**Exam Question Paper Template**

**QI. Software Requirement Specification (SRS)**

1. **Abstract**
2. **Functional Requirements**
3. **Non-Functional Requirements**
4. **Identification of Users**
5. Number of modules

**QII. Maven Java Application Development**

You are given the following GitHub repository link:

<https://github.com/Kumbhambhargavi75/CampusMgmtSystem/>

1. **Import the project into Maven environment**

***Steps to import :***

**Open Eclipse**

1.Launch **Eclipse IDE** (preferably Eclipse IDE for Enterprise Java Developers).

**2. Clone Repository**

Go to the menu:

**File → Import → Git → Projects from Git (with smart import)** → Next.

Choose **Clone URI** → Next.

Enter the **GitHub repository URL** (SSH or HTTPS):

* + Example (HTTPS):
  + https://github.com/username/repository.git

Select the **branches** you want (usually main or master) → Next.

Choose the **directory** where Eclipse will store the repo → Finish.

Now you can see the imported project in the eclipse under project explorer

1. Questions on pom.xml (check the pom.xml correct or not and Resolve dependencies using pom.xml)
2. Build the project to generate the **WAR/JAR file**
3. Verify the generated artifact in the target/ folder
4. Solve the SBQ’s ( .M2repository, junit,test,build,finalname Etc related to pom .xml))

Note : Example , Add the **[JQuery](https://mvnrepository.com/artifact/org.webjars.bower/jquery) » [3.6.1](https://mvnrepository.com/artifact/org.webjars.bower/jquery/3.6.1)** dependency to your project

---- go to web browser type **[JQuery](https://mvnrepository.com/artifact/org.webjars.bower/jquery) » [3.6.1](https://mvnrepository.com/artifact/org.webjars.bower/jquery/3.6.1)** dependency

----you can see the dependency as below , copy and paste in pom.xml file in the dependencies and save the file.

<dependency>

<groupId>org.webjars</groupId>

<artifactId>jquery</artifactId>

<version>3.6.1</version>

</dependency>

**QIII. Git and GitHub**

**Task:** Work with Git and GitHub as follows:

Open gitbash in ECLIPSE ----Righclick on your mavenproject-->show in local terminal -->git bash

1. Initialize a Git repository and add your project files
2. Set global config,username and user email
3. Solve the given **SQB (Short Question Based)** Git tasks (e.g., branch, merge, revert,stash,restore,merge conflict,gitignore,clone ) Push your Maven project to GitHub

Gitbash open in the eclipse with the path of your project.

$git init

$git add .

$git commit -m “first commit”

$git config --global user.name “your user name”

$git config --global user.email “ your email id”

