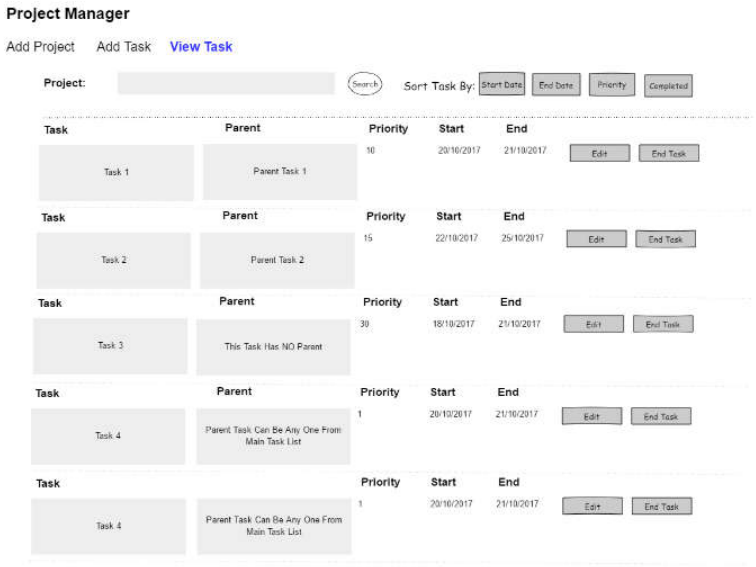


SINGLE PAGE APPLICATION ON PROJECT MANAGER

Java FSD 8 Hour Code Capsule

ABSTRACT

The Project Manager solution should be a Single Page Application (SPA) to keep track of upcoming projects and their respective tasks and their status and priorities.

Developer

Java Full Stack Developer

Contents

Important Instructions 2

Business-Requirement: An Overview 2

Methodology 3

- Follow TDD 3
- Load Testing 3
- Continuous Integration 4

Technical Spec – Solution Development Environment 4

- Front End Layer 4
- Middle Tier Layer 4
- Database & Integration Layer 4
- Ancillary Layer 4
- Deployment & Infrastructure 5
- Editors 5

Wireframes 6

- Add User 6
- Add Project 7
- Add Task 8
- View Task 8
- Update Task 9

Requirements 9

- Requirement 1: Add User Screen 9
- Requirement 2: Add Project Screen 10
- Requirement 3: Add Task Screen 11
- Requirement 4: View Task Screen 11

Database Tables 12

Software Requirements 12

Architecture 13

Architecture Diagram for Java Enterprise Full Stack 14

Important Instructions 14

Assessment Deliverables 15

Important Instructions

1. Adhere to 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. If you are using C#.Net/VB.Net, ASP.net, ASP.net MVC, WCF, Web API to build Rest API, then use relevant tools to build the project.
10. Write web service which takes input and return required details from database.
11. Use JSON format to transfer the results.

Business-Requirement: An Overview

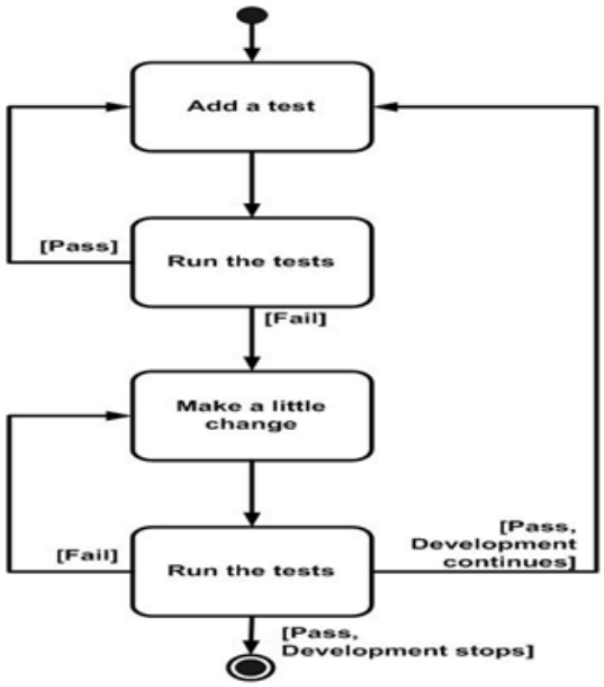
The Project Manager Single Page Application allows you to manage projects and their respective tasks. It allows you to set priorities to each project and task. You can associate one manager to each project and task-owner for each task. Each task will have parent task, start date, end date and task owner. Below are the features of Project Manager:

