**Software Requirement Specification (SUS)**

**The Hospital Management System (HMS) comprehensive application designed in my manage patient reconds, appointments, medical staff, and haloperations. The piks secure sorage of sensitive medical and administrative data, ensuring confidentiality and compliance wit healthcare standards**

**Based in this scenario, amover the following 110 Marka)**

**AM**

**Functional Requirements-21**

**Non-Functional Requirement 2M**

**d. Idemification of es-M**

**Q2. Maven Java Application Development (30M)**

**You are tired as a Software Engineer at a tech startup. Your first tasking and setting up a Maven-baved lava application thit your team will continue developing**

**You are given the following GitHub repository link** [**https://github.com/Komblumbhargavi75/HospitalMano**](https://github.com/Komblumbhargavi75/HospitalMano) **vitam**

**Solve the following related to the ponicunt file and update it accordlingly and finally executi**

**1. Download the given repository and show the bit of the files-3M**

**2 myn CLEAN package, when I run this gating an unknown lifecycle phad arrin why-1M**

**3. Add the dependency servlet-api of 2.5 to your project -2M**

**4. You try to build the project with JDK 21. hut compilation faila Wity swight His happen? How**

**in pom.xml-2M**

**5. In the dependency sectiondependency**

**groupld-SE/groupld**

**artifactId>junit/artifactId**

**Versing 4.6.0/verw**

**dependency What will Maven do in this case? How can you solet-M**

**6 . After building the WAll you notice de generated file is at sta SNAPSHOT war instead of HospitalMgist System was to fit 3M**

📝 Q1. Software Requirement Specification (SRS) for Hospital Management System (HMS)

Think of HMS like a hospital app. It helps to store patient details, book appointments, manage doctors and staff, and generate reports.

✅ a. Functional Requirements (What the system should DO)

Add, update, and delete patient details.

Book, cancel, and reschedule appointments.

Maintain records of doctors and nurses.

Store patient medical history and reports.

Generate bills and manage payments.

Secure login for patients, doctors, and admin.

✅ b. Non-Functional Requirements (How the system should WORK)

Security – Patient data should be private (passwords, encryption).

Performance – Should handle many users at the same time.

Scalability – Should work even if the hospital grows bigger.

Usability – Easy to use interface for doctors, patients, and staff.

Reliability – System should not crash often.

✅ c. Identification of Actors (Who uses the system)

Patient – books appointments, views reports, pays bills.

Doctor – checks patients, writes prescriptions.

Nurse/Staff – updates patient care details, helps scheduling.

Admin – manages users, doctors, and overall system.

Pharmacist – gives medicines based on prescriptions.

Lab Staff – uploads test results.

**🖥️ Q2. Maven Java Application Development**

Now let’s make a Java project with Maven. Maven is just a tool that helps us build, run, and manage Java projects.

**✅ Step 1: Download the GitHub project**

Open terminal / command prompt and type:

git clone <https://github.com/Komblumbhargavi75/HospitalMano>

cd HospitalMano

Now you will see files like:

pom.xml

src/main/java

src/test/java

**✅ Step 2: Clean and Package**

Run:

mvn clean package

⚠️ If you type mvn CLEAN package → error occurs because commands are lowercase only.

**✅ Step 3: Add servlet-api dependency**

In the pom.xml file, inside <dependencies> add this:

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>servlet-api</artifactId>

<version>2.5</version>

<scope>provided</scope>

</dependency>

**✅ Step 4: Fix JDK 21 Issue**

Some libraries are old and don’t work with JDK 21.

Add this inside pom.xml:

<properties>

<maven.compiler.source>1.8</maven.compiler.source>

<maven.compiler.target>1.8</maven.compiler.target>

</properties>

👉 This tells Maven to use Java 8 compatibility.

**✅ Step 5: Fix Wrong JUnit Dependency**

Wrong:

<dependency>

<groupld>SE</groupld> <!-- spelling mistake -->

<artifactId>junit</artifactId>

<version>4.6.0</version>

</dependency>

Correct:

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.6</version>

<scope>test</scope>

</dependency>

**✅ Step 6: Change WAR File Name**

By default, Maven generates:

HospitalMano-1.0-SNAPSHOT.war

If you want the file name to be HospitalManagementSystem.war, add this in pom.xml:

<build>

<finalName>HospitalManagementSystem</finalName>

</build>

🎯 Final pom.xml (Complete and Correct)

Here is the full corrected file you can use directly:

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.hospital</groupId>

<artifactId>HospitalManagementSystem</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>war</packaging>

<properties>

<maven.compiler.source>1.8</maven.compiler.source>

<maven.compiler.target>1.8</maven.compiler.target>

</properties>

<dependencies>

<!-- Servlet API -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>servlet-api</artifactId>

<version>2.5</version>

<scope>provided</scope>

</dependency>

<!-- JUnit for Testing -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.6</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<finalName>HospitalManagementSystem</finalName>

</build>

</project>

**✅ That’s it.**

**7. Add the central dependency of Java Servlet API-4.0.0-601 to your existing project check complete pom.xml file and run it " -2M**

**8. A developer removes the <dependencies> section completely. Will Maven still build the project? What issues might occur during testing7-2M**

**9. Failed to execute goal [32morg.apache maven plugins maven-compiler-plugin:3.13.0 compile(m [Im(default-compile)[m on project [imiprojectdemo[m: [1:31mCompilation failure[m [ERROR] [1:31mNo compiler is provided in this environment. Perhaps you are**

**running on a JRE rather than a JDK? Identify the error?-2M**

**10. In Build is having finalname as finalName localhost:8080/FoodSystem</finalName, is it works, if not how can you fix it7-2M**

**11. Your project is meant to deploy on Tomcat, but <pakagingjar</pakaging is like this inpom.xml.How do you solve it? -2M**

**12. In the <arl tag, Written <url>http://maven.java.org12/url>**

**Will Maven accept this? What is the correct purpose of the curl element in pom.xml?--3M**

**13. Check the complete pom.xml and push into the github of your account.-3M**

**Q3. Git & GitHub Integration with Maven Project (30M)**

**Using the Maven project created above, perform the following Git tasks:**

**1. You made changes to one file in the above project but haven't staged them yet. You realizethey were a mistake. What Git command will you use to discard the local changes-2M**

**2. You made a commit hut typed the wrong commit message. You haven't pushed it yet. How do you Fix 17-2M**

**3. You want to view the commit history of the current branch in a readable format. What Git command should you use 2M**

**4. You're on the main branch, Create the branch Featuer/patientand switch to that branch. What commands you use?-2M**

**5. You've made some commits locally and now want to upload them to the remote repository. What do you run?-2M**

**6. You want to see all the branches that exist both locally and on the remote. How7-2M Create a branch Suggestions and merge with patient branch, how can it he? -3M**

**8. How do you pull the latest changes from the remote repository and merge them into your local branch-2M**

**9. Specify the git command to when pushing for the first time and want to set the remate branch -2M**

**10. You cloned a remote repository, but later you find that you need to push your changes to a different remote repository How do you configure your local repository to push to this new remote-3M**

**11. After running git pull, you notice that your local branch is behind the remote branch. How would you proceed to bring your local branch up to date without losing your local changes-3M**

**12. You've pushed a patient branch to a remote repository, but now you need to delete the branch from the remote, How would you do that?-2M**

**13. How do you apply a patch file provided by your teanmate and include it in your commit history-3M**

**Q4. Docker containerization for Maven Application-20M (10-2-20)**

**Task: Containerize your Maven project using Docker.**

**a. Refer the given link/documentation to write a Dockerfile for the Maven project**

[**https://github.com/Kumbhambhargavi75/1HospitalMgmtSystem**](https://github.com/Kumbhambhargavi75/1HospitalMgmtSystem)

**Ensure it copies the WAR/JAR and runs on Tomcat (or relevant base image) give every question in a very detailed way. i dont know anything**

**7) Add the central dependency of Java Servlet API 4.0.0 (the requested 4.0.0-601 looks non-standard) — update POM and run (2M)**

Short answer: use the official servlet API dependency and mark it provided (Servlet container like Tomcat provides it at runtime). I couldn’t find an artifact named 4.0.0-601 on Maven Central — the standard artifact is javax.servlet:javax.servlet-api:4.0.0 or the newer jakarta.servlet:jakarta.servlet-api:4.x. Use the javax.servlet 4.0.0 if your code targets Servlet 4.0.

Maven Central

Dependency to add inside <dependencies>:

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>javax.servlet-api</artifactId>

<version>4.0.0</version>

<scope>provided</scope>

</dependency>

How to check / run:

Put that dependency inside your pom.xml <dependencies>...</dependencies>.

Run:

mvn clean package

You should get target/YourArtifactId-1.0-SNAPSHOT.war (or whatever <finalName> you configured).

**8) A developer removes the <dependencies> section completely. Will Maven still build? Issues during testing? (2M)**

Will Maven build?

Maybe. Maven will still run, but compilation will fail if your code refers to any external classes (like javax.servlet.\*, junit, Spring, etc.). If your project only uses the JDK classes and has no external libs, mvn package may succeed.

