|  |  |
| --- | --- |
| Single Page Application on Work Out Tracker  Java FSD 8 Hour Code Capsule | Abstract  The client would like to develop an independent application; Personal Workout Tracker single page app; which is used to manage personal workout and fitness plans.  Developer  Java Full Stack Developer |

Contents

[Important Instructions 2](#_Toc499817718)

[Business-Requirement: An Overview 2](#_Toc499817719)

[Business-Requirement: Summary 3](#_Toc499817720)

[Technical Spec – Solution Development Environment 3](#_Toc499817721)

[Front End Layer 3](#_Toc499817722)

[Middle Tier Layer 3](#_Toc499817723)

[Database & Integration Layer 3](#_Toc499817724)

[Ancillary Layer 3](#_Toc499817725)

[Deployment & Infrastructure 4](#_Toc499817726)

[Editors 4](#_Toc499817727)

[Wireframes 5](#_Toc499817728)

[Wireframe: Start Workout Screen 5](#_Toc499817729)

[Wireframe: Add Workout Screen 6](#_Toc499817730)

[Wireframe: Add Workout Screen 7](#_Toc499817731)

[Wireframe: Edit Workout Screen 8](#_Toc499817732)

[Wireframe: End Workout Screen 9](#_Toc499817733)

[Wireframe: Start Workout Screen 10](#_Toc499817734)

[Wireframe: Track Workout screen 11](#_Toc499817735)

[Software Requirements 12](#_Toc499817736)

[Database Structure 13](#_Toc499817737)

[Architecture 13](#_Toc499817738)

[Architecture Diagram for Java Enterprise Full Stack 14](#_Toc499817739)

[Important Instructions 15](#_Toc499817740)

[Assessment Deliverables 15](#_Toc499817741)

[For Front End 15](#_Toc499817742)

[For Backend - Packaged code files (Source code and WAR). 15](#_Toc499817743)

[For Testing 15](#_Toc499817744)

[Source Code Management 16](#_Toc499817745)

# 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 and test cases.
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.
12. Create test cases using cucumber features & respective scenarios.

# Business-Requirement: An Overview

Before beginning work on any project, it's usually a good idea to know what you're building. Below is a basic list of things we want our users to be able to do:

The client would like to develop an independent application; Personal Workout Tracker single page app; which is used to manage personal workout and fitness plans.

Workout Tracker:

1. This web application allows user to enter in and save the workout plans.
2. Each new workout entered by user will capture calories burnt per min along with
3. It will allow you to start the workout and end the workout whenever the user does the workout and ends the workout respectively.
4. It will present graphs that show your progress over time.
5. It will provide report on number of calories burnt.
6. It will provide report on number of minutes the workout was done on each day, each week and each month and also show the graph for the same.

# Business-Requirement: Summary

Workout Tracker

1. Add/Update/Delete/View Workout
2. Add/Update/Delete/View Category
3. Start Workout
4. End Workout

# 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** |
| 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

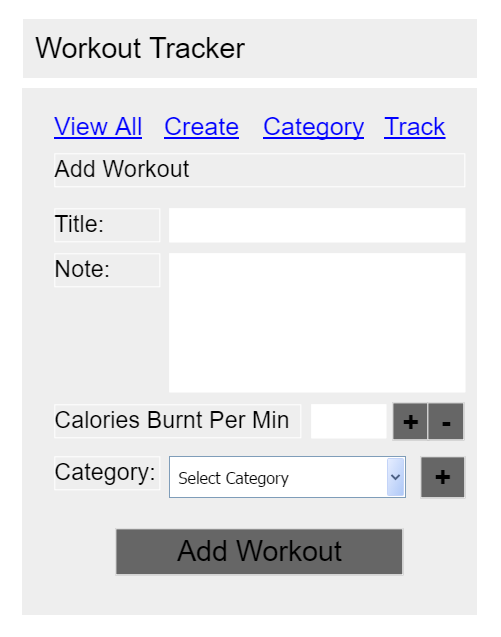
## Wireframe: Start Workout Screen



Remain disabled by default. It will be enabled only after the particular workout is started by clicking on start button

It has a functionality of autofilter. It will filter the search results based on what user types in the search box. It is implemented using angular-js filter.