1. Add/Edit/View/Delete User
2. Add/Edit/View/Delete Project
3. Add/Edit/View/Delete Task

4. You can make one task a parent of another task.
5. On view task screen:
    a. User can search tasks by project name and sort by start date, end date or completed status.
    b. User can edit task
    c. User can end the task once it is finished. Once ended, user cannot edit the task. User can only view the task once finished.

Methodology

Follow TDD



Development approach to follow

a. Use JUNIT (> v4) to write your test cases

b. Use Emma Coverage (http://emma.sourceforge.net/) tool to measure the depth of your test cases - whether happy/unhappy scenarios are all covered in your testing.

c. Externalize your test data from your test code

d. The minimum acceptable Emma coverage statistic for methods - 85 %

Load Testing

One of the major improvement areas for associates is to write code that can sustain concurrent, multiple hits to the services written. To measure how scalable your code can be load testing should be incorporated as a part of development process rather than an effort conceived post delivery.

a. Set up Apache JMeter (Open source v>3) in your environment

b. Package the same JUNIT test cases along with the dependencies and deploy them to JMeter

c. Simulate at least 500 concurrent threads and run the load test for at least 10 minutes

d. The findings of the report should be included as a part of final assessment deliverable.

Continuous Integration

a. Check in your code - ideally every day or at worst every 2 days or so.

b. Set up Jenkins on the cloud and integrate with your Github code base.

c. Trigger build on every check in -the build should, apart from compiling, also run the test cases.

d. Save the build report's as a part of final assessment deliverable - this will measure the regular check in code by associates as well as TDD nature of development

Technical Spec – Solution Development Environment

Front End Layer

Framework(s)/SDK/Libraries	Version
Angular or	2 or above
React	13 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
Java Stack	Spring Boot	1.5 or above
	Spring MVC	4.0 or above
	JDK	1.7 or above
	Maven	3.x or above

Database & Integration Layer

Technology	Framework(s)/SDK/Libraries	Version
Java Stack	Hibernate	4.0 or above
	JAX-RS Jersey/ Spring Restful	
	MySQL	5.7.19
MongoDB	MongoDB	3.4
	NoSQL	

Ancillary Layer

Technology	Framework(s)/SDK/Libraries	Version
Source Code Management Tool	GIT	2.14.2
Build Tool/JAVA Stack	Maven	3.x
Testing Tool/JAVA Stack	JUnit	4.x

