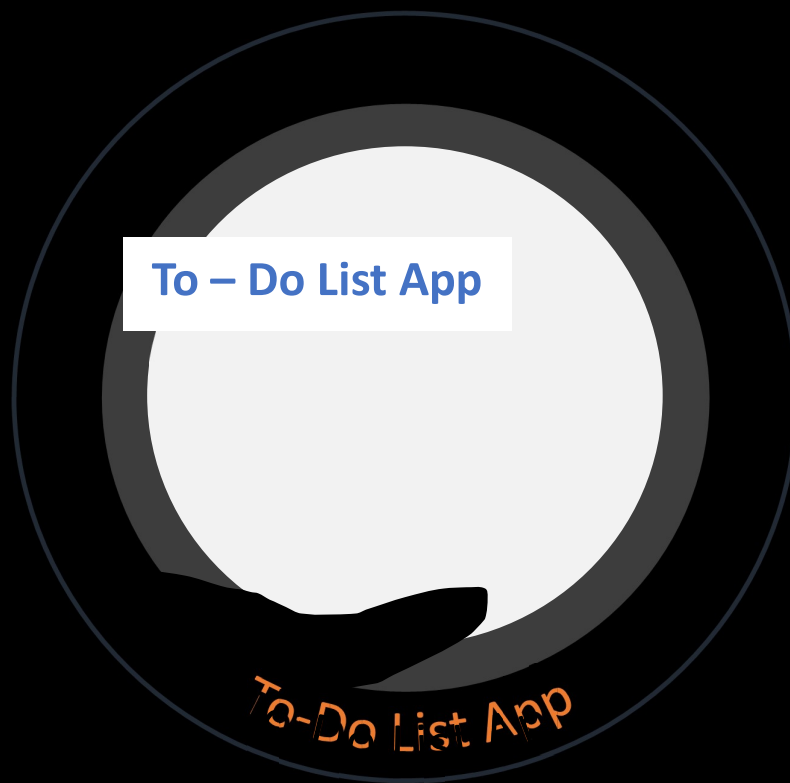




MEAN Stack Assessment Document





Contents

MEAN Stack	2
Best Practices, Tools, and Technologies	3
Declarations.....	3
Naming Conventions	3
Use of Version Control.....	3
Documentation Using Postman.....	4
Unit Tests.....	4
UI Unit Tests	4
Other Tools And Technologies To Use	4
Assessment – To-Do List Application.....	5
Problem Statement	5
Set-up Requirements.....	5
Functionality.....	5
Wireframes.....	6

MEAN Stack

The MEAN stack is a JavaScript-based framework for developing scalable web applications. The term MEAN is an acronym for **M**ongoDB, **E**xpress, **A**ngular, and **N**ode — the four key technologies that make up the layers of the technology stack.

MongoDB - A NoSQL, object-oriented database designed for use with cloud applications.

Express(.js) - A web application framework for Node(.js) that supports interactions between the front end (e.g., the client side) and the database.

Angular - Often referred to as the “front end”; a client-side JavaScript framework used to create dynamic web applications to work with interactive user interfaces.

Node(.js) - The premier JavaScript web server used to build scalable network applications.

The assessment is to be implemented using MEAN stack technologies.





Best Practices, Tools, and Technologies

While implementing the assessment make sure that the best practices are followed. Best practices ensure quality end-product with enhanced code readability that helps easy code maintenance. When best practices are followed one can ensure timely delivery of services, products, and support.

Declarations

All the declarations at the beginning of program ○ Single place to look for

declarations ○ Reduced possibility of unwanted re-declarations

Naming Conventions

Naming conventions are fundamental as it helps to maintain code consistency that makes the code readable and easy to understand. While developing the application follow the standard conventions as mentioned below.

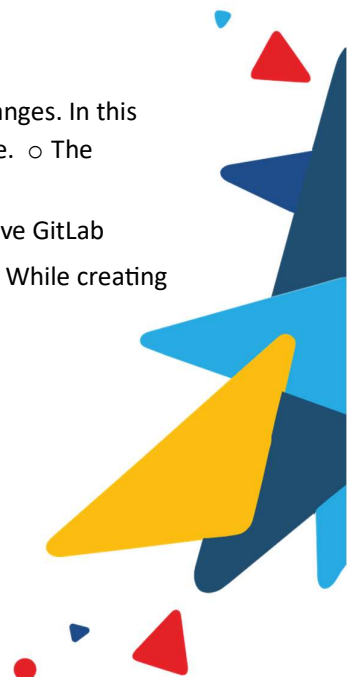
- camelCase is used for variable & function names (var lastName) ○ camelCase is used for filenames ○ Modules, variable names always start with a letter (var firstName) ○ Give spaces around operator (var sum = no1 + no2) ○ Encourage code readability by using appropriate indentation ○ Global variables & constants should be written in UPPERCASE ○ Use proper comments (//) to enhance code readability ○ Choose meaningful names while naming file, folders, variables, methods
- Use conventional (Angular) suffix for files - .component.ts, .directive.ts, .module.ts, .pipe.ts or .service.ts
- For further naming conventions refer ○ <https://angular.io/guide/styleguide>


Use of Version Control

- Version control helps in reverting back to changes in-case of accidental changes. In this assessment make use of version control in order to store, manage the code. ○ The platform which would be used for version control would be @GitLab. GitLab is similar to GitHub that offers more features. ○ Ensure that you have GitLab account. Further create a GitLab repo that has access to required members. ○ While creating a GitLab repo, ensure naming convention as follows,

“ng-training-”(followed by assignment name)

Example: ng-training-assignment -1 ○ Make sure all the changes are committed to the repo.