Issues you’ll see:

Compile errors (missing types).

Unit tests fail because testing frameworks (JUnit) are no longer on classpath.

Plugins or integration steps expecting dependencies may fail.

Fix: restore the <dependencies> block and re-add the required dependencies (copy from a working backup or add the servlet & junit dependencies shown above).

**9) Identify the error:**

Failed to execute goal org.apache.maven.plugins:maven-compiler-plugin:3.1:compile

... Compilation failure ... No compiler is provided in this environment. Perhaps you are running on a JRE rather than a JDK?

Cause & fix (2M)

Cause: Maven needs the Java compiler (javac). That error shows your environment uses a JRE (Java Runtime) instead of a JDK (Java Development Kit) or JAVA\_HOME points to a JRE. Maven cannot compile without the JDK. (This is a very common error.)

Stack Overflow

Roufid

Fix (quick):

Install a JDK (e.g., OpenJDK 11 or 17).

Ensure javac exists:

javac -version

java -version

javac must print a version (not an error).

Set JAVA\_HOME to JDK location and add $JAVA\_HOME/bin to PATH.

Linux / macOS (example for openjdk11):

export JAVA\_HOME=/usr/lib/jvm/java-11-openjdk

export PATH=$JAVA\_HOME/bin:$PATH

Windows (PowerShell example):

setx JAVA\_HOME "C:\Program Files\Java\jdk-11.0.x"

$env:PATH = $env:JAVA\_HOME + "\bin;" + $env:PATH

Re-run:

mvn -v # check Maven sees correct Java

mvn clean package

In IDEs (Eclipse/IntelliJ) also check the Java runtime used by the IDE and Maven integration.

**10) If <finalName> is set to localhost:8080/FoodSystem — will it work? How to fix? (2M)**

No — that won't work. <finalName> must be a filename (no URL, no slashes). Maven will try to create an artifact with that literal name and it will be invalid or cause unexpected results.

What it should be: just the WAR file base name, e.g.

<build>

<finalName>FoodSystem</finalName>

</build>

This produces FoodSystem.war. To make the app accessible at <http://localhost:8080/FoodSystem>, deploy FoodSystem.war to Tomcat webapps.

If you want the app at root / rename the WAR to ROOT.war or use Tomcat context configuration (Tomcat decides URL path from WAR name or context config). See Tomcat docs about context path and ROOT.war.

Apache Tomcat

Baeldung on Kotlin

**11) Project meant to deploy on Tomcat but <packaging>jar</packaging> — how to solve? (2M)**

Change packaging to war in pom.xml:

<packaging>war</packaging>

Why: Tomcat expects a WAR (Web Application Archive). jar builds a normal Java library/application, not a web archive.

Important notes after switching:

Keep the servlet API dependency with <scope>provided</scope> so Tomcat supplies it at runtime.

Ensure your webapp folder layout exists: src/main/webapp/WEB-INF/web.xml (or use Servlet annotations).

References: Maven packaging types and their lifecycles.

Baeldung on Kotlin

+1

**12) <url>http://maven.java.org12/url> — will Maven accept it? Purpose of the <url> element? (3M**)

Will it accept it? No — your XML is malformed (closing tag missing < and > in your example) and the URL looks invalid (maven.java.org12 is not a correct hostname). Maven requires well-formed XML.

Correct syntax & purpose:

<url>http://example.com/your-project-homepage</url>

The <url> element is optional and simply points to the project’s homepage (used in generated reports and documentation). It is for humans and tooling to know where the project lives.

Apache Maven

**13) Check the complete pom.xml and push into your GitHub account (3M)**

Steps (assume repo already cloned and you edited pom.xml):

# check status

git status

# stage changes (pom.xml and anything else you changed)

git add pom.xml

# commit

git commit -m "Fix pom: add servlet-api, set packaging=war, set finalName"

# if you cloned from original repo and want to push to YOUR GitHub:

# 1) create a new empty repo in your GitHub account (via GitHub website), call it hospital-mgmt

# 2) then set remote and push:

git remote add myorigin [https://github.com/<your-github-username>/hospital-mgmt.git](https://github.com/%3cyour-github-username%3e/hospital-mgmt.git)

git push -u myorigin main # or master, depending on your branch name

If the repo already has origin and you want to push to that remote, use git push origin main (or whichever branch you are on).

**Q3 — Git & GitHub Integration with Maven Project (30M)**

I’ll answer each numbered Git question with the precise command(s) and a short explanation.

You changed one file but haven't staged it yet and want to discard local changes (2M)

# modern command

git restore path/to/file

# older syntax works too

git checkout -- path/to/file

This resets that file to the last committed state.

**You made a commit but typed the wrong commit message (haven't pushed yet). Fix it (2M)**

git commit --amend -m "Correct commit message"

This replaces the last commit message (and last commit). Don't --amend after pushing to a shared remote unless you coordinate with teammates.

View commit history of current branch in readable format (2M)

git log --oneline --graph --decorate --all

This gives a compact, visual history.

You're on main. Create branch Feature/patient and switch to it (2M)

git checkout -b Feature/patient

# or

git switch -c Feature/patient

You made commits locally and want to upload them to remote (2M)

git push origin Feature/patient

If it’s the first time pushing that branch and you want to set upstream:

git push -u origin Feature/patient

**See all branches both locally and on the remote (2M)**

git branch -a

git branch shows local; -a adds remote branches as remotes/origin/....

**Create branch Suggestions and merge it into Feature/patient (3M)**

# create and switch to Suggestions

git checkout -b Suggestions

# make changes, commit them:

git add .

git commit -m "Add suggestions"

# switch to target branch

git checkout Feature/patient

# merge

git merge Suggestions

# push the merged result

git push origin Feature/patient

**How do you pull the latest changes from the remote and merge into your local branch? (2M)**

# simple way (fetch+merge)

git pull origin Feature/patient

# safer two-step (recommended)

git fetch origin

git merge origin/Feature/patient

Many prefer git pull --rebase to avoid unnecessary merge commits.

Pushing for the first time and want to set the remote branch (set upstream) (2M)

git push -u origin Feature/patient

After that you can simply git push and git pull.

**You cloned a repo but later need to push changes to a different remote repo — how to configure (3M)**

Add a new remote and push there:

git remote add new-origin [https://github.com/<your-user>/<new-repo>.git](https://github.com/%3cyour-user%3e/%3cnew-repo%3e.git)

git push -u new-origin main

Or change the existing origin URL:

git remote set-url origin [https://github.com/<your-user>/<new-repo>.git](https://github.com/%3cyour-user%3e/%3cnew-repo%3e.git)

git push -u origin main

**After git pull, local branch is behind remote — bring local up to date without losing local changes (3M)**

Two clean ways:

A) Rebase your local commits onto updated remote (recommended):

git fetch origin

git rebase origin/your-branch

B) Stash your local work, pull, then reapply:

git stash push -m "WIP"

git pull origin your-branch

git stash pop

Use rebase for a linear history; stash is fine for uncommitted work.

**Delete a branch from the remote (2M)**

git push origin --delete Feature/patient

# or older style

git push origin :Feature/patient

Apply a patch file provided by teammate and include it in your commit history (3M)

If your teammate gave a patch created by git format-patch or git diff > file.patch:

If patch is a git mbox/format-patch (series):

git am < 0001-some-change.patch

git am applies commits from mailbox patch and preserves author/commit message.

If patch is a plain unified diff (.patch):

git apply fix.patch # apply changes to working tree

git add .

git commit -m "Apply patch from teammate: fix.patch"

If you want to preserve author/commit: use git am format or ask for format-patch.

**Q4 — Docker containerization for Maven Application (20M)**

Goal: Build the Maven WAR and run it on Tomcat inside Docker. I’ll give one simple, clear Dockerfile (multi-stage) that builds with Maven and then runs with Tomcat.

Approach: Multi-stage build — first stage uses a Maven image to compile & package your project; second stage uses an official Tomcat image and copies the WAR into webapps/. Tomcat will deploy it. (This is the usual, recommended approach.)

Baeldung on Kotlin

cprime.com

A. Example Dockerfile (multi-stage)

Put this Dockerfile at your project root (where pom.xml is):

# Stage 1: Build with Maven

FROM maven:3.8.8-jdk-11 AS build

WORKDIR /usr/src/app

# copy only maven files first to cache dependencies

COPY pom.xml .

COPY src ./src

# build the WAR (skip tests to speed up; remove -DskipTests to run tests)

RUN mvn -B clean package -DskipTests

# Stage 2: Run with Tomcat

FROM tomcat:9.0-jdk11-openjdk

# Remove default webapps if you want a clean container (optional)

RUN rm -rf /usr/local/tomcat/webapps/\*

# copy the WAR generated by maven into tomcat as ROOT.war (or FoodSystem.war)

# Adjust path: target/<your-artifact>-<version>.war

COPY --from=build /usr/src/app/target/HospitalManagementSystem.war /usr/local/tomcat/webapps/ROOT.war

EXPOSE 8080

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

Notes:

I copy the WAR to ROOT.war to make the app available at <http://localhost:8080/>. If you want <http://localhost:8080/FoodSystem>, copy it as FoodSystem.war instead and access /FoodSystem/.