Testing Tool/JAVA Stack	Spring Test	4.x

Deployment & Infrastructure

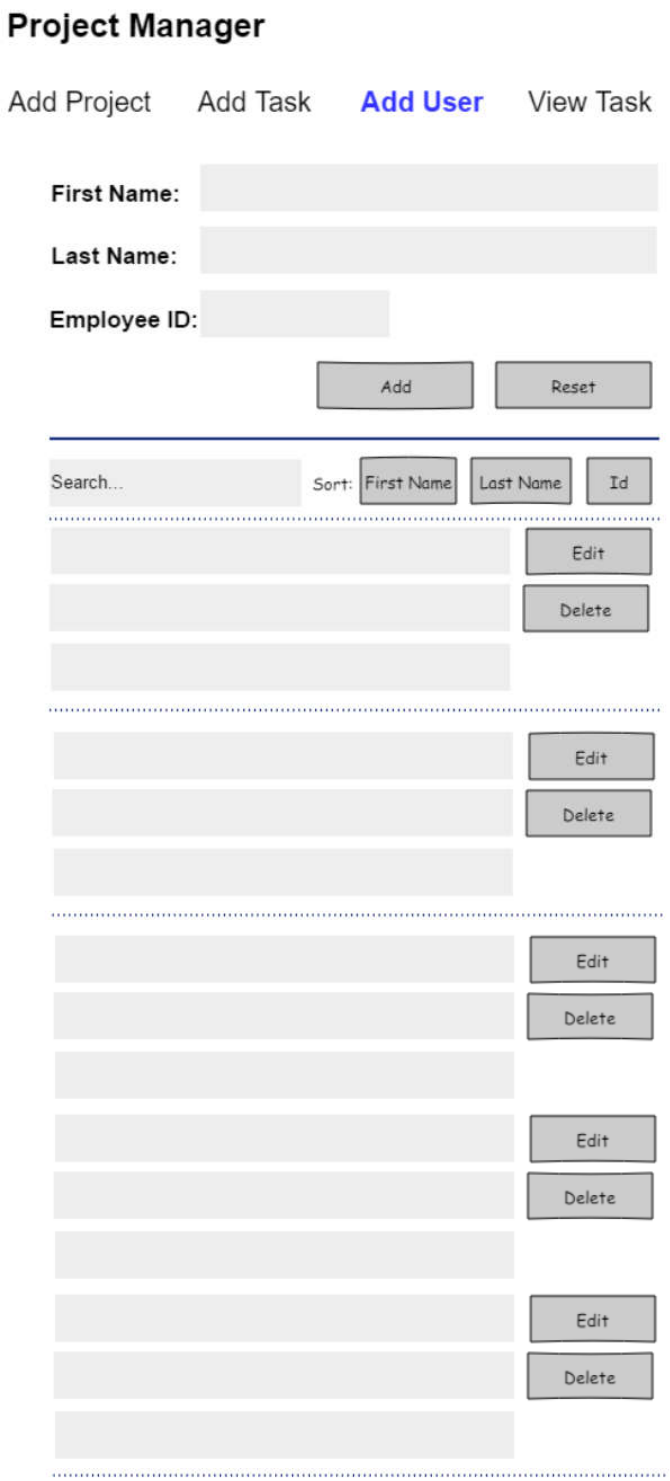
Technology	Framework(s)/SDK/Libraries	Version
Docker	-	
Apache Tomcat	-	
Apache HTTP (XAMPP)	-	
Node	-	
Dependency Management Tool	NPM	