- 
- **Once assessment implementation is complete do share the GitLab link on the company's official mail-id (through which you have received the assessment details).**
 - Implement the Folder Structure systematically.
 - Follow Modular approach.
 - Avoid one single file having more than 600 lines of code.
 - Create Separate Folders/Files.
 - Encourage loosely coupled approach.
 - Throughout the whole application follow a Consistent Pattern.
 - File/Folders Naming.
 - Components/Modules/Directives/Services Naming.

Documentation Using Postman

Follow below link to understand the documentation process using Postman. It is mandatory to maintain complete documentation for the assessment. **Once assessment implementation is complete do share the documentation link on the company's official mail-id (through which you have received the assessment details).**

<https://learning.postman.com/docs/publishing-your-api/api-documentation-overview/>

Note: Postman documentation is optional. However, candidates who are able to, should complete it.

Unit Tests

It is an integral and mandatory process during assessment development that essential unit tests are written and verified. Do follow the link below for unit test reference. Use Mocha for writing the unit tests.

<https://mochajs.org/#:~:text=Mocha%20is%20a%20feature%2Drich,to%20the%20correct%20test%20cases.>

UI Unit Tests

It is an integral and mandatory process during assessment development that essential user interface unit tests are written and verified. Do follow the link below for UI unit test reference.

<https://docs.angular.lat/guide/testing>

Other Tools And Technologies To Use

- In assessment implementation use **SLDS** and CSS libraries. Refer below for the same.
 - <https://www.npmjs.com/package/@salesforce-ux/design-system>
 - <https://www.lightningdesignsystem.com/>

Note: If you're not familiar with this library, you can use other libraries, but having experience with SLDS is beneficial.





Assessment – To-Do List Application

Problem Statement


Develop a To-Do list application where users can create, read, update, delete the tasks user want to maintain on To-do work list.

Set-up Requirements

- Server-side
 - Server
To use Node.js with Express.js to set up the server. □ Database
Use MongoDB to store the to-do list.
 - APIs
GET /api/tasks - Retrieve all tasks.
POST /api/task - Create a new task.
PUT /api/task/:id - Update a task by ID.
DELETE /api/task/:id - Delete a task by ID.
- Client-side And UI
 - Framework
Use Angular to build the frontend.
 - Components
 1. Task List Component - displays the list of tasks.
 2. Task Form Component - A form to create and update tasks.
 - Services
 1. Task Service - Handle API calls to the backend.

Functionality

- Process and Tasks
 1. Add task - allow users to add a new task .
 2. View task – Display and navigate the task list.
 3. Edit task – allow user to update existing task.
 4. Delete task – allow user to delete task.
 - UI/UX

UI should be user friendly and easy to navigate.
Ensure responsive UI design for both desktop and mobile views.
- 

- Unit and UI Testing
For server-side use Mocha for writing the unit tests.
Write the appropriate test cases and test the implementation accordingly.
For client-side use Angular default UI unit tests libraries.

Wireframes

Please find below the assessment wireframes. Note and follow the minute UI details as mentioned in the wireframes. Implement the UI accordingly. **Screen 1**

<div>Tasks</div> <div>All Tasks</div> <div>4 records</div>					<div>New Task</div> <div>Refresh</div> <div>Search</div>
<input type="checkbox"/>	Assigned To	Status	Due Date	Priority	Comments
<input type="checkbox"/>	User 1	Completed	12/10/2024	Low	This task is good
<input type="checkbox"/>	User 2	In Progress	14/09/2024	High	This
<input type="checkbox"/>	User 3	Not Started	18/08/2024	Low	This
<input type="checkbox"/>	User 4	In Progress	12/06/2024	Normal	This task is good

20

First

Prev

1

Next

Last

Screen 2

Tasks

New Task

Refresh

Assigned To

User 1

Status

Not Started

Due Date

12 May 2016

Priority


Normal

Description

Cancel

Save

Screen 3



Tasks New Task Refresh

Edit Task

*Assigned To: User 1

*Status: Not Started


Due Date: 12 May 2016

*Priority: Normal

Description:

Cancel Save

Screen 4



Tasks New Task Refresh

All Tasks

4 records

Search

	Assigned To	Status	Due Date	Priority	Comments
<input type="checkbox"/>					
<input type="checkbox"/>					
<input type="checkbox"/>					
<input type="checkbox"/>					
<input type="checkbox"/>					

Delete

Do you want to delete task{{task.name}}?

No Yes

20

First Prev 1 Next Last