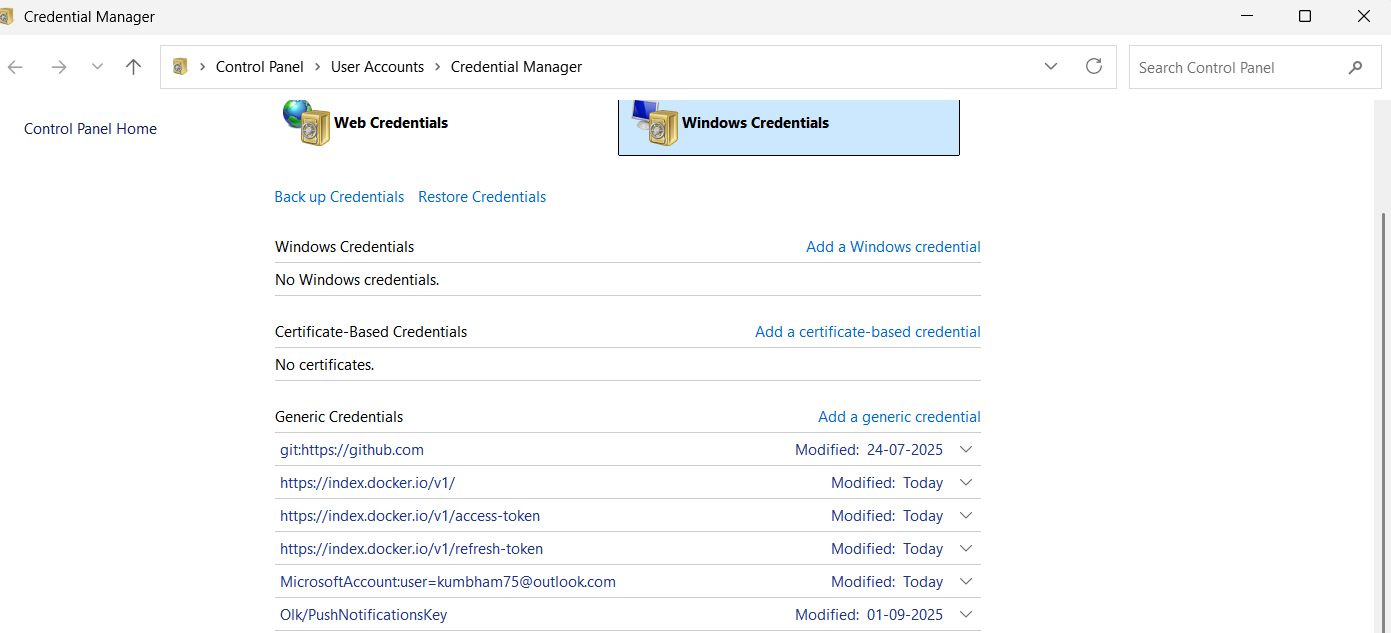
Solve the given **SQB’s**

Push your Maven project to your GitHub

Go to your github crate a repository for mavaneproject name

Come to git bash and push

Before pushing Check the windows credentials and remove the git hub accounts/login existing if any



$git branch -M main

$git remote add origin1 <https://github.com/Kumbhambhargavi75/LMSWEBP.git>

$git push -u origin1 main

Note : if you are trying to push for the given github, access denied getting so change origin(trying to push to the owner) to origin1 or other-name and clear the github accounts in windows credentials if exist

OR push by setting URL also

git remote set-url origin <https://github.com/YOUR>GITHUBUSERNAME/LMSWEBP.git

Now Refresh your github and check files exist or not

**QIV. Docker and Docker Compose**

Note: Containerize your Maven project using Docker.(open the docker desktop and docker hub account)

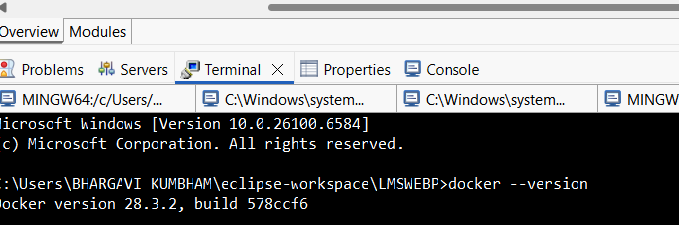
**Tasks:**

1. Creation of Dockerfile and build a docker image and push it to dockerhub
2. Run the image in a containerand expose to the particular port number(ex:9098)
3. Create a docker-compose.yml file and app work with the services like databases

Open gitbash in ECLIPSE ----Righclick on your mavenproject-->show in local terminal -- >terminal

OR can do in windows power shell

Here I opened in eclipse only ----Check the docker version exist or not



>docker ps –a   
Displays the list of images running currently or previously or stopped

>docker image ls

displays the list of docker images in the local server

>docker login

login to docker hub with required credentials

SBQ’s ( pull the image Locally/Hub and run a container,check the list of containers, stop the container, start the container, run it in interactive mode,build the image ,run it on localhost by given port number)

1. **Dockerfile Creation** 
   * Ensure it copies the WAR/JAR and runs on Tomcat (or relevant base image).

> code .

Opens the vs code write the create the Dockerfile and write it

Option 1: **Keep Tomcat default (8080)** // check your tomcat server port, otherwise getting an error

FROM tomcat:9.0

COPY target/\*.war /usr/local/tomcat/webapps/ROOT.war

EXPOSE 8080

CMD ["catalina.sh","run"]

### Explanation

1.FROM tomcat:9.0  
 Uses the official Tomcat 9 base image.

