|  |  |
| --- | --- |
| Single Page Application on Task manager  MEAN FSD 8 Hour Code Capsule | Abstract  The Task Manager solution should be a Single Page Application (SPA) to keep track of upcoming tasks, finished tasks and their respective priorities.  Developer  MEAN Full Stack Developer |

Contents

[Important Instructions 2](#_Toc496274541)

[Business-Requirement: An Overview 2](#_Toc496274542)

[Technical Spec – Solution Development Environment 3](#_Toc496274543)

[Front End Layer 3](#_Toc496274544)

[Middle Tier Layer 3](#_Toc496274545)

[Database & Integration Layer 3](#_Toc496274546)

[Ancillary Layer 3](#_Toc496274547)

[Deployment & Infrastructure 3](#_Toc496274548)

[Editors 3](#_Toc496274549)

[Wireframes 4](#_Toc496274550)

[Add Task 4](#_Toc496274551)

[View Task 4](#_Toc496274552)

[Update Task 5](#_Toc496274553)

[Database Tables 5](#_Toc496274554)

[Software Requirements 6](#_Toc496274555)

[Architecture 6](#_Toc496274556)

[Architecture Diagram for MEAN Enterprise Full Stack 7](#_Toc496274557)

[Important Instructions 8](#_Toc496274558)

[Assessment Deliverables 8](#_Toc496274559)

# Important Instructions

1. Adhere to (Follow) the design specifications mentioned in the case study.
2. Please make sure that your code does not have any compilation errors while submitting your case study solution.
3. The final solution should be a zipped code having solution. Solution code will be used to perform Static code evaluation.
4. Implement the code using best design standards.
5. Use Internationalization for all the labels and messages in Rest API Development.
6. Do not use System out statements or console.log for logging in Rest API and FrontEnd respectively. Use appropriate logging methods for logging statements/variable/return values.
7. If you are using Spring Restful or Jersey JAX-RS to develop Rest API, then use Maven to build the project and create WAR file.
8. If you are using Node and Express to develop Rest API, then use Grunt/Gulp/NPM to build/minify the project and create application for deployment.
9. Write web service which takes input and return required details from database.
10. Use JSON format to transfer the results.

# Business-Requirement: An Overview

**The Task Manager Single Page Application** allows you to manage task and set their priorities in range of 0 to 30. Each task will have parent task (optional/another task can be made parent), start date and end date. Below are the features of Task Manager:

1. Add/Edit/View/Delete Task
2. You can make one task a parent of another task.
3. On view task screen:
   1. User can search task by task name, parent task, start date, end date and priority.
   2. User can edit task
   3. User can end the task once it is finished. Once ended, user cannot edit the task. User can only view the task once finished.
4. **All search fields on view task screen are using auto-filter property of angularjs**.

# Technical Spec –Solution Development Environment

## Front End Layer

|  |  |
| --- | --- |
| **Framework(s)/SDK/Libraries** | **Version** |
| AngularJS | 1.5 or above |
| Bootstrap | 3.0 or above |
| CSS | 3 |
| HTML | 5 |
| JavaScript | **1.8 or above** |
| JQuery | **1.3** |

## Middle Tier Layer

|  |  |  |
| --- | --- | --- |
| **Technology** | **Framework(s)/SDK/Libraries** | **Version** |
| MEAN Stack | Node.JS | 6.x (Latest: 6.11.4) |
| Express.JS | 4.16.1 |

## Database & Integration Layer

|  |  |  |
| --- | --- | --- |
| **Technology** | **Framework(s)/SDK/Libraries** | **Version** |
| MEAN Stack | MongoDB/MySQL |  |
| NoSQL |  |
| Mongoose.JS | 4.12.1 |

## Ancillary Layer

|  |  |  |
| --- | --- | --- |
| **Technology** | **Framework(s)/SDK/Libraries** | **Version** |
| Source Code Management Tool | GIT | 2.14.2 |
| Build Tool/MEAN Stack | Grunt/Gulp | 1.0.1/3.9.1 |
| Dependency Management Tool/MEAN Stack | NPM | 4.x |
| Testing Tool/MEAN Stack | Jasmine |  |

## Deployment & Infrastructure

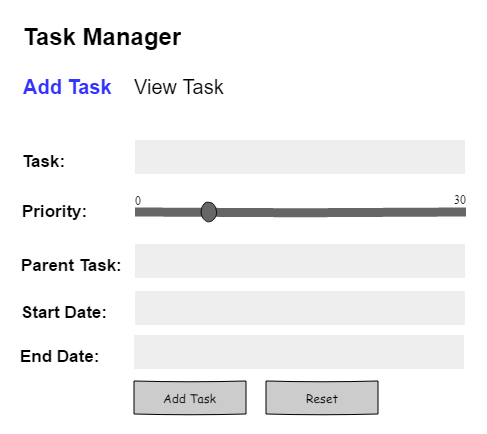
|  |  |  |
| --- | --- | --- |
| **Technology** | **Framework(s)/SDK/Libraries** | **Version** |
| Docker | - |  |
| Apache HTTP (XAMPP) | - |  |
| Node | - |  |
| Dependency Management Tool | NPM |  |

