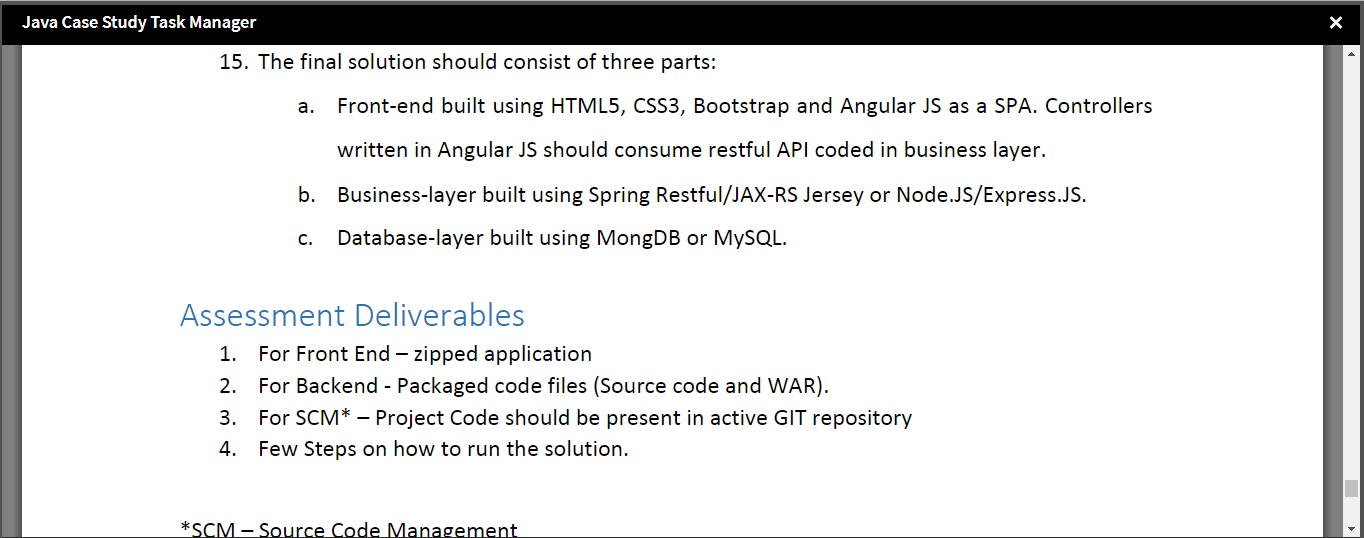
Requirements for the Capsule Program - 

**Core Content**

1. UI Layer Submission
2. Middleware Server side Services Submission
3. Database connectivity, Integration &Performance Testing Submission
4. Deployment Submission
5. **Final Submission - All the individual Step 1 to Step 4 are given in detail below which should cover the final submission. Also attachments are given in the end of this document for the source code.**

# All Information on #1 UI Layer Submission

# project-manager-ui

If you are running the application from localhost/Docker(@latest version), the url needed to be hit to access the application is - [http://localhost:4200](http://localhost:4200/)

If you are running the application from Docker Tool Box for windows (for older windows versions), the url needed to be hit to access the application is - [http://192.168.99.100:4200](http://192.168.99.100:4200/)

## The following should be executed only when your service layer is up and running:

{{baseurl}} = the context url with dns and port and application context

Example -

Local PC/Docker - http://localhost:8080/projectmanagerbackend/

Docker Tool Box for older windows - http://192.168.99.100:8080/projectmanagerbackend/

#### IF DATABASE IS EMPTY WE NEED TO hit the following services in the order given

###### The following services can be hit from Browser as well after bringing up the service, since, they are normal GET methods. The POSTMAN Collection can be imported into POSTMAN from the location -

####### project-manager-backend/other-resources/postman-import-json\_for\_docker\_toolbox/

####### project-manager-backend/other-resources/postman-import-json\_for\_localhost/

{{baseurl}}user/dump

{{baseurl}}parent/dump

{{baseurl}}project/dump

{{baseurl}}task/dump

Information on docker commands and execution -

# First Step - Go to UI project root folder and execute the build command: ng build [-or-] ng build --prod

## Docker for UI Layer - PROJECT MANAGER Project

# General Commands to be executed in Docker related to push/build/run-

docker build -t your\_image\_name your\_repo\_url

docker tag your\_image\_name docker\_username/your\_image\_name

docker push your\_image\_name docker\_username/your\_image\_name

docker run -d -p 80:80/tcp -p 80:80/udp --name your\_image\_name your\_image\_id\_value

# Example for push/build/run-

docker build -t project-manager-ui https://github.com/fsduseriiht/project-manager-ui.git

docker tag project-manager-ui amitabhadockerwork/project-manager-ui

docker push amitabhadockerwork/project-manager-ui

docker run -d -p 80:80/tcp -p 80:80/udp --name project-manager-ui\_running <your\_image\_id\_value>

# All Information on #2 Middleware Server side Services Submission

# project-manager-backend

## Information on the services exposed -

#### URL

{{baseurl}} = the context url with dns and port and application context

Example -

Local PC/Docker - http://localhost:8080/projectmanagerbackend/

Docker Tool Box for older windows - http://192.168.99.100:8080/projectmanagerbackend/

#### IF DATABASE IS EMPTY WE NEED TO hit the following services in the order given

###### The following services can be hit from Browser as well after bringing up the service, since, they are normal GET methods. The POSTMAN Collection can be imported into POSTMAN from the location -

####### project-manager-backend/other-resources/postman-import-json\_for\_docker\_toolbox/ ####### project-manager-backend/other-resources/postman-import-json\_for\_localhost/