Editors

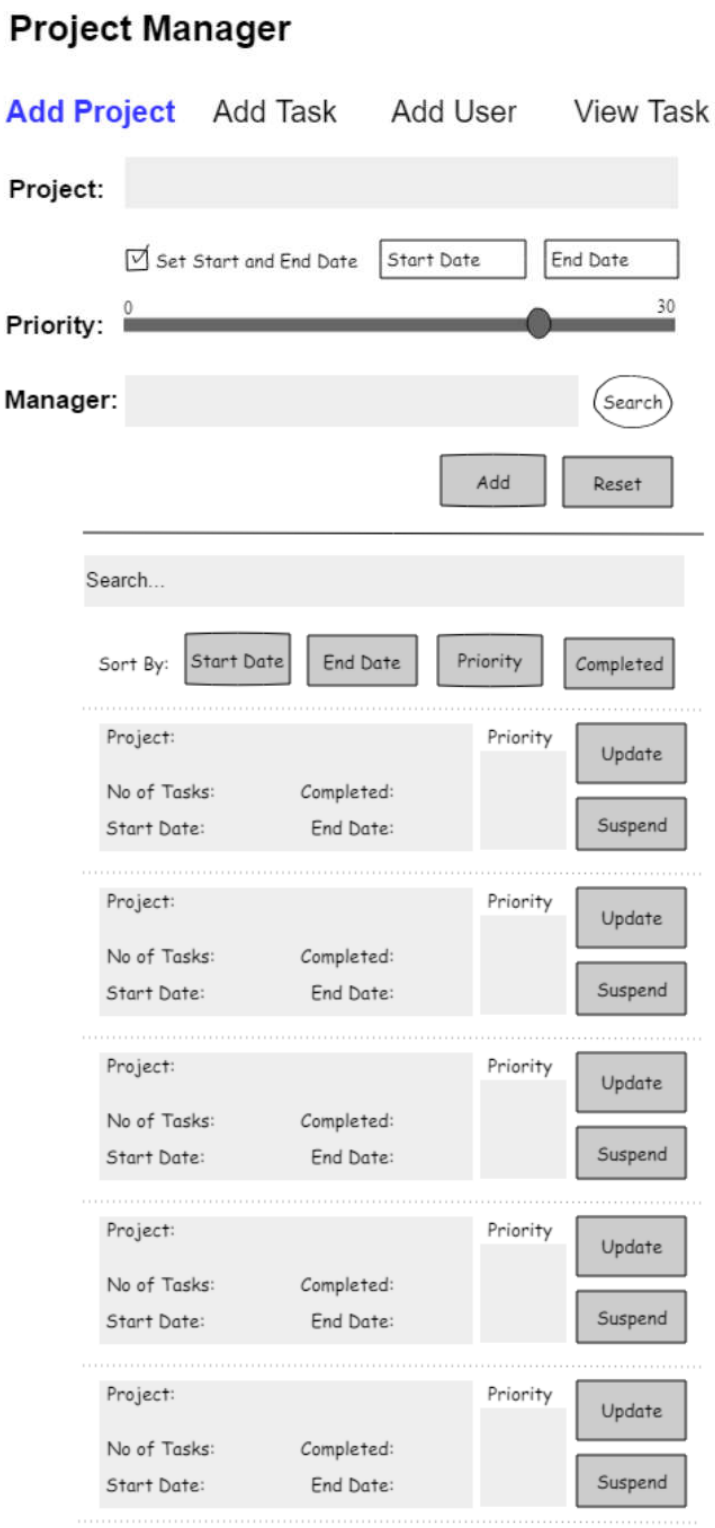
Name	Version
Adobe Brackets	
Sublime Text	
Atom Editor	
Eclipse J2EE Editor	