## Editors

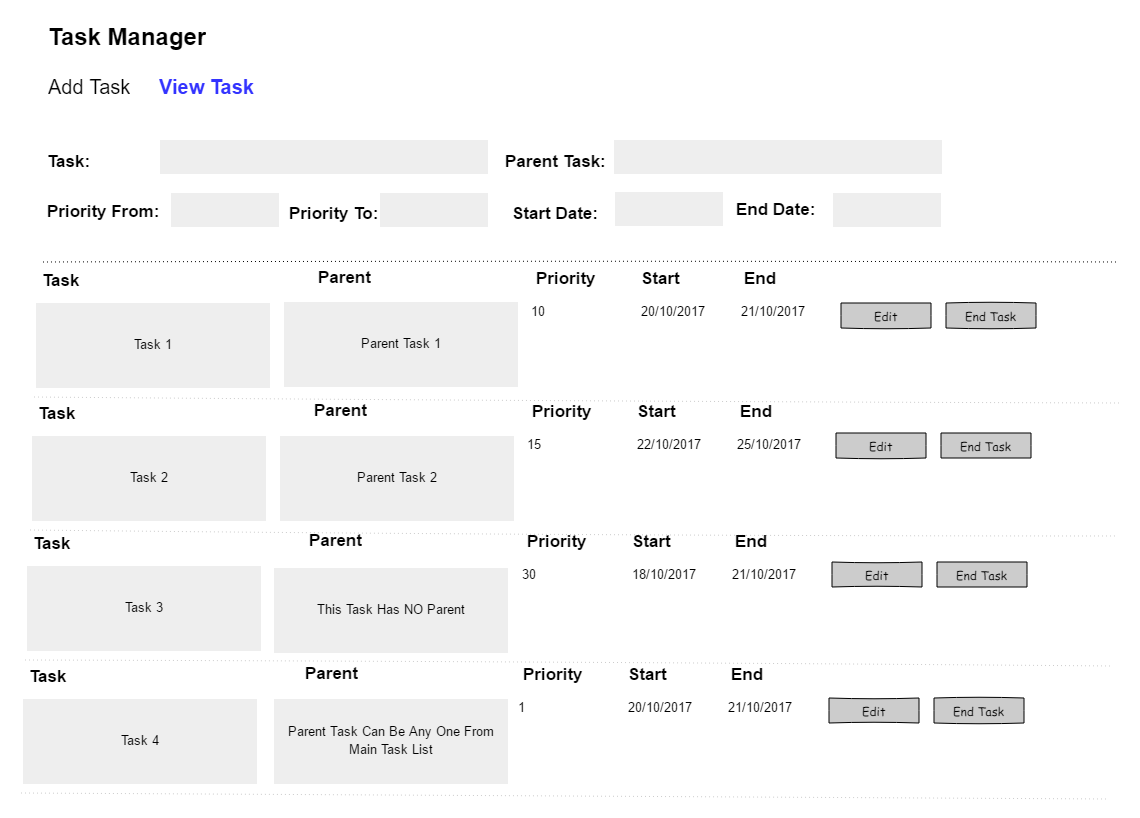
|  |  |
| --- | --- |
| **Name** | **Version** |
| Adobe Brackets |  |
| Sublime Text |  |
| Atom Editor |  |

# Wireframes

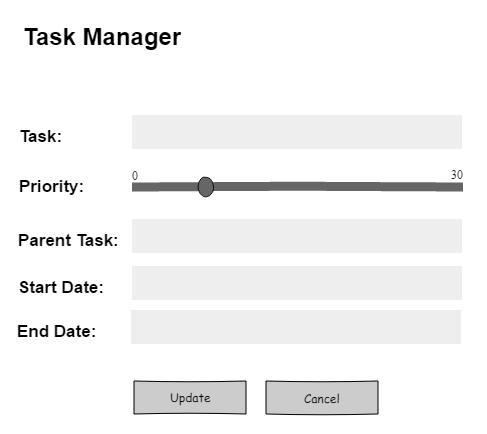
## Add Task



## View Task



## Update Task



# Database Tables

|  |  |  |  |
| --- | --- | --- | --- |
| **Parent Task Table** |  | **Task Table** |  |
|  |  |  |  |
| Parent\_ID |  | Task\_ID |  |
| Parent\_Task |  | Parent \_ID |  |
|  |  | Task |  |
|  |  | Start\_Date |  |
|  |  | End\_Date |  |
|  |  | Priority |  |
|  |  | Finished |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Software Requirements

This case study assumes knowledge of programming and hands-on with below mentioned skills.

The technologies included in Full Stack are not limited to following but may consist of:

* UI Layer (HTML5, CSS3, Bootstrap, JavaScript, Jquery, **AngularJS**, **JSP**)
* Middleware Restful API (Node, Express)
* Database Persistence ( MongooseJS)
* Database layer (MongoDB, NoSQL)
* Ancillary skills (GIT, Docker, Grunt/Gulp) etc.

To complete this case study, you should be comfortable with basic single page web application concepts including REST and CRUD. The environment setup is built into virtual environment you are logged in to.