If your artifact name differs, replace HospitalManagementSystem.war with the real filename in target/.

You can remove -DskipTests to run tests in the image (slower).

Build and run commands:

# build image

docker build -t hospital-app:1.0 .

# run container (map container 8080 to host 8080)

docker run --rm -p 8080:8080 --name hospital-app hospital-app:1.0

Then open <http://localhost:8080/> in the browser (or /FoodSystem/ if you named war accordingly).

If you already have a WAR locally: you can skip the Maven stage and use a simpler Dockerfile:

FROM tomcat:9.0-jdk11-openjdk

COPY target/HospitalManagementSystem.war /usr/local/tomcat/webapps/ROOT.war

EXPOSE 8080

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

How Tomcat determines context path: Tomcat decides the web app URL from the WAR filename or META-INF/context.xml. To deploy to root (/) you can name the WAR ROOT.war, or configure Tomcat context files. See Tomcat docs.

Apache Tomcat

B. Extra tips and troubleshooting

If you see 404 after deploying: ls /usr/local/tomcat/webapps/ inside container to confirm WAR is present; check Tomcat logs (docker logs <container>).

Permissions: if file copy fails, ensure Dockerfile paths are correct and file actually exists in target/.

Smaller images: you can use tomcat:9-jdk11-temurin or other variants — check Docker Hub for available tags.

Build locally for debugging: run mvn clean package on your machine first, then test WAR locally on Tomcat before containerizing.

Quick checklist (copy-paste) — to finish everything

Edit pom.xml: set <packaging>war</packaging>, add servlet dependency (provided), set <finalName>HospitalManagementSystem</finalName> (or FoodSystem).

Build locally:

mvn clean package

# check target/\*.war exists

ls target

Docker build & run:

docker build -t hospital-app:1.0 .

docker run --rm -p 8080:8080 hospital-app:1.0

Push your code to GitHub:

git add pom.xml Dockerfile

git commit -m "Add servlet dependency, change packaging to war, add Dockerfile"

git push -u origin main

Useful references I used (for learning / verification)

how to declare javax.servlet-api:4.0.0 dependency on Maven Central.

Maven Central

common fix for No compiler is provided ... (you need JDK and correct JAVA\_HOME).

Stack Overflow

Roufid

Maven packaging types (jar / war / pom) and lifecycles.

Baeldung on Kotlin

<url> element purpose in pom.xml.

Apache Maven

Tomcat context path rules (ROOT.war, naming rules) and deployment behavior.

Apache Tomcat

Docker + Tomcat: copy WAR into /usr/local/tomcat/webapps and Tomcat will deploy it (multi-stage recommended).

Baeldung on Kotlin

cprime.com

**1 You have written the above Dockerfile**

**Which CLI command will you use to build the Docker image with the name hospital-mgint**

**2. After building the image, you want to list all available Docker images on your machine**

**Which command will you run**

**3. You need to run the container and map container port 8080 to host port 9090.**

**Write the full Docker run command.**

**4. pull the officialredis image from Docker Hub, run it in a container and list of containers that are running right now and verify whether it is accessible on** [**http://localhost:8080**](http://localhost:8080)

**5. After festing locally, you want to tag and push the image to Docker Hub under your account yourusername/ms**

**6. While running the container, you forgot to expose port mapping. Now you cannot access the app from your browser. What steps/commands will you take to stop the container and rerun it correctly?**

**7. The container crashes immediately after startup.**

**Which command will you use to inspect logs and debug the issue?**

**8. Push your custom Docker image (with the deployed application) to your Docker Hub account and make it available publicly**

**9. You want to connect into the running container interactively to check files Which CLI command will you use to open a shell inside the container?**

**10, you want to list the removed containers Which Docker CLI commands will you use?**

**05. DOCKER COMPOSE-10M**

**Write the compose file for CampusMgmtSystem application and MongoDB and service with docker-compose -SM**

**2. Container 1-Use the Docker image of your application you pushed to Docker Hub and run it on**

**7079-3M**

**3. Container 2 Runs a MongoDB container as the database and work on that-2M give every question in a very detailed way. i dont know anything**

**1) Build the Docker image called hospital-mgint**

Assuming your Dockerfile is in the current directory (project root):

# build and tag as hospital-mgint:latest

docker build -t hospital-mgint:latest .

What this does:

docker build reads Dockerfile and builds an image.

-t hospital-mgint:latest gives the image the name (repository) hospital-mgint and tag latest.

. means build context is current directory.

Optional: To include your Docker Hub username (recommended when pushing):

docker build -t yourusername/hospital-mgint:1.0 .

**2) List all Docker images on your machine**

docker images

# or the newer syntax:

docker image ls

What you’ll see: columns REPOSITORY, TAG, IMAGE ID, SIZE — look for hospital-mgint (or yourusername/hospital-mgint).

**3) Run the container mapping container port 8080 → host port 9090**

Full run command (detached, named container, port mapping):

docker run -d --name hospital-mgint-container -p 9090:8080 hospital-mgint:latest

Explanation:

-d runs container in background.

--name hospital-mgint-container gives the container a friendly name.

-p 9090:8080 maps host port 9090 to container port 8080. So open <http://localhost:9090> in browser.

hospital-mgint:latest is the image to run.

If the container uses Tomcat and you want it accessible at host 9090, that mapping is correct.

If you prefer to remove container when stopped (no leftover containers):

docker run -d --rm --name hospital-mgint-container -p 9090:8080 hospital-mgint:latest

**4) Pull official redis image, run it, list running containers, and verify HTTP access at** [**http://localhost:8080**](http://localhost:8080)

Commands:

# pull latest redis image

docker pull redis:latest

# run redis in background and map default redis port 6379

docker run -d --name redis-server -p 6379:6379 redis:latest

# list running containers

docker ps

Important note about HTTP

Redis does not speak HTTP. Redis exposes a TCP protocol on port 6379. So <http://localhost:8080> will NOT work for Redis.

How to verify Redis is working:

From host (if you have redis-cli installed):

redis-cli -h localhost -p 6379 ping

# expected response: PONG

Or use a one-off container to run redis-cli:

docker run --rm -it --network host redis:latest redis-cli -h 127.0.0.1 -p 6379 ping

# or if not using host networking:

docker exec -it redis-server redis-cli ping

To check mapped ports:

docker ps --format "table {{.Names}}\t{{.Image}}\t{{.Ports}}"

So answer: you can confirm Redis with redis-cli/PING — not with HTTP.

**5) Tag and push the image to Docker Hub as yourusername/ms**

Steps:

Log in to Docker Hub (interactive — supply username/password or use token):

docker login

Tag image locally with Docker Hub repo name:

# If your local image is hospital-mgint:latest

docker tag hospital-mgint:latest yourusername/ms:latest

(Replace yourusername.)

Push to Docker Hub:

docker push yourusername/ms:latest

Notes:

You must have a Docker Hub account; if repo yourusername/ms does not exist, Docker Hub will create it (or you may need to create it on the website depending on settings).

After push succeeds, image will be available at <https://hub.docker.com/r/yourusername/ms>.

**6) You forgot port mapping and cannot access app — stop and rerun correctly**

If container is running without port mapping:

List running containers to get name/ID:

docker ps

Stop and remove the container:

docker stop <container-name-or-id>

docker rm <container-name-or-id>

Example:

docker stop hospital-mgint-container

docker rm hospital-mgint-container

Re-run with proper port mapping:

docker run -d --name hospital-mgint-container -p 9090:8080 yourusername/ms:latest

(Or hospital-mgint:latest if using local image.)

Tip: You cannot change port mapping of an already-running container — you must recreate it.

7**) Container crashes immediately after startup — inspect logs and debug**

Check status (all containers including exited):

docker ps -a

Get logs for the container:

docker logs <container-name-or-id>

# follow logs live:

docker logs -f <container-name-or-id>

Other helpful debugging commands:

Inspect container metadata:

docker inspect <container-name-or-id>

Run container interactively to see startup output (replace your-image):

docker run --rm -it --entrypoint /bin/sh your-image

# or to see startup, run without -d:

docker run --name debug-run -p 9090:8080 your-image

If it's a Java/Tomcat app, docker logs will show stacktrace — missing libs, wrong JAVA\_HOME, port in use, or config errors are common causes.

**8) Push your custom Docker image (deployed app) to Docker Hub and make it public**

(Combines steps from #5 plus making it public.)

Commands:

# login

docker login

# tag local image to your hub repo

docker tag hospital-mgint:latest yourusername/ms:latest

# push to Docker Hub

docker push yourusername/ms:latest

Make repo public:

Sign in to Docker Hub web UI → go to your repository yourusername/ms → Settings → set visibility to Public (if not already public).

(This step is done in the Docker Hub website UI; CLI does not toggle visibility.)