Wireframes

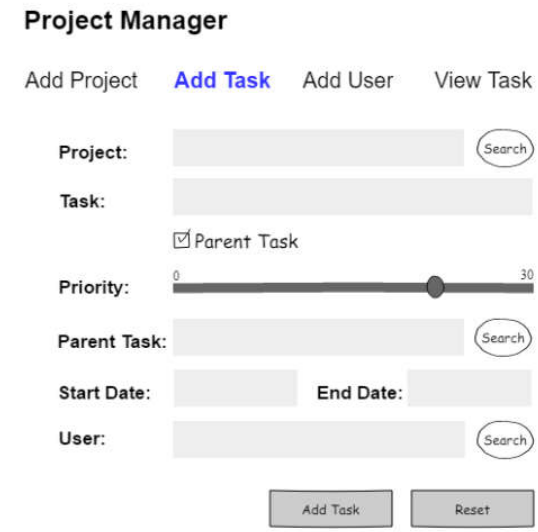
Add User



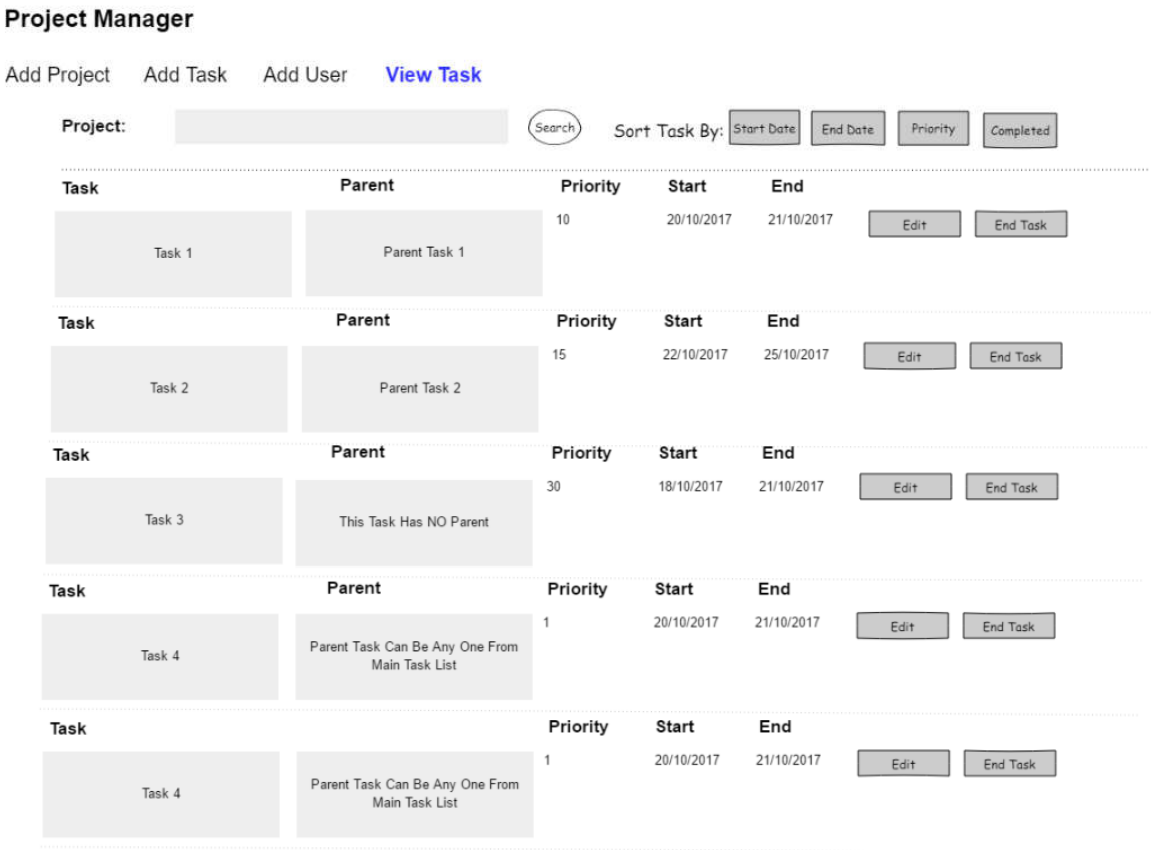
Add Project


Project Manager
Add Project
Add Task
Add User
View Task
Project:
Set Start and End Date
Start Date
End Date
Priority:
0
30
Manager:
Search
Add
Reset
Search...
Sort By:
Start Date
End Date
Priority
Completed
Project:
No of Tasks:
Completed:
Start Date:
End Date:
Priority
Update
Suspend