2.COPY target/\*.war /usr/local/tomcat/webapps/ROOT.war  
 Copies your WAR file(s) (from Maven/Gradle build) into Tomcat’s webapps folder.  
 When Tomcat starts, it will automatically unpack and deploy them.

By naming it ROOT.war, Tomcat deploys it as the **root web application (**/**)**, so you can access it at:http://localhost:<mapped-port>/

3.EXPOSE 8080

.Important: This does **not** actually change Tomcat’s port.

.Tomcat by default listens on **8080**, unless you edit server.xml.

1. CMD ["catalina.sh","run"]

.CMD defines the **default command** to run when the container starts.

.catalina.sh is Tomcat’s startup script.

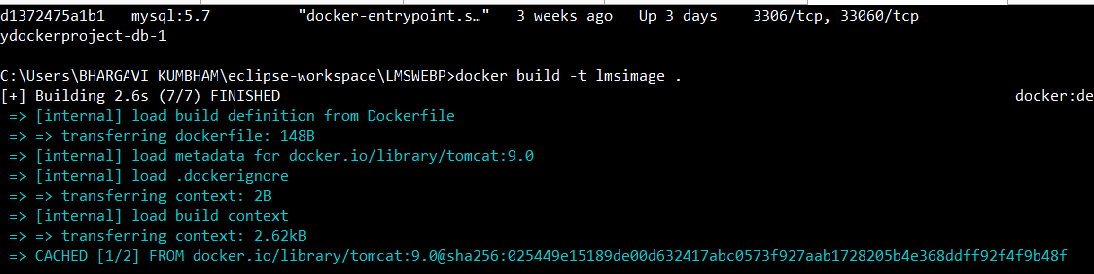
.The argument "run" tells it to:

Start Tomcat **in the foreground** (inside the container).

1. **Image Building** 
   * Build the Docker image using docker build -t <image\_name> .

Example : docker build -t lmsimage .

1. docker build → tells Docker to build an image.
2. -t <image\_name> → tags the image with a name (so you can run it later).
3. Example: -t lmsimage
4. Or if pushing to Docker Hub: -t kumbhambhargavi/lmsimage:latest
5. . → build context = the **current directory** (must contain your Dockerfile).



**RUN THE IMAGE**

$docker run -d -p 7089:8080 --name lmcontainer lmsimage

1. d → run in detached mode (in background).
2. -p 7089:8080 → map **host port 7089** → tomcat **container port 8080**.
3. --name lmcontainer → give the container a friendly name (lmcontainer).
4. lmsimage → start from your image.

Now check <http://localhost:7089/>

(opens the LMSWEBP APP)

1. **Push to Docker Hub**

* Tag and push the created image to your Docker Hub account.

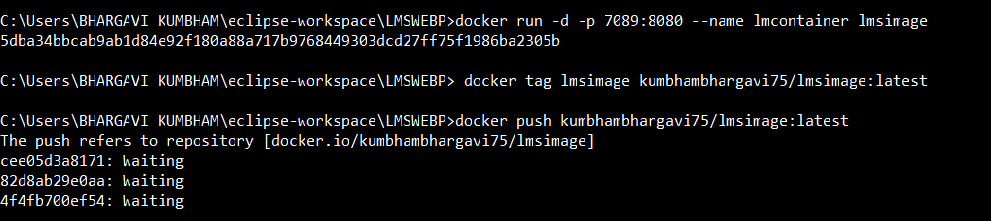
Docker Hub images must be tagged as:

<dockerhub-username>/<repository-name>:<tag>

Ex: docker tag lmsimage kumbhambhargavi/lmsimage:latest

docker push kumbhambhargavi/lmsimage:latest

This will upload your image to your Docker Hub repo.



Go to https://hub.docker.com/repositories → you can see lmsimage.