**9) Open an interactive shell inside a running container**

If container is running and has bash:

docker exec -it <container-name> /bin/bash

If bash is not available (alpine or minimal images), use sh:

docker exec -it <container-name> /bin/sh

Example:

docker exec -it hospital-mgint-container /bin/bash

This opens a shell inside container so you can ls, inspect config, check log files, etc.

If container is not running, you can start a temporary container with an interactive shell:

docker run --rm -it yourusername/ms:latest /bin/sh

**10) List the removed containers**

Important: removed containers are gone — Docker does not keep a list of containers that have been deleted. But you can:

List all containers (running + stopped/exited):

docker ps -a

List only stopped/exited containers:

docker ps -a -f "status=exited"

# or

docker ps -f "status=exited"

If you want to remove (delete) all stopped containers:

docker container prune

If by “removed containers” you mean dangling images (untagged images), list them:

docker images -f "dangling=true"

If you need a historical record of container events (start/stop/delete), inspect Docker event log (streaming):

docker events --since '24h'

But once a container is removed, docker ps -a will not show it.

DOCKER COMPOSE (10M) — docker-compose.yml for CampusMgmtSystem + MongoDB

You asked:

Compose file for CampusMgmtSystem app and MongoDB.

Container 1: Use your app image (from Docker Hub) and run it on host port 7079.

Container 2: MongoDB container as database.

Here’s a simple, complete docker-compose.yml that does exactly that.

version: "3.8"

services:

campus-app:

image: yourusername/ms:latest # <-- replace with your Docker Hub image

container\_name: campus-app

restart: always

ports:

- "7079:8080" # host:container (assuming app listens on 8080 in container)

environment:

- SPRING\_DATASOURCE\_URL=mongodb://mongo:27017/campusdb

- MONGO\_URI=mongodb://mongo:27017/campusdb

# add any other env vars your app expects, e.g. DB user/password

depends\_on:

- mongo

networks:

- campus-net

mongo:

image: mongo:6.0 # official MongoDB image

container\_name: campus-mongo

restart: unless-stopped

ports:

- "27017:27017"

volumes:

- ./mongo-data:/data/db # persist DB on host

environment:

MONGO\_INITDB\_DATABASE: campusdb

networks:

- campus-net

healthcheck:

test: ["CMD", "mongo", "--eval", "db.adminCommand('ping')"]

interval: 10s

timeout: 5s

retries: 5

networks:

campus-net:

driver: bridge

How it works (simple)

campus-app uses your Docker Hub image yourusername/ms:latest. Replace that with the image you pushed.

It maps host port 7079 to container port 8080 (adjust container port if your image uses a different internal port).

mongo runs MongoDB, stores data in ./mongo-data on the host so DB persists across restarts.

depends\_on ensures Docker starts mongo before campus-app. (It does not guarantee Mongo is ready — we added a healthcheck to improve readiness.)

Commands to run

# start services in background

docker-compose up -d

# follow logs

docker-compose logs -f

# stop and remove containers (and network), but keep volumes

docker-compose down

# stop and remove containers, network and volumes

docker-compose down -v

Verify

Check containers:

docker-compose ps

# or

docker ps

Access app in browser:

<http://localhost:7079>

(or **[Error! Hyperlink reference not valid.](http://<VM-IP>:7079)** if running on remote host.)

Check Mongo is up:

docker exec -it campus-mongo mongo --eval "db.stats()"

Final tips & checklist (so you don’t get lost)

Replace yourusername with your Docker Hub username wherever I wrote it.

Always docker login before docker push.

To check port mapping quickly:

docker ps --format "table {{.Names}}\t{{.Ports}}"

If app uses a different internal port (not 8080), change the :8080 to that port in all -p HOST:CONTAINER mappings and in docker-compose.

If using Windows and Docker Desktop, these commands still work in PowerShell/Git Bash.

If containers fail to start, docker logs <name> is your first friend.

**------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

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

**The cafeteria management wants a Food Ordering System that enables students and faculty to order food online, make digital payments, and receive real-time order status updates. The system should dine-in and take-away options while allowing administrators to manage the menu support both**

**Based on this scenario, answer the following:[10 Marka)**

**a. Abstract-4M**

**b. Functional Requirements-2M**

**c. Non-Functional Requirements-2M**

**d. Identification of Users-2M**

**QII: Maven Web Application Development (30M)**

**You are hired as a Software Engineer at a tech startup. Your first task involves understanding and setting up a Maven-based Web application that your team will continue developing**

**You are given the following GitHub repository link**

[**https://github.com/Kumbhambhargavi75/FoodSystem**](https://github.com/Kumbhambhargavi75/FoodSystem)

**Solve the following related to the pom.xmi file and update it accordingly and finally exécute it**

**1. The finalName in the build section is written as SfinalName>Food-System/finalName> What artifact will be generated in the target folder, and why might this cause deployment issues in Tomcat? -2M**

**2. Download the given repository and show the list of the files?-2M**

**3. In the dependency section, the version of JUnit is left out like this:**

**<dependency>**

**<groupId>junit</groupId>**

**<artifactId>junit</artifactId>**

**</dependency>>What will Maven do in this case? How can you ensure a proper version is used-3M**

**4. What kind of error will Maven throw during mvn install? How do you debug such XML. mistakes-2M**

**Suppose you misspelled <artifactId> as <artificatld> in the <dependency block**

**5. You written like this <packing> tag in the POM**

**What does Maven assume by default, and what impact will it have on this web project7-2M**

**6. You add an extra dependency:**

**<dependency>**

**<groupId>mysql</groupId>**

**<artifactId>mysql-connector/artifactId>**

**<version>8.0.30</version>**

**</dependency> give every question in a very detailed way from scartch like iam a child**

) Abstract — 4M

The Food Ordering System is a web application for the campus cafeteria that lets students and faculty browse the menu, place orders (dine-in or take-away), pay digitally, and receive live order status updates. Administrators can update menu items, prices and view order history. Kitchen staff see incoming orders and update their preparation status. The system must be simple to use, secure for payments, and fast enough to handle many simultaneous orders during peak hours.

b) Functional Requirements — 2M (what the system must do)

Write these as simple short items:

User registration & login (students, faculty) — create account, login, logout.

Browse menu — view categories (breakfast, lunch, snacks), item details and images.

Place order — choose Dine-in or Take-away, add items, select quantity, checkout.

Digital payments — support card/UPI/college wallet; record payment status.

Order status — show real-time updates: Pending → Preparing → Ready → Completed.

Admin panel — add/edit/remove menu items, change prices, view/resolve orders.

Kitchen view — list of active orders with ability to change status and mark ready.

(Each item is a single, testable function.)

c) Non-Functional Requirements — 2M (how well it must work)

Performance: Page loads < 2s; system supports X concurrent users (define X for your college).

Availability: 99% uptime during cafeteria hours.

Security: HTTPS, secure storage of payment tokens (no raw card data), role-based access control.

Usability: Clean UI, mobile responsive, accessible (large fonts, clear buttons).

Maintainability: Clear code structure and documentation so future devs can update menu/features.

d) Identification of Users — 2M

List user roles and short description:

Student / Faculty (Customer): Orders food, selects dine-in/takeaway, pays, checks status.

Cafeteria Admin: Manages menu, prices, view reports and orders.

Kitchen Staff: Sees incoming orders, updates order status (Preparing / Ready).

System Administrator / DevOps: Manages deployments, backups, DB, and system security.

Payment Gateway (external): Third-party service that processes payments.

QII — Maven Web Application Development (30M)

You were given: <https://github.com/Kumbhambhargavi75/FoodSystem> — I checked the repo listing and copied the top-level files here for you.

1) finalName typo:

The finalName in the build section is written as SfinalName>Food-System/finalName> — What artifact will be generated in the target folder, and why might this cause deployment issues in Tomcat? — 2M

What finalName does (simple):

<finalName> controls the filename / folder under target/ that Maven will produce for your build (for webapps it controls the WAR name / exploded folder). The exploded WAR directory defaults to target/<finalName>.

Apache Maven

Two possible situations (what that broken text means):

If the tag is intended and correct: e.g.

<build>

<finalName>Food-System</finalName>

</build>

Maven will create target/Food-System.war (or target/Food-System/ for exploded war). Tomcat will use the WAR filename to decide the web context (/Food-System), so your app URL would be <http://host:8080/Food-System>. If you expected a different context (like root / or /cafeteria) this mismatch causes deployment surprises.

Apache Maven

Octopus Deploy

If the tag is actually malformed (like the snippet SfinalName>Food-System/finalName> that is missing < or </ or is mistyped): Maven may fail to read the POM (XML not well-formed) or will ignore a wrong tag and fall back to defaults (artifactId-version.war). If Maven falls back, the artifact will be target/<artifactId>-<version>.war and not the name you wanted — again Tomcat context will be different. If the POM is malformed XML you will get a non-parseable POM error and the build stops.

Stack Overflow

Apache Maven

Why this can cause Tomcat deployment issues:

Wrong WAR name → wrong context path (Tomcat maps WAR filename to context). If you expected to access / or /FoodSystem but mvn produced a different name, Tomcat will expose a different URL.