1. Displays the list of workout ideas entered by user.
2. User can edit, delete the workout
3. User can start the workout when user wants to start the workout. It will get disabled after the workout is started.
4. User can end the workout after user is done with the workout. It is disabled by default. It will be enabled only if workout is started.
5. The screen has a search bar with auto-filter enabled i.e. as user types the workout; he/she wants to search the results will get auto filtered below.
6. Navigation in the website happens as a SPA (Single Page Application)

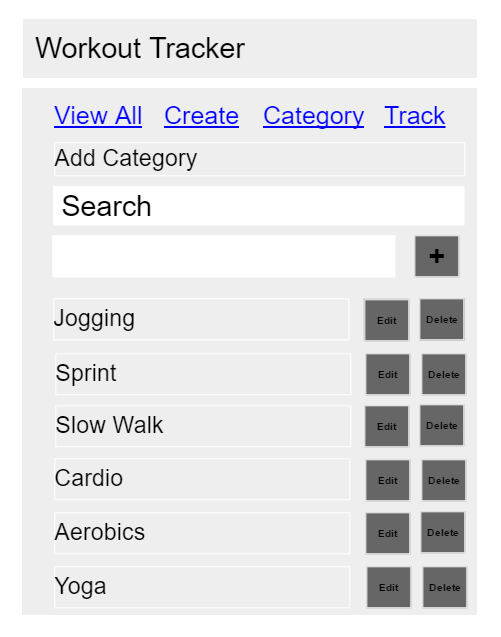
Wireframe: Add Workout Screen

When the page loads, ‘Add Workout’ is disabled. It gets enabled only after the Title, Calories Burnt Per Min and Category is selected.

Floating Point Numbers only .

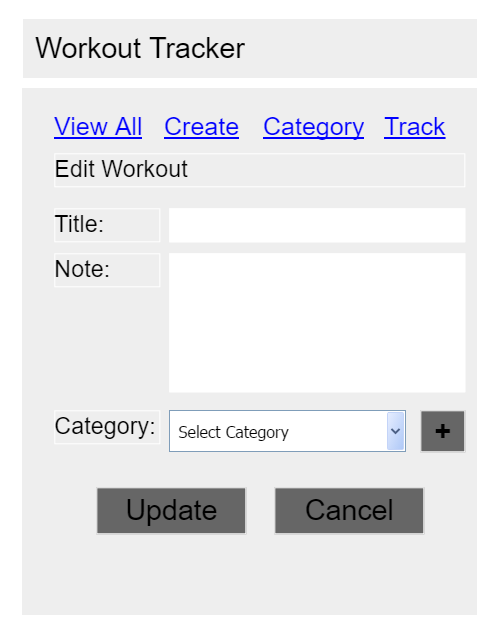
* Clicking on + will add by 0.1
* Clicking on – will decrease by 0.1

1. This screen allows you to add the workout.
2. You can also manage the category to which workout belongs in this screen.
3. You will need to input the calories burnt per min if the user does this workout.
4. Ref: https://whatscookingamerica.net/Information/CalorieBurnChart.htm

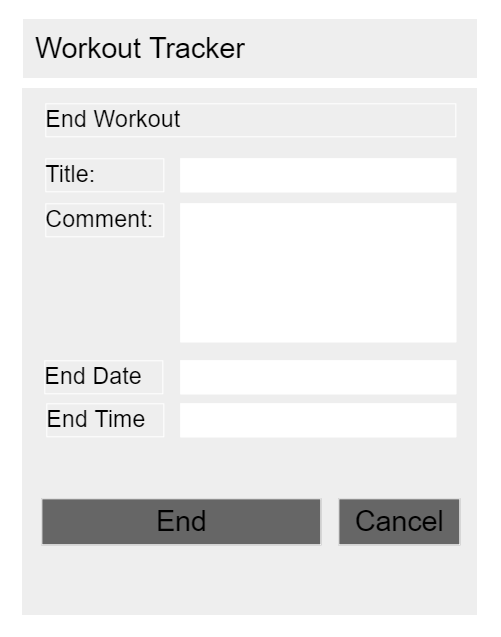
Wireframe: Add Workout Screen

Proper validation should be in place to prevent user from entering blank category

1. It allow user to manage categories
2. The screen has auto-search box which will allow user to filter the search results while typing in the search box.

Wireframe: Edit Workout Screen

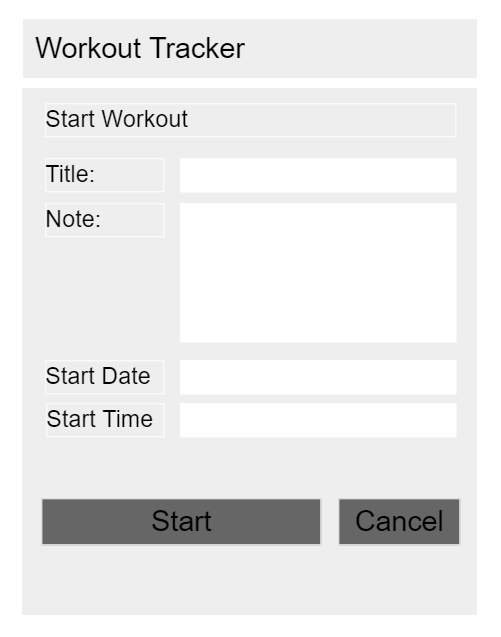
1. It allows users to edit any workout they may have entered

Wireframe: End Workout Screen

Date Format. By default, populated with current date

Time Format. By default, populated with current time

1. It allows users to end the workout which is already started.
2. Users can input the end date and end time.
3. By default workout title is populated. This is un-editable.
4. By default current date and current time is populated.