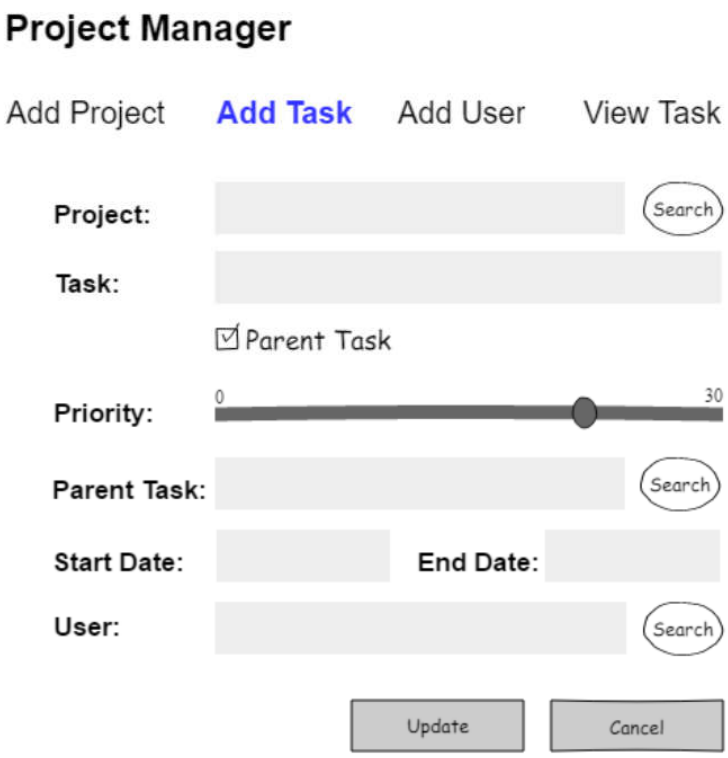
Add Task



View Task



Update Task



Requirements

Requirement 1: Add User Screen

1. Create Add User Screen UI
    a. Page should have header.
    b. Page should be responsive
    c. Navigation to “Add Project” Page, “Add Task” Page, “View Task” Pages should be as per SPA standards.
    d. This page will open when user clicks on “Add User” link.
    e. Page is divided into 2 sections:
        i. Add User
        ii. View User
    f. Add User
        i. Fields to be input by the user are first name, last name and employee id.
        ii. Proper validations should be in place to prevent user from entering blank first name or last name and employee id. All fields are mandatory.
        iii. Clicking on add button will create a JSON object and send it to the restful API for saving the user information in the database.
        iv. Each new user which is updated in the database via add user section will be automatically updated in real time in view user section.
    g. View User
        i. It will list all the users in the list view.
        ii. Each user information is grouped into list group (bootstrap class)

iii. Each user information contains first name, last name and employee id.
iv. It will also have edit and delete button.
v. Clicking on edit button will populate the user information in the top add user form. The add button at this instance will change to update button.
vi. Clicking on the update button will update only this record in the database via POST method call to restful API.
vii. This section also has 3 buttons, each for sorting the users by their first name, last name and employee id.
viii. It also has a search filter to search particular user from the list of users available in the list.

Requirement 2: Add Project Screen

2. Create Add Project Screen UI
    a. Page should have header.
    b. Page should be responsive
    c. Navigation to "Add Project" Page, "Add Task" Page, "View Task" Pages should be as per SPA standards.
    d. This page will open when user clicks on "Add Project" link.
    e. Page is divided into 2 sections:
        i. Add Project
        ii. View Project
    f. Add Project
        i. Fields to be input by the user are project title.
        ii. If 'set start and end date' is checked, then only input fields for start date and end date should be enabled, else they are disabled. If enabled, then default start date should be present date and default end date should be 1 day ahead of present date.
        iii. If user is selecting the start and end date of his/her preference, then proper validation should be in place for checking that end date is not before start date.
        iv. HTML5 Input type for start date and end date is date type.
        v. Priority is set using range control. By default the priority is set to 0.
        vi. Proper validations should be in place to prevent user from entering blank project.
        vii. The user will add manager to the project by clicking on the search button. List of all the users which are added in the application will be available for selection in bootstrap modal box. Modal box will also have filter to search for users. User will be able to select any one user and assign it as manager. This field is un-editable.
        viii. Clicking on add button will create a JSON object and send it to the restful API for saving the project information in the database.
        ix. Each new project which is updated in the database via add project section will be automatically updated in real time in view project section.
    g. View Project
        i. It will list all the projects in the list view.
        ii. Each project information is grouped into list group (bootstrap class)

iii. Each project information contains project name, no of tasks in the project, number of tasks completed, start date of the project, end date of the project and priority.
iv. It will also have edit and suspend button.
v. Clicking on edit button will populate the project title and other details in the top add project form. The add button at this instance will change to update button.
vi. Clicking on the update button will update only this record in the database via POST method call to restful API.
vii. This section also has 4 buttons, each for sorting the project names by start date, end date, priority and completed status.