{{baseurl}}user/dump

{{baseurl}}parent/dump

{{baseurl}}project/dump

{{baseurl}}task/dump

#### User\_TABLE

GET METHOD - {{baseurl}}user/dump - to pre-populate db with user details dump

GET METHOD - {{baseurl}}user/list - is used to display all records in user table db

POST METHOD - {{baseurl}}user/ - is used to create a single user record in user table in db

PUT METHOD - {{baseurl}}user/{{userId}} - is used to update the user table single record in db

DEL METHOD - {{baseurl}}user/{{userId}} - is used to delete single user record from user table in db

#### Parent\_TABLE

GET METHOD - {{baseurl}}parent/dump - to pre-populate db with parent task details dump

GET METHOD - {{baseurl}}parent/list - is used to display all the parent task values in the db

POST METHOD - {{baseurl}}parent/ - is used to create a parent task in the db

PUT METHOD - {{baseurl}}parent/{{parentId}} - is used to update a parent task in the db

DEL METHOD - {{baseurl}}parent/{{parentId}} - is used to delete a parent task from the db

#### Project\_TABLE

GET METHOD - {{baseurl}}project/dump - to pre-populate db with project details dump

GET METHOD - {{baseurl}}project/list - is used to display all records in project table db

POST METHOD - {{baseurl}}project/ - is used to create a single project record in project table in db

PUT METHOD - {{{baseurl}}project/{{projectId}} - is used to update the project table single record in db

DEL METHOD - {{baseurl}}project/{{projectId}} - is used to delete single project record from project table in db

#### Task\_TABLE

GET METHOD - {{baseurl}}task/dump - to pre-populate db with task details dump

GET METHOD - {{{baseurl}}task/list - is used to display all records in task table db

GET METHOD - {{baseurl}}task/countperproject - is used to display count of records in task table db for each project

GET METHOD - {{baseurl}}task/list/{{projectId}} - is used to display records in task table db for particular {{projectId}}

POST METHOD - {{baseurl}}task/ - is used to create a single task record in task table in db

PUT METHOD - {{baseurl}}task/{{taskId}} - is used to update the task table single record in db

PUT METHOD - {{baseurl}}task/convert/{{taskId}} - is used to convert task with {{taskId}} in task table to a parent task, thereby purging the task details in task table

DEL METHOD - {{baseurl}}task/{{taskId}} - is used to delete record from TaskTable in db

### useful links related to docker:

<https://cloud.docker.com/repository/docker/amitabhadockerwork/project-manager-backend>

### Docker commands

docker pull amitabhadockerwork/project-manager-ui   
docker pull amitabhadockerwork/project-manager-backend

## Docker for Service Layer - Project Manager Project

# General Commands to be executed in Docker related to push/build/run-

docker build -t your\_image\_name your\_repo\_url

docker tag your\_image\_name docker\_username/your\_image\_name

docker push your\_image\_name docker\_username/your\_image\_name

docker run --rm -i -t -d -p 8080:8080/tcp -p 8080:8080/udp --name your\_image\_name your\_image\_id\_value

# Example for push/build/run-

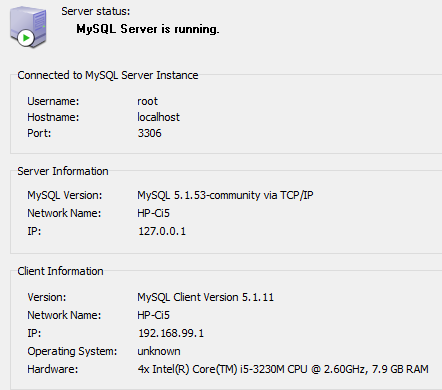
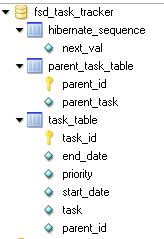
docker build -t project-manager-backend https://github.com/fsduseriiht/project-manager-backend.git

docker tag project-manager-backend amitabhadockerwork/project-manager-backend

docker push amitabhadockerwork/project-manager-backend

docker run --rm -i -t -d -p 8080:8080/tcp -p 8080:8080/udp --name project-manager-backend\_running <image\_id>

# All Information on #3 Database connectivity, Integration &Performance Testing Submission

**Note** - For Database, if you do not have MySql Installed in your system then you can use the H2 Database server profile also. Only you have to hit the the following urls if you do not find any data at application start (in the order given below)-

1. http://localhost:8080/tasktrackerbackend/parent/dump
2. http://localhost:8080/tasktrackerbackend/task/dump

# All Information on #3 Deployment Submission

For Deployment you can use Docker as mentioned above.

You can also deploy locally as per following steps.

**For service layer -**

1. Pull from the git repo - **https://github.com/fsduseriiht/tasktracker-fsd-service.git**
2. Build using maven (mvnw) which is given as a part of the source code package. Set java home folder only if necessary in the mvnw.cmd file as per your own system. [**mvnw clean install**]
3. Go to target folder generated as a part of build result of the previous step and then execute the war file generated as a result of the build process using [**java -jar <war file name>].**
4. Eg. - java -jar tasktracker-backend-0.0.1-SNAPSHOT.war
5. Check using post man or the logs in cmd prompt if the service is up or not.
6. Now start the ui application as follows.

**For UI - Layer -**

1. Pull from git repo - **https://github.com/fsduseriiht/tasktracker-fsd-ui.git**
2. The build folder is already available in the folder from repo. All you need is to run **ng serve** command to run the UI application from command prompt
3. Hit the url - **http://localhost:4200**

**Attachments -**

* UI Layer - 
* Service Layer - 
* DB Layer - Not needed as application will create the db/table if not exist.
* PostMan configuration script - 

Capsule Program Git URL - **https://github.com/fsduseriiht/capsule-program.git**