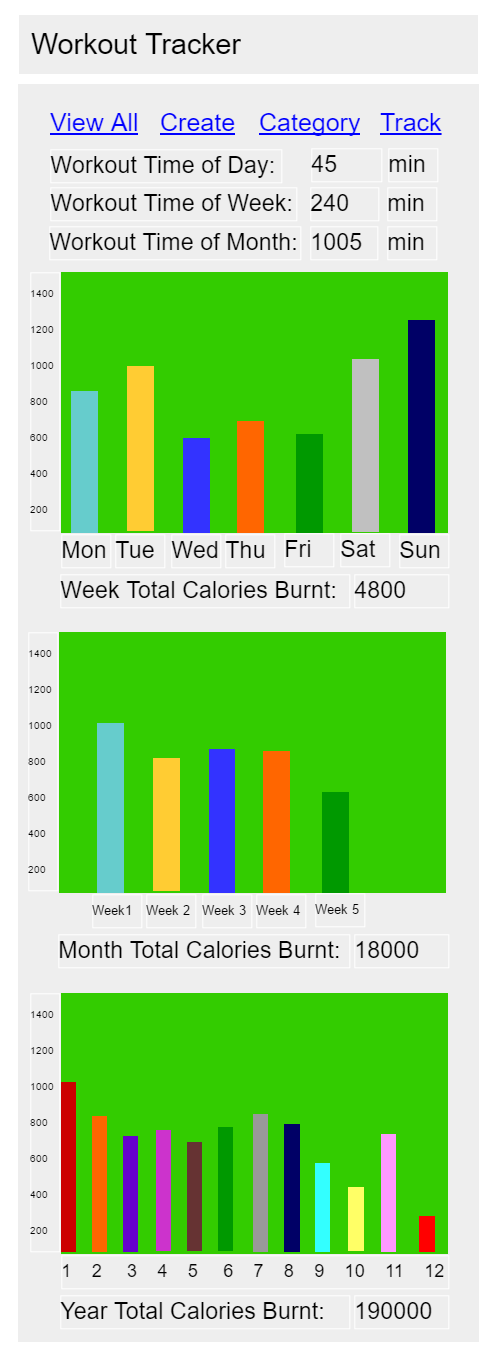
Wireframe: Start Workout Screen

Date Format. By default, populated with current date

Time Format. By default, populated with current time

1. It allows users to start the workout.
2. Users can input start date and start time.
3. By default workout title is populated. This is un-editable.
4. By default current date and current time is populated.

## Wireframe: Track Workout screen



1. This page displays various reports:
   1. Current day workout time. It will calculate the sum of all the workouts done on the current day.
   2. Current week workout total time. It will calculate the sum-time of all the workouts done in the week.
   3. Current month workout total time. It will calculate the sum-time of all the workouts done in the month.
   4. Charts:
      1. Week wise calorie burn chart
      2. Month wise calorie burn chart
      3. Year wise calorie burn chart

# 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 (Spring Restful, JAX-RS, Spring MVC)
* Database Persistence ( Hibernate)
* Database layer (MySQL, MongoDB)
* Ancillary skills (GIT, Docker, Maven, Cucumber, Appium, Selenium) 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