Requirement 3: Add Task Screen

1. Create Add Task Screen UI.
    a. Page should have header.
    b. Page should be responsive
    c. Navigation to "Add Project" Page, "Add Task" Page, "View Task" Pages should be as per SPA standards.
    d. This page will open when user clicks on "Add Task" link.
    e. The user will select the project for which tasks are required to be added by clicking on the search button. List of all the projects which are added in the application will be available for selection in bootstrap modal box. Modal box will also have filter to search for projects. User will be able to select any one project. This field is un-editable.
    f. Next, user can add the task to the project.
    g. Proper validation should be in place to check if this field is empty.
    h. If this task is parent task, then remaining fields i.e. priority, parent task, start date and end date will be disabled. User will be able to add title for parent task only.
    i. Next, priority for the task can be set using range control in HTML5.
    j. Next, parent task can be searched and linked. This field is un-editable. It is not mandatory.
    k. Next, start date and end date can be set using HTML5 date input type.
    l. By default, date of present day is set as start date. Date of next date to present date is set as end date. End date can never be before start date.
    m. The user will add user to the project by clicking on the search button. List of all the users which are added in the application will be available for selection in bootstrap modal box. Modal box will also have filter to search for users. User will be able to select any one user and assign it as task owner. This field is un-editable.
    n. Once, all fields are filled with information, user can click on add button to convert the data to JSON format and send it to rest API to POST to database.

Requirement 4: View Task Screen

1. Create View Task Screen UI.
    a. Page should have header.
    b. Page should be responsive
    c. Navigation to "Add Project" Page, "Add Task" Page, "View Task" Pages should be as per SPA standards.
    d. This page will open when user clicks on "View Task" link.

e. The user will select the project for which tasks are required to be viewed by clicking on the search button. List of all the projects which are added in the application will be available for selection in bootstrap modal box. Modal box will have search filter to search for projects. User will be able to select any one project. This field is un-editable.
f. Once the project is selected, all its tasks along with parent-task or no-parent-task are displayed in the grid view along with priority, start date and end date.
g. This screen will also have edit task and end task button.
h. Clicking on end task will end the task to set its status to complete.
i. Clicking on edit task will open the edit task screen which will have same rules as that of add task. User will not be able to change the project, parent task status on edit task screen.
j. User will only be able to edit task description, priority, change/remove parent task, change start date and end date on edit task screen.
k. Clicking on update will post the changes to database via rest api.

Database Tables

Parent Task Table
Parent_ID
Parent_Task

Task Table
Task_ID
Parent _ID
Project_ID
Task
Start_Date
End_Date
Priority
Status

Project Table
Project_ID
Project
Start Date
End Date
Priority

Users Table
User_ID
First Name
Last Name
Employee_ID
Project_ID
Task_ID

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, Angular)
- Middleware Restful API (Spring Restful, JAX-RS, Spring MVC)
- Database Persistence (Hibernate)
- Database layer (MySQL, MongoDB)
- Ancillary skills (GIT, Docker, Maven) 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

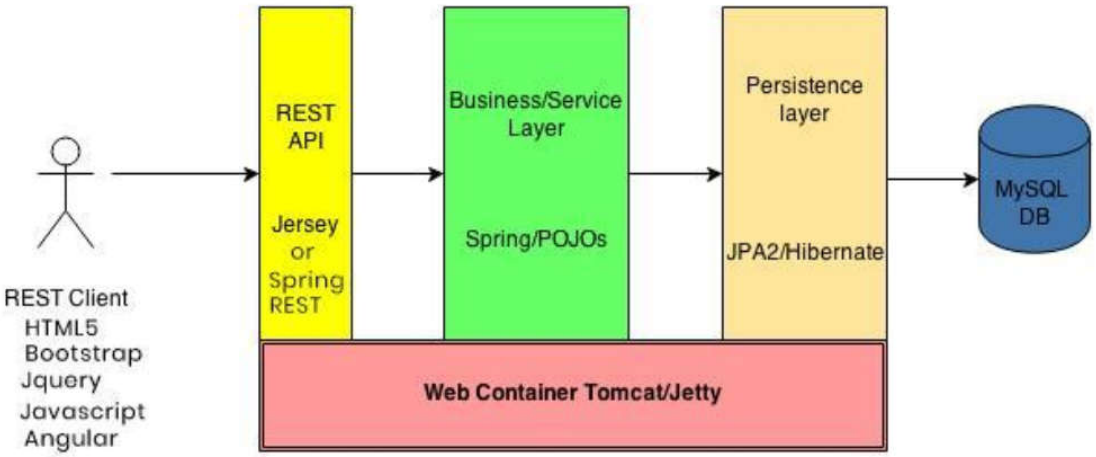
A physical architecture is an arrangement of physical elements, (system elements and physical interfaces) that provides the designed solution for a product, service, or enterprise. It is intended to satisfy logical architecture elements and system requirements. Workout Tracker follows a three layered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