**You may use angular-cli to create your template project.**

**Ref1: https://cli.angular.io/**

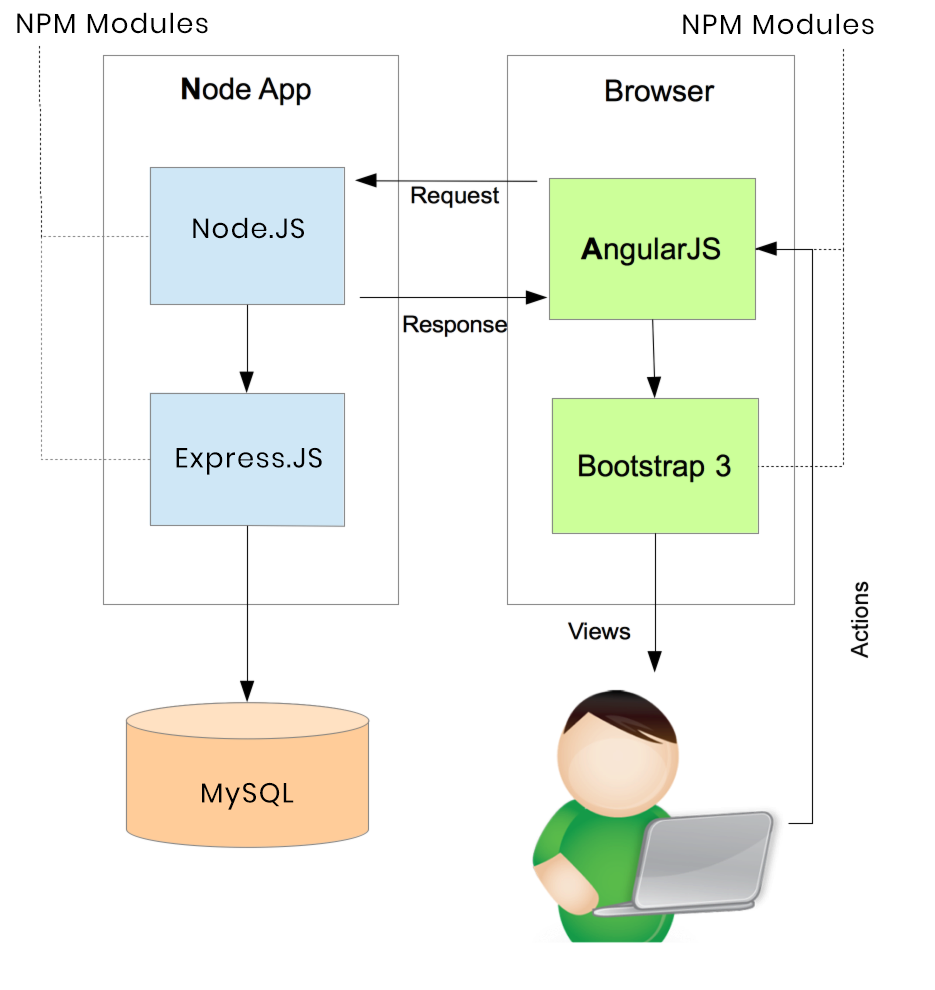
**Ref2: https://github.com/angular/angular-cli**

# Architecture

A physical architecture is an arrangement of physical elements, (system elements and physical interfaces) that provides the designed solution for a product, service, or enterprise. It is intended to satisfy logical architecture elements and system requirements. Workout Tracker follows a three layered architecture namely presentation layer, business logic layer and data access layer.

* **Presentation Tier** is the tier in which the users interact with an application. It is a single-page-application of responsive nature. Presentation Tier will consume restful API implemented in Business Tier to display content to the user.
* **Business Tier** *is mainly working as the bridge between Data Tier and Presentation Tier*. All the Data passes through the Business Tier before passing to the presentation Tier. Business Tier is the sum of Business Logic Layer, Data Access Layer and Value Object and other components used to add business logic. It exposes Rest API which can be called by Presentation Tier to display content to the user. It will also send the data from Presentation tier to Data Tier using Rest API.
* **Data Tier** is basically the server which stores all the application’s data. Data tier contents Database Tables, XML Files and other means of storing Application Data.

# Architecture Diagram for MEAN Enterprise Full Stack



# Important Instructions

1. Adhere to the design specifications mentioned in the case study.
2. Feel free to create front-end and back-end of single page application from scratch. You are free to use angular-cli commands at command prompt to create SPA (Single Page Application) template.
3. Please make sure that your code does not have any compilation errors while submitting your case study solution.
4. The final solution should consist of three parts:
   1. Front-end built using HTML5, CSS3, Bootstrap and Angular JS as a SPA. Controllers written in Angular JS should consume restful API coded in business layer.
   2. Business-layer built using Spring Restful/JAX-RS Jersey or Node.JS/Express.JS.
   3. Database-layer built using MongDB or MySQL.

# Assessment Deliverables

1. For Front End – zipped application
2. For Backend - Packaged code files (Source code and WAR).
3. For SCM\* – Project Code should be present in active GIT repository
4. Few steps on how to run the solution.

\*SCM – Source Code Management