Octopus Deploy

If POM is malformed, Maven build fails and no WAR is produced at all (so nothing to deploy).

How to fix (very simple): open pom.xml, find <build> and ensure you have:

<build>

<finalName>Food-System</finalName>

</build>

Then run:

mvn clean package

# check target/ for Food-System.war

ls -la target/

2) Download repository and list files — 2M

How to download (one command):

git clone <https://github.com/Kumbhambhargavi75/FoodSystem.git>

cd FoodSystem

ls -la

(If you don’t have git, download ZIP from GitHub page.)

Files/folders I can see at repo root (top level):

.settings

src/main/webapp

target

.classpath

.project

Dockerfile

pom.xml

(These items come from the repo file list shown on GitHub.)

GitHub

If you cd into the project and run tree you’ll see full structure (install tree or use ls -R).

Commands to view pom.xml:

cat pom.xml # print to terminal

3) Missing JUnit <version> — what Maven does & how to ensure correct version — 3M

What happens if version is missing:

Maven requires a version for a dependency unless the version is provided by a parent POM or a <dependencyManagement> (BOM or parent). If the version is missing and not managed, Maven will complain (build fails) — common error: dependencies.dependency.version is missing or similar. In short: Maven can’t pick a version by magic (unless inherited/managed).

Stack Overflow

Apache Maven

How to make sure a proper version is used (simple steps):

Quick fix (explicit): add a version in your dependency:

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

(That ensures Maven downloads JUnit 4.13.2.)

Better in multi-module projects: put versions centrally in a parent POM’s <dependencyManagement> or import a BOM (e.g. Spring Boot parent) so child modules can omit <version> safely.

To check what version is actually used: run:

mvn help:effective-pom # shows the POM after inheritance and dependencyManagement applied

mvn dependency:tree # shows resolved dependency versions and transitive dependencies

help:effective-pom is very useful to see which version will be used after inheritance.

Apache Maven

+1

4) What error will Maven throw during mvn install for XML/tag mistakes and how to debug — 2M

If you misspell <artifactId> as <artificatld> (typo):

Two possibilities:

If you wrote an unknown tag but the XML is still well-formed, Maven will usually see the <dependency> is missing a required child (artifactId) and will fail the build with a POM validation error during validate (it will say a dependency element is missing required parts or show a Missing artifact style error when it can’t resolve the dependency).

If the XML is not well-formed (badly closed tags), Maven will throw Non-parseable POM / Failed to parse POM errors and stop immediately. Example errors are “Non-parseable POM” or “Failed to parse POMs” — these mean your pom.xml is malformed XML.

Stack Overflow

Examples Java Code Geeks

How to debug (simple checklist):

Open pom.xml in an IDE (Eclipse/IntelliJ) — the XML editor will highlight typos immediately.

Run quick XML check:

xmllint --noout pom.xml # on Linux/Mac (installs via libxml)

Run Maven in debug to see full trace:

mvn -X clean install

Use mvn help:effective-pom to see the final merged POM (helps find missing / overridden pieces).

Apache Maven

Stack Overflow

Fix: change <artificatld> → <artifactId> and ensure the dependency block has <groupId>, <artifactId>, and (if needed) <version>.

5) You wrote <packing> instead of <packaging> — what happens? — 2M

What Maven assumes by default:

If <packaging> is missing, Maven defaults to jar packaging. So if you accidentally write <packing> (typo), Maven does not see the packaging and will treat the project as jar.

Apache Maven

Impact for a web project:

Web projects must use <packaging>war</packaging>. With default jar, Maven will create a JAR (not a WAR). That means the WAR plugin won’t run and you’ll not get a .war file to deploy to Tomcat -> deployment fails or Tomcat can’t accept the artifact. Fix: correct the tag:

<packaging>war</packaging>

6) Malformed MySQL dependency snippet — what happens & correct snippet — 3M

You showed:

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector/artifactId>

<version>8.0.30</version>

</dependency>

Problems here:

The <artifactId> closing tag is malformed (mysql-connector/artifactId>). This is invalid XML → Maven will likely report a non-parseable POM or XML parsing error and stop.

Stack Overflow

Also the usual correct artifactId for Connector/J is mysql-connector-java (older coords) or mysql-connector-j (newer coords depending on version). Using a wrong artifactId would lead Maven to try to resolve a non-existent artifact and then show a Missing artifact error. To be safe, use the official coordinates.

Correct dependency (one that works for 8.0.30):

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>8.0.30</version>

</dependency>

or for newer coordinates (some versions):

<dependency>

<groupId>com.mysql</groupId>

<artifactId>mysql-connector-j</artifactId>

<version>8.0.33</version>

</dependency>

(Maven Central / repo info shows how to use these coordinates.)

Maven Central

Maven Repository

To test / fix:

# after fixing pom.xml

mvn clean package # or mvn dependency:resolve

mvn dependency:tree # to see that mysql-connector is resolved

Quick checklist (copy & paste) — for your assignment submission

Fix pom.xml typos: <finalName>, <packaging>, <artifactId>, <version> — watch angle brackets carefully.

For JUnit add a version (4.13.2) or rely on parent <dependencyManagement>.

For MySQL use mysql:mysql-connector-java:8.0.30 or com.mysql:mysql-connector-j:8.0.xx.

Maven Central

Maven Repository

Commands to build and inspect:

git clone <https://github.com/Kumbhambhargavi75/FoodSystem.git>

cd FoodSystem

mvn help:effective-pom

mvn dependency:tree

mvn -X clean package

ls -la target/

**But Maven throws an error saying dependency not found. What mistake was made, and what is the correct Maven coordinates for MySQL. connector?-3M**

**7. Your teammate adds the following plugin section:**

**<plugin>**

**<artifactId>tomcat7-maven-plugin</artifactId>**

**<version>2.2</version>**

**</plugin>**

**But when you try to run mva tomcat7:run, it fails.**

**What is missing in the plugin definition, and why won't Maven recognize it? -3M**

**8. in the <url>tag, Written <url>localhost:8080/FoodSystem</url>**

**Will Maven accept this? What is the correct purpose of the <url> element in pom.xml7-2M 9. A developer removes the <dependencies section completely. Will Maven still build the project? What issues might occur during testing?-2M**

**10. When deployed in Tomcat, the context path given as** [**http://localhost:8080/FoodSystem-0.0.1-SNAPSHOT**](http://localhost:8080/FoodSystem-0.0.1-SNAPSHOT)**, what will be the output Why-3M**

**11. You have finalName>Food-system-xyz/finalName> from the <build> section.**

**What will the generated WAR name be, and how will it affect the Tomcat deployment URL?-2M**

**12. <version>0.0.1-SNAPSHOT</version>, What WAR file name will Maven generate, and how will this impact other projects depending on the SNAPSHOT version?-2M**

**13. Add the central dependency of Java Servlet API, 3.1-605 to your existing project check the complete pom.xml file and run it -2M**

**QHL.Git & GitHub Integration with Maven Project (30M)**

**Using the Maven project created above, perform the following Git tasks:**

**1. The cafeteria team is starting the Food Ordering System. You want to initialize a Git repository inside the project folder. What command will you nan -2M**

**1. Before committing, you want to set your Gir asername and email globally so all commits have your identity. Which commands will you use?-2M**

**3. After initializing the repo, you need to connect it to a Gitti lub repository. What helps to do this?-2M 4. You want to commit your staged changes with the message "Added Order Service for Food Ordering**

**System" Write the command-2M 5. You're on main but want to create and switch to a new branch called feature/payment How will you do? -3M**

**You staged the wrong file temp.txt. How do you remove it from staging but keep the file incally?-2M How can you verify the list of remote repositories commected to your project?-2M**

**You're on main but want to temporarily save your uncomoutted changes to test Feature jayment branch Which Git command will you use? Later, you want to restore your stashed changes. What command do you run-2M**

**follow?-2M 9. Your teaminate completed feature/Reviews and you want to merge it into main. What steps will you**

**10. A teaminate wants to work on the Food Ordering System. Which Git command will they use to copy the project from GitHub to their machine?-3M**

**11. You completed the feature real-time-status branch and now want to push it to GitHub for collaboration Which command do you use?-2M**

**12. lastead of HTTPS, you want to connect to GitHub using an SSH key for secure authentication. What are the two steps/commands you need?-3M give every question in a very detailed way from scartch like iam a child**

Q6 — "Maven throws 'dependency not found'. What mistake was made, and what are the correct Maven coordinates for MySQL Connector?" (3M)

What likely went wrong (simple):

The POM had a wrong artifactId or a malformed XML tag (for example you showed mysql-connector/artifactId> and a broken closing tag). That makes Maven either fail to parse the POM or it tries to find an artifact with the wrong name and cannot find it → dependency not found.

Also MySQL’s coordinates changed in recent years, so using old coordinates can lead to “not found” errors.