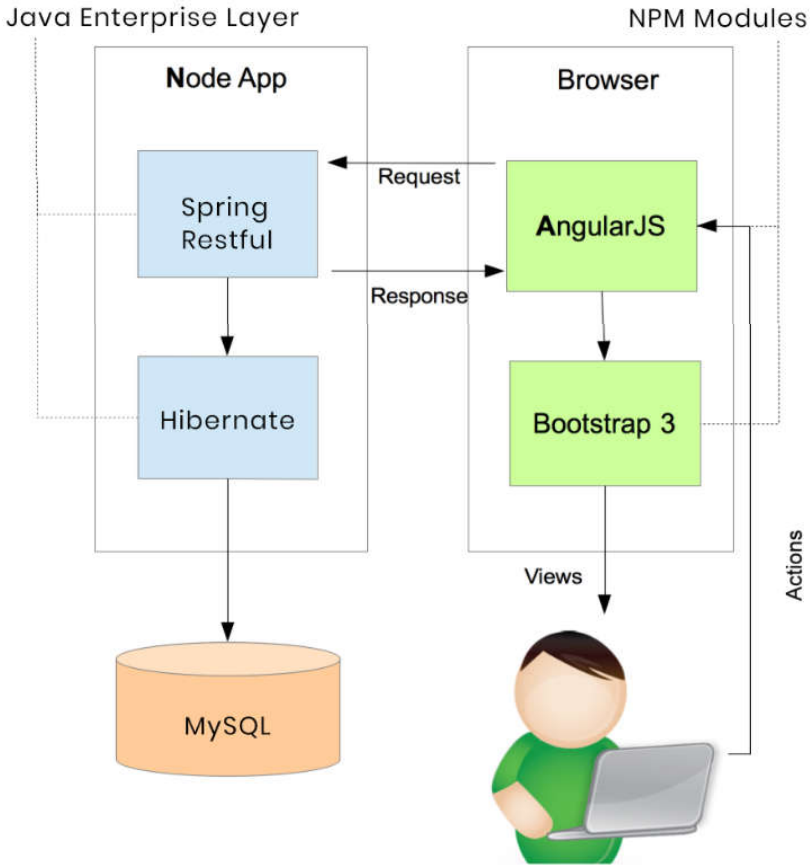
Presentation Tier is the tier in which the users interact with an application. It is a single-page-application of responsive nature. Presentation Tier will consume restful API implemented in Business Tier to display content to the user.

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 Java Enterprise Full Stack





Important Instructions

12. Adhere to the design specifications mentioned in the case study.
13. 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.

14. Please make sure that your code does not have any compilation errors while submitting your case study solution.
15. The final solution should consist of three parts:
    a. Front-end built using HTML5, CSS3, Bootstrap and Angular as a SPA. Controllers written in Angular JS should consume restful API coded in business layer.
    b. Business-layer built using Spring Restful/JAX-RS Jersey or Node.JS/Express.JS.
    c. 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.
5. Regular Build Reports from CI server
6. Emma coverage reports
7. Load test reports

*SCM – Source Code Management