# Database Structure

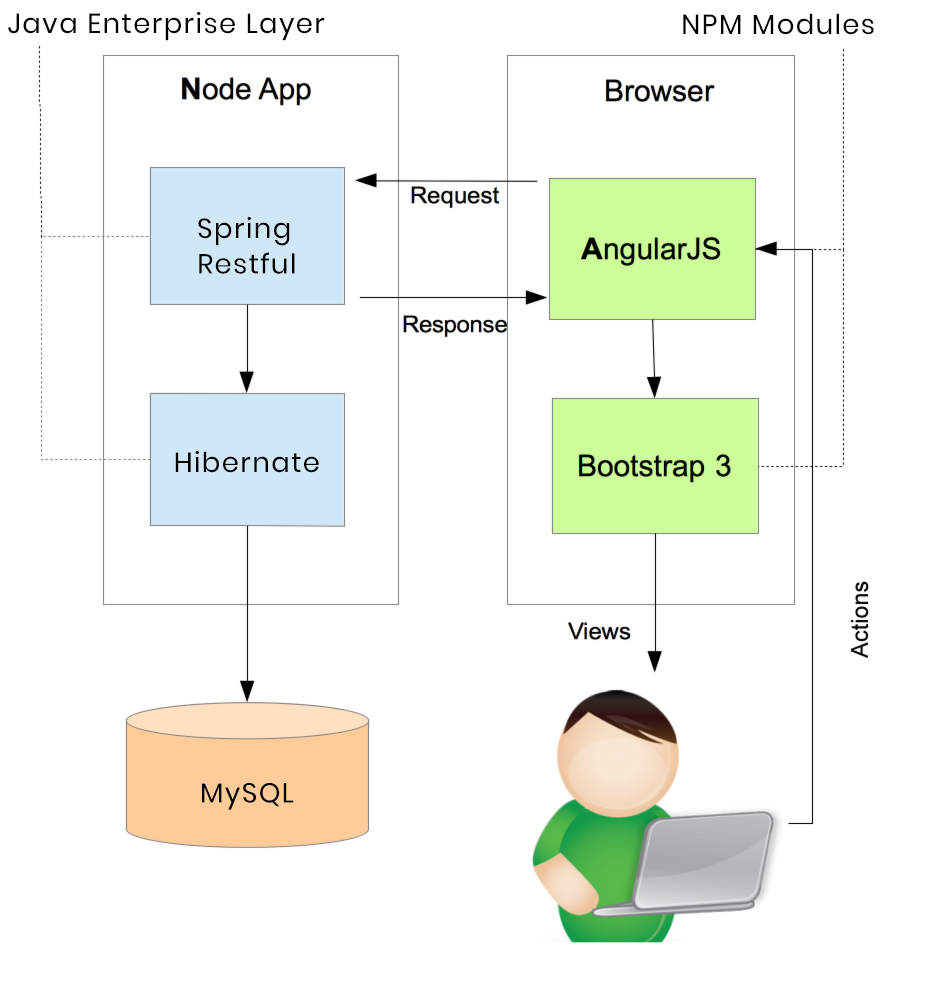
|  |  |  |
| --- | --- | --- |
| **workout\_collection** | | |
| workout\_title | varchar\_128 |  |
| workout\_note | varchar\_256 |  |
| calories\_burn\_per\_min | float |  |
| category\_id | integer | foreign |
| workout\_id | integer | primary |
|  |  |  |
| **workout\_active** | | |
| workout\_id | integer | foreign |
| start\_time | time |  |
| start\_date | date |  |
| end\_date | date |  |
| end\_time | time |  |
| comment | varchar\_64 |  |
| status | boolean |  |
|  |  |  |
| **workout\_category** | | |
| category\_id | integer | primary |
| category\_name | varchar\_64 |  |

# 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

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 MongoDB or MySQL.

# Assessment Deliverables

## For Front End

* 1. SPA (Single Page Application) developed using HTML5, Bootstrap, Javascript & Angular. Angular implementation should be done using the combination of controllers, services and factory.
  2. UI Layer should call respective controllers. Controllers should inject respective services and services should inject respective factories.
  3. Restful API calls should be implemented in factories. Each Restful API should receive JSON data as a response from its call.
  4. In case restful API’s are not available, front end team can simulate API call by creating the JSON objects in factories and then injecting them in controllers. Once in controllers, the same JSON objects can be used to populate the UI.
  5. Page Navigation should be implemented using ui-view ng-router with clean URL’s
  6. UI should be responsive.

## For Backend - Packaged code files (Source code and WAR).

* 1. The war with the source code and executable which can be deployed in Apache Tomcat.
  2. Should have implemented Restful API which can be tested using POSTMAN tool.
  3. Document each API in the following format **(below is a dummy data)**:
     1. API URL: <http://www.abc.com/edit/>workout/id/1
     2. Type: GET or POST or PUT or DELETE
     3. Request Data: {title:`Jogging`, time:24000, calories:5}
     4. Response Data: success or failure or JSON object
  4. After deploying the application in the apache tomcat, each documented Restful API should work.
  5. The same document will be used by UI team as a reference to implement Restful API call in angular.

## For Testing

* 1. Create test cases and respective feature files in eclipse, with different scenarios based on “Workout Tracker”.
  2. Consider these scenarios:
     1. Add workout test for workout tracker
     2. Add category test for workout tracker
     3. Edit workout test for workout tracker
     4. Start workout test for workout tracker
     5. End Workout test for workout tracker
     6. Delete workout test for workout tracker

## Source Code Management

All project Code should be present in active GIT repository