Correct coordinates (recommended now):

Use the official Connector/J coordinates published by MySQL:

<dependency>

<groupId>com.mysql</groupId>

<artifactId>mysql-connector-j</artifactId>

<version>8.0.30</version> <!-- or a newer 8.0.x you prefer -->

<!-- runtime is usually fine for JDBC drivers -->

<scope>runtime</scope>

</dependency>

(Older projects sometimes used mysql:mysql-connector-java, but the current official coordinates are com.mysql:mysql-connector-j.)

MySQL Developer Zone

Stack Overflow

How to fix step-by-step (child style):

Open pom.xml.

Find the broken dependency for MySQL. Fix any broken tags so it looks exactly like the XML above (angles, closing tags matter!).

Save and run:

mvn clean package

mvn dependency:tree # to check the dependency was downloaded/resolved

If Maven still says "not found", copy the exact groupId/artifactId/version into a browser search of Maven Central to confirm that version exists.

Q7 — Tomcat plugin snippet fails when running mvn tomcat7:run. What is missing and why? (3M)

You showed:

<plugin>

<artifactId>tomcat7-maven-plugin</artifactId>

<version>2.2</version>

</plugin>

What’s missing: the <groupId>. Maven needs the groupId for plugins that are not in the default plugin groups. The Tomcat 7 plugin’s groupId is org.apache.tomcat.maven. Without it Maven cannot find the plugin by artifactId alone (unless you configured pluginGroups in settings.xml).

Correct plugin block:

<plugin>

<groupId>org.apache.tomcat.maven</groupId>

<artifactId>tomcat7-maven-plugin</artifactId>

<version>2.2</version>

</plugin>

Why: Maven resolves plugin prefixes only in standard plugin groups or those added to <pluginGroups> in your settings. Tomcat’s plugin is published under org.apache.tomcat.maven, so you must specify that groupId or Maven won’t recognize tomcat7:run.

Apache Tomcat

Apache Software Foundation

Q8 — In <url> tag you wrote <url>localhost:8080/FoodSystem</url>. Will Maven accept this? What is <url> for? (2M)

Short: Maven will accept a string, but that tag is for the project's homepage or project URL (where users find docs or project site), not for telling Tomcat where to deploy. Also include the scheme (like http://) if you use a real URL.

Example correct use:

<url>http://example.com/my-food-system</url>

Purpose: It documents where the project lives (homepage, project site) for humans and some tools — not a deployment target. Deployment to Tomcat is done by copying the WAR or using the Tomcat plugin (configuration in plugin <configuration>), not by <url> in the POM.

Apache Maven

Stack Overflow

Q9 — A developer removes the <dependencies> section completely. Will Maven still build? What problems in testing? (2M)

Simple answer: Maven might still run basic phases, but your app will very likely fail to compile or tests will fail, because classes from external libraries (JDBC driver, servlet API used in compile, JUnit, etc.) aren’t available on the classpath.

What will happen:

If your code imports classes from libraries that were in <dependencies>, mvn compile will show “cannot find symbol” / missing class errors.

Tests will fail because JUnit or other test libs are missing.

If you somehow compiled earlier and removed dependencies, your package may produce a broken artifact or a JAR/WAR missing required libraries.

Fix: Restore necessary <dependency> blocks (groupId, artifactId, version). Use mvn dependency:tree to see what you need.

Q10 — If deployed in Tomcat the context path is <http://localhost:8080/FoodSystem-0.0.1-SNAPSHOT>, what will be the output and why? (3M)

Short explanation: Tomcat maps the context path to the WAR file (or exploded directory) name. If your WAR file is named FoodSystem-0.0.1-SNAPSHOT.war, Tomcat will, by default, host it at <http://localhost:8080/FoodSystem-0.0.1-SNAPSHOT> (the context path is the file name without .war). That’s why you see the version suffix in the URL. If you want a different URL (like /FoodSystem or /), rename the WAR or use a Tomcat context configuration file.

Apache Tomcat

Octopus Deploy

Q11 — You have finalName>Food-system-xyz/finalName> in <build>. What will the generated WAR be and how will it affect Tomcat URL? (2M)

Two cases:

If you meant <finalName>Food-system-xyz</finalName> (correct XML): Maven will produce target/Food-system-xyz.war (or exploded folder) and Tomcat will map it to <http://host:port/Food-system-xyz> — that becomes the context path.

If the tag is malformed (like missing angle brackets), Maven will fail to parse the POM (XML error) or ignore it — build will fail or fallback to default artifact name <artifactId>-<version>.war. So always ensure the tag is exactly <finalName>...</finalName>.

Apache Maven

Apache Tomcat

Q12 — <version>0.0.1-SNAPSHOT</version>: what WAR name will Maven generate, and how does SNAPSHOT affect other projects? (2M)

WAR name produced: if you have default finalName (artifactId), Maven produces artifactId-0.0.1-SNAPSHOT.war — the version becomes part of the filename.

SNAPSHOT impact (important):

-SNAPSHOT means this is a mutable, in-development version. When a project depends on 0.0.1-SNAPSHOT, Maven may fetch a newer snapshot when building (it’s not a stable release). That causes non-repeatable builds and other projects that depend on this snapshot may get different binaries over time. Use SNAPSHOT only during active development, not for released dependencies.

Apache Maven

Stack Overflow

Q13 — Add central dependency of Java Servlet API 3.1 to project; check complete pom.xml and run it. (2M)

Correct dependency (Servlet API 3.1.0) — use provided scope for web apps (the servlet container — Tomcat — already provides this at runtime):

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>javax.servlet-api</artifactId>

<version>3.1.0</version>

<scope>provided</scope>

</dependency>

This is the official artifact on Maven Central. Use provided so Maven compiles your code against the servlet API but does not package it into the WAR (Tomcat provides it).

Maven Repository

repo.maven.apache.org

Example complete minimal pom.xml for your FoodSystem webapp (copy-paste, then edit groupId/artifactId/version as you like)

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

<http://maven.apache.org/xsd/maven-4.0.0.xsd>">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>FoodSystem</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>war</packaging>

<name>Food Ordering System</name>

<url>http://your-college.example.com/foodsystem</url>

<properties>

<maven.compiler.source>1.8</maven.compiler.source>

<maven.compiler.target>1.8</maven.compiler.target>

</properties>

<dependencies>

<!-- Servlet API for compile (provided by Tomcat at runtime) -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>javax.servlet-api</artifactId>

<version>3.1.0</version>

<scope>provided</scope>

</dependency>

<!-- MySQL Connector (JDBC driver) -->

<dependency>

<groupId>com.mysql</groupId>

<artifactId>mysql-connector-j</artifactId>

<version>8.0.30</version>

<scope>runtime</scope>

</dependency>

<!-- JUnit for tests -->

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<finalName>FoodSystem</finalName>

<plugins>

<plugin>

<groupId>org.apache.tomcat.maven</groupId>

<artifactId>tomcat7-maven-plugin</artifactId>

<version>2.2</version>

<configuration>

<port>8080</port>

<path>/FoodSystem</path>

</configuration>

</plugin>

</plugins>

</build>

</project>

How to run/build locally (commands):

# check POM

mvn help:effective-pom

# build package (creates target/FoodSystem.war)

mvn clean package

# run with tomcat plugin (will use plugin config)

mvn tomcat7:run

# or if plugin needs a fully configured manager URL, use its configuration with username/password

If you get plugin not found, ensure plugin <groupId> is present (see Q7), and check network/proxy.

— Git & GitHub Integration with Maven Project (30M) — very detailed, step-by-step like you’re a child

Assume you are inside your project folder (where pom.xml is).

1. Initialize a Git repository inside the project

Command:

git init

What this does (simple): creates a .git folder and starts tracking version history in this folder.

2. Set your Git username and email globally (so commits show your identity)

Commands:

git config --global user.name "Your Name"

git config --global user.email "you@example.com"

Why: Git uses these values to label every commit you make.

3. After initializing, connect to a GitHub repository (remote)

Option A — add a remote to an already created GitHub repo:

git remote add origin <https://github.com/your-username/YourRepo.git>

Option B — clone from Github (they get a copy to their machine):

git clone <https://github.com/your-username/YourRepo.git>

(If you use SSH later, use the git@github.com:... form.)

4. Commit staged changes with message "Added Order Service for Food Ordering System"

Steps:

git add . # stage all changed files

git status # see staged files

git commit -m "Added Order Service for Food Ordering System"

What it does: git add stages files; git commit records the snapshot with your message.

5. Create and switch to a new branch feature/payment

Command (new Git way):

git switch -c feature/payment

Or older alternative:

git checkout -b feature/payment

Simple: this creates the branch and moves you onto it so you can work separately.

6. You staged the wrong file temp.txt. Remove it from staging but keep the file locally

Command:

git reset temp.txt

What happens: temp.txt remains in the folder but it’s no longer staged for commit.

7. Verify the list of remote repositories connected to your project

Command:

git remote -v

It shows: the origin (and others) fetch/push URLs.

8. You want to temporarily save uncommitted changes to test another branch (stash), then restore later

Save/stash changes:

git stash save "WIP: testing payment flow"

# or simply

git stash

Switch branches and test.

Restore the latest stash:

git stash pop # applies stash and removes it from list

# or

git stash apply # applies stash but keeps it in stash list

9. Your teammate completed feature/Reviews and you want to merge it into main. Steps:

Switch to main:

git switch main

Make sure main is up to date:

git pull origin main

Merge the feature branch:

git merge feature/Reviews

If there are conflicts, Git will tell you; open the conflicted files, fix them, then:

git add <fixed-files>

git commit # completes the merge commit if necessary

Push the merged main to remote:

git push origin main

10. A teammate wants to copy the project from GitHub to their machine (clone)

Command they should run:

git clone <https://github.com/your-username/YourRepo.git>

(Or, using SSH: git clone git@github.com:your-username/YourRepo.git)

11. You completed feature/real-time-status and want to push it to GitHub

Commands:

git switch feature/real-time-status

git add .

git commit -m "Implement real-time order status"

git push -u origin feature/real-time-status

-u sets the upstream so in future you can just do git push.

12. Instead of HTTPS you want to use SSH for GitHub — two main steps (short & clear)

Step A — create an SSH key locally:

ssh-keygen -t ed25519 -C "you@example.com"

# (Press Enter to accept defaults; give a passphrase if you want)

If your system doesn’t support ed25519, use:

ssh-keygen -t rsa -b 4096 -C "you@example.com"

Step B — add the public key to your GitHub account:

Copy the public key:

cat ~/.ssh/id\_ed25519.pub

Go to GitHub → Settings → SSH and GPG keys → New SSH key → paste and save.

Now change remote to SSH or clone via SSH:

git remote set-url origin git@github.com:your-username/YourRepo.git

# or clone using:

git clone git@github.com:your-username/YourRepo.git

That’s it — you’ll now authenticate via your SSH key instead of typing username/password.

Quick checklist you can copy-paste

Fix the MySQL dependency to com.mysql:mysql-connector-j:8.0.30.

MySQL Developer Zone

Ensure Tomcat plugin includes <groupId>org.apache.tomcat.maven</groupId>.

Apache Tomcat

Use <packaging>war</packaging> for webapps (default is jar if missing).

Stack Overflow

Add servlet API dependency javax.servlet:javax.servlet-api:3.1.0 with provided scope.

Maven Repository

Build & test: mvn clean package, mvn tomcat7:run (after plugin fix).

**Your wammate emailed you a patch file with a CSS fix. You need to apply it. test it locally, and mahade it m your Oft history**

**Which Ost command is med to apply the patch and how do you ensure it becomes part of your commins 3M**

**QIV. Docker containerization for Maven Application- 20M (10-2-28M)**

**Task: Containerize your Maven project using Docker**

**a. Refer the given link/documentation to write a Dockerfile for the Maven project**

**b.** [**https://github.com/Kumbhambhargavi75/FoodSystem**](https://github.com/Kumbhambhargavi75/FoodSystem)

**c. Ensure it copies the WAR/JAR and runs on Tomcat for relevant base image)**

**Answer the following:**

**1. You want to pull the official nginx:latestimage from Docker Huh. Which command do you use?**

**2. You want to run the pulled nginx image as a container named myngias, exposing port 8090 on the host What command will you use?**

**3. How do you check which host port is mapped to container port 8080 4. You notice one of your containers is consuming high CPU and want to stop it**

**How do you stop a running container named my app?**

**5. You want to run your web app container and expose its internal port 5000 on host port 8080 Which command should you use?**

**6. pull the officialpython image from Docker Hub, run it in a container and list of containers that are**

**running, 7. verify whether web-app it is accessible on port:8080 if getting conflict how do you solve**

**8. If the application does not nan properly on port 8080, update the configuration and rebuild the Docker**

**image and run on 9090 port. 9. How can you stop a running container and start it**

**10. How will you Check container status**

**V. DOCKER -COMPOSE-10M**

**Your have successfully buil the Food Ordering System using Maven and pushed the application image to Docker Hub. The next step is to run the application along with a database so that it can manage orders, user and transactions**

**1. Write a docker-compose yml file for yourFood Ordering System PostgreSQL -5m**

**2. Pulls your Maven-built Food Ordering System image from Docker Hub and runs it on port 7078-3M**

**3. Runs a PostgreSQLeontainer as the database and insert the data-2M**

QIII: Git – Applying Patch

👉 Scenario: Your teammate sends you a .patch file (say fix-style.patch) with a CSS fix. You want to apply it, test it locally, and make it part of your Git history.

1. Apply the patch file

git apply fix-style.patch

git apply takes the changes inside the patch file and applies them to your working directory.

After this, your code will be updated, but not yet committed.

2. Test it locally

Run your application (for example in Eclipse, or with mvn clean install and deploy). Check if the CSS fix works.

3. Stage the changes

git add .

(This adds the patched files to staging.)

4. Commit it

git commit -m "Applied CSS fix from teammate patch"

Now the patch changes are part of your Git history.

👉 If you want to apply and commit patch in one go:

git am fix-style.patch

git am = apply patch + create a commit automatically with the author’s details.

QIV: Docker Containerization for Maven Application

We want to run the Food Ordering System inside a Docker container.

a. Write a Dockerfile

Example Dockerfile for a Maven WebApp (WAR file):

# Step 1: Use Maven to build the project

FROM maven:3.8.6-jdk-11 AS build

WORKDIR /app

COPY . .

RUN mvn clean package -DskipTests

# Step 2: Use Tomcat as runtime

FROM tomcat:9.0

COPY --from=build /app/target/FoodSystem-0.0.1-SNAPSHOT.war /usr/local/tomcat/webapps/FoodSystem.war

EXPOSE 8080

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

Questions

1. Pull nginx image

docker pull nginx:latest

2. Run nginx container on port 8090

docker run -d --name mynginx -p 8090:80 nginx:latest

3. Check which host port maps to container port 8080

docker ps

Look under PORTS column.

4. Stop high CPU container (myapp)

docker stop myapp

5. Run web app container, internal 5000 → host 8080

docker run -d -p 8080:5000 my-web-app

6. Pull Python image, run container, list containers

docker pull python:latest

docker run -it --name mypython python:latest

docker ps

7. Verify web app on port 8080, resolve conflict

If another service already uses port 8080 → use a different port:

docker run -d -p 9090:5000 my-web-app

8. If app fails on 8080 → update Dockerfile, rebuild & run on 9090

docker build -t my-web-app .

docker run -d -p 9090:8080 my-web-app

9. Stop & restart a container

docker stop myweb

docker start myweb

10. Check container status

docker ps -a

QV: Docker-Compose

We want Food Ordering System app + PostgreSQL DB running together.

1. docker-compose.yml

version: '3.8'

services:

foodsystem:

image: your-dockerhub-username/foodsystem:latest

ports:

- "7078:8080"

depends\_on:

- db

db:

image: postgres:14

environment:

POSTGRES\_USER: foodadmin

POSTGRES\_PASSWORD: foodpass

POSTGRES\_DB: fooddb

ports:

- "5432:5432"

2. Pull and run app on port 7078

When you run:

docker-compose up -d

It will pull FoodSystem image from Docker Hub

Expose it at <http://localhost:7078/FoodSystem>

3. Run PostgreSQL and insert data

Database is automatically started inside the container.

Connect:

docker exec -it <db-container-id> psql -U foodadmin -d fooddb

Then insert sample data:

INSERT INTO orders (id, item, price) VALUES (1, 'Pizza', 200);

✅ Done! You now have:

Patch applied correctly in Git.

Dockerized Maven WebApp.

Docker-Compose running App + PostgreSQL.

Do you want me to also write the exact commands to push your built Docker image to Docker Hub (so that your compose file can use it)?

You said:

give every question in a very detailed way from scartch like iam a child

ChatGPT said:

Thought for 21s

Great — I’ll explain every step slowly and simply, as if you’re seeing this for the first time. I’ll show the exact commands, explain what each one does, how to test, and what to do if something goes wrong.

I’ll cover (A) applying a Git patch and making it part of history, (B) Docker basics and the 10 Docker questions, and (C) Docker Compose for the Food Ordering System (including inserting initial data).

A. Git — apply a patch, test, and make it part of your history (3 marks — explained like a child)

Scenario: Your teammate emailed fix-style.patch which changes some CSS. You must apply it, test it locally, and make it a commit in your repo.

1) Two ways to apply a patch — same idea, different effects

git apply fix-style.patch

What it does: applies the changes inside the patch file to your working folder (files on disk change), but no commit is created and the patch author metadata is not preserved.

Use when: you just want the changes in your working copy so you can test and then make your own commit.

git am fix-style.patch

What it does: applies the patch and creates a commit using the author/date/message from the patch (if patch was created with git format-patch). This makes the patch part of history automatically.

Use when: the patch is from a colleague and you want to preserve their author info and message.

2) Step-by-step — apply, test, commit, push (child-style)

Assume you’re already inside the project folder (where .git and pom.xml live).

A. Save the patch file into the project folder

(If patch is in email, download it to the project folder or copy it there.)

B. Apply the patch to working directory (no commit yet)

git apply fix-style.patch

If this runs with no output and exit code 0, the files were updated.

If you see errors like error: patch failed: then the patch did not apply cleanly (maybe files changed since patch created).

If patch fails to apply:

Try git apply --reject fix-style.patch — this will write .rej files for hunks that failed; open those .rej files and manually fix the files.

If the patch was made by git format-patch (mailbox), use git am fix-style.patch instead — it can handle mailbox format.

C. Check what changed (very important)

git status # shows modified files

git diff # shows the actual code changes

Look at the changed file(s) to ensure the CSS fix looks right.

D. Test locally (how to test CSS fix)

If the app runs in Tomcat: build & run locally and open the page.

mvn clean package

# Deploy target/FoodSystem.war to Tomcat (or use mvn tomcat7:run if configured)

Alternatively, if frontend files served statically: open the page in browser (<http://localhost:8080/>

...) and check the CSS change visually.

Use browser DevTools (F12) to inspect the changed CSS rule and confirm it's loaded.

E. Stage and commit the change (if you used git apply)

git add path/to/changed-file.css

git commit -m "Apply CSS fix from teammate (fix-style.patch)"

If you want the patch author's metadata preserved (author name and email) but used git apply, you can set the author on commit:

git commit --author="Their Name <their.email@example.com>" -m "Apply CSS fix from teammate"

F. If you used git am it already created a commit

Check commit with:

git log -1

It will show the latest commit (author and message from the patch).

G. Push the commit to remote (so it's in project history on GitHub)

git push origin your-branch-name

3) Verify it’s now part of history

git log --oneline

# or for more detail:

git log -1 # shows last commit author, date, message

If you see the commit, the patch is in Git history.

B. Docker — containerization basics & answers to the 10 Docker questions (child-level detail)

I’ll first give a quick Dockerfile example for your Maven/Tomcat webapp, then answer each numbered question with commands and simple explanations.

Dockerfile (multi-stage build — build with Maven, run with Tomcat)

Create a file named Dockerfile in your project root:

# 1) Build stage: use Maven to build the WAR

FROM maven:3.8.6-jdk-11 AS build

WORKDIR /app

COPY . .

# build the project and skip tests for faster iteration; remove -DskipTests to run tests

RUN mvn clean package -DskipTests

# 2) Run stage: use official Tomcat to run the WAR

FROM tomcat:9.0

# Replace the path below with your actual WAR name in target/

COPY --from=build /app/target/FoodSystem-0.0.1-SNAPSHOT.war /usr/local/tomcat/webapps/FoodSystem.war

EXPOSE 8080

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

What happens

Stage 1 (build): Maven runs and produces target/\*.war.

Stage 2 (run): Tomcat image runs and we copy the WAR into Tomcat’s webapps folder.

EXPOSE 8080 documents the port; docker run -p maps it to host.

If your WAR name is different, update the COPY --from=build ... source path.

Now the 10 Docker questions — exact commands + simple explanations

1) Pull the official nginx:latest image from Docker Hub

docker pull nginx:latest

What it does: downloads the nginx image to your machine so you can run containers from it.

2) Run the pulled nginx image as container named mynginx, exposing host port 8090

docker run -d --name mynginx -p 8090:80 nginx:latest

-d runs in background (detached).

--name mynginx gives the container a friendly name.

-p 8090:80 maps host port 8090 to the container’s port 80 (nginx default).

After this, open <http://localhost:8090> in browser — you should see nginx welcome page.

3) How to check which host port is mapped to container port 8080

Method A — list running containers

docker ps

Look at the PORTS column, e.g. 0.0.0.0:8090->80/tcp shows host:container mapping.

Method B — show mapping for a specific container

docker port mycontainer 8080

# example:

docker port mywebapp 8080

This prints something like 0.0.0.0:9090 meaning container port 8080 is reachable at host 9090.

4) Stop a running container named myapp

docker stop myapp

What it does: sends a graceful stop signal; container shuts down.

If it does not stop in time, force kill:

docker kill myapp

5) Run your web app container and expose internal port 5000 on host port 8080

docker run -d --name webapp -p 8080:5000 your-image-name:tag

This maps host 8080 → container 5000.

Replace your-image-name:tag with the image built from your Dockerfile.

6) Pull official python image, run it in a container, and list running containers

docker pull python:latest

# run interactive container (example: sleep forever so it stays running)

docker run -d --name mypython python:latest sleep infinity

# list running containers

docker ps

sleep infinity keeps the container alive so you can docker exec into it; replace with -it python:latest /bin/bash if you want interactive shell.

7) Verify whether web app is accessible on port 8080 and solve conflicts

Check accessibility

From terminal:

curl -I <http://localhost:8080/>

-I shows HTTP headers; a 200 OK means reachable.

From browser: open <http://localhost:8080>.

If you get a port conflict (port already used):

Find the process using port 8080:

Linux/macOS:

sudo lsof -i :8080

# or

sudo ss -tulpn | grep 8080

Windows (PowerShell):

netstat -ano | findstr :8080

Stop the conflicting service (for example Tomcat or another Docker container):

# If it's a Docker container named otherapp:

docker stop otherapp

# If it's a system service (Linux example)

sudo systemctl stop tomcat

Or use a different host port when running your container:

docker run -d -p 9090:5000 your-image-name

# now open <http://localhost:9090>

8) If the application does not run properly on port 8080, update config, rebuild, and run on 9090

Common steps

Edit app config or Dockerfile as needed (for example, your app may listen on a different internal port; ensure the Dockerfile/cmd matches).

Rebuild image:

docker build -t my-web-app:latest .

Run on host port 9090:

docker run -d --name myweb -p 9090:8080 my-web-app:latest

Check logs for errors:

docker logs -f myweb

Use -f (follow) to stream logs.

9) How to stop a running container and then start it

docker stop myweb # stop, graceful

docker start myweb # start an existing stopped container

To restart immediately:

docker restart myweb

Note: docker run creates a new container; docker start restarts an existing stopped container.

10) How to check container status

List running containers:

docker ps

List all containers (running and stopped):

docker ps -a

See resource usage (CPU / memory) for running containers:

docker stats

Get detailed info about a container:

docker inspect myweb

C. Docker Compose — run FoodOrderingSystem + PostgreSQL (10 marks) — step-by-step like a child

We will create docker-compose.yml to run two containers: foodsystem (your app) and db (Postgres). I’ll show the compose file, the init SQL to pre-load data, how to run it, and how to verify the DB data.

1) docker-compose.yml (full example)

Create a file docker-compose.yml in project root:

version: "3.8"

services:

foodsystem:

image: your-dockerhub-username/foodsystem:latest

# If you haven't pushed to Docker Hub yet you can build locally with 'build: .'

# build: .

ports:

- "7078:8080" # host:container -> app will be reachable at <http://localhost:7078>

depends\_on:

- db

environment:

SPRING\_DATASOURCE\_URL: jdbc:postgresql://db:5432/fooddb

SPRING\_DATASOURCE\_USERNAME: foodadmin

SPRING\_DATASOURCE\_PASSWORD: foodpass

restart: unless-stopped

db:

image: postgres:14

environment:

POSTGRES\_USER: foodadmin

POSTGRES\_PASSWORD: foodpass

POSTGRES\_DB: fooddb

volumes:

- db-data:/var/lib/postgresql/data

# Mount SQL files into /docker-entrypoint-initdb.d to auto-run on first container start

- ./db-init:/docker-entrypoint-initdb.d

ports:

- "5432:5432"

restart: unless-stopped

volumes:

db-data:

Explain like a child

foodsystem is your web app. It uses the image from Docker Hub your-dockerhub-username/foodsystem:latest. If you don’t want to pull from Docker Hub you can uncomment build: . to build the image from the local Dockerfile.

ports: "7078:8080" maps host port 7078 to container 8080 — open <http://localhost:7078> to see your app.

db runs PostgreSQL and uses ./db-init to run SQL files automatically only the first time the DB container creates the database.

2) How to add initial SQL to insert data (so the DB has example rows)

Create a folder db-init next to docker-compose.yml. Inside it create a file 01-init-data.sql with sample SQL:

-- db-init/01-init-data.sql

CREATE TABLE IF NOT EXISTS orders (

id SERIAL PRIMARY KEY,

item VARCHAR(100),

price INTEGER,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

INSERT INTO orders (item, price) VALUES

('Veg Thali', 120),

('Tea', 15),

('Pizza Slice', 80);

Important: Files in ./db-init are executed once when the Postgres data directory is initialized. If you already ran the db service and it has persisted data, the init scripts will not run again unless you remove the volume.

3) Steps to run the whole stack

A. If your app image is on Docker Hub

# in directory with docker-compose.yml

docker-compose pull

docker-compose up -d

Docker Compose will pull your-dockerhub-username/foodsystem:latest and postgres:14 then start both containers.

B. If you want to build locally instead of pulling

Edit docker-compose.yml and replace the image: line for foodsystem with:

foodsystem:

build