1. ***SQB***

A.How to run Ubuntu Image

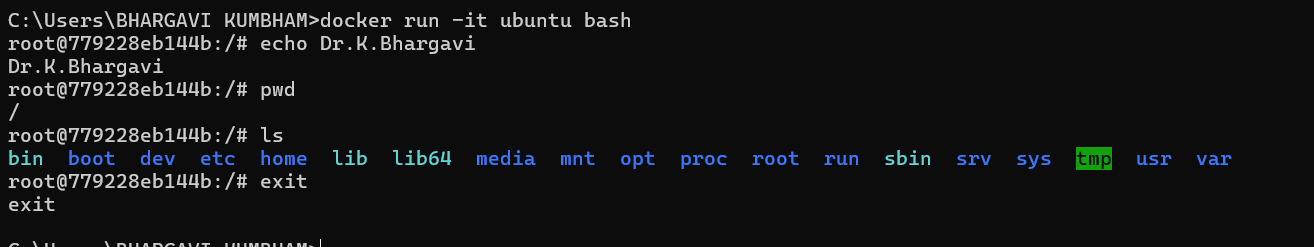
docker run [image name]

>docker run Ubuntu

executing a docker image in the local server

docker run -it <container id / name> bash

running a ubuntu container in interactive mode…. Using bash commands to ammend   
 the existing image



B.Pushing the images from local repository to docker hub

Tag and push the imge:  
docker push <image name> push the image to the docker hub repository

C.stops running container

docker stop containerid

>docker stop 96724fc65bd8

1. remove container

>docker rm 96724fc65bd8

Answer short questions related to Docker commands, image vs. container, etc

**QV.Docker Compose**

**Task:** Write a docker-compose.yml file for a multi-container setup.

1. **Service 1** – Demonstarte your app on 9090
2. **Service 2** – Configure a database container (choose WordPress /MySQL/PostgreSQL/MongoDB Ensure both containers are running together.

Specify the username, password, and database name required for the your system.

1. Demonstrate service startup with docker-compose up

**Example : Docker compose of myproject\_app(Myproject name ) and MySql**

**Run your app on 9090**

**version: "3.9"**

**services:**

**app:**

**image: myprojectimage**

**container\_name: myproject\_app**

**ports:**

**- "9090:8080" # HostPort:ContainerPort, App accessible at http://localhost:9090**

**environment:**

**DB\_HOST: db**

**DB\_USER: user**

**DB\_PASSWORD: password**

**DB\_NAME: mydb**

**depends\_on:**

**- db**

**db:**

**image: mysql:8**

**container\_name: myproject\_db**

**environment:**

**MYSQL\_ROOT\_PASSWORD: rootpassword**

**MYSQL\_DATABASE: mydb**

**MYSQL\_USER: user**

**MYSQL\_PASSWORD: password**

**volumes:**

**- db\_data:/var/lib/mysql**

**ports:**

**- "3306:3306" # Optional, only if you want to connect from host**

**volumes:**

**db\_data:**

You have:

**Your own project image** (let’s say myprojectimage)

**A database image** (let’s say mysql:8 or postgres)

Now you want to **compose them together** with Docker Compose and run them.

How to Run

Save the file as docker-compose.yml in your project folder.

In terminal, go to that folder.

**Run:**

docker-compose up -d

This will start both app (your image) and db (MySQL).

Check if services are running:

**docker-compose ps**

Access your app at:  
[http://localhost:9](http://localhost:8080/)090

## Stopping and Restarting

**Stop all:**

docker-compose down

**Start again:**

docker-compose up -d

## Scale a Service (Multiple Instances)

Example: Run 2 WordPress containers:

Ex: scaling one of the services(database) to 2 **instances**.

**--scale = Run multiple instances of the same service, all managed by Docker Compose.**

Ex: docker-compose up --scale wordpress=2 -d

1. docker-compose up → starts the services defined in your docker-compose.yml.
2. --scale wordpress=2 → tells Docker Compose to create **2 replicas** (containers) of the service named **wordpress**.
3. -d → runs the containers in **detached mode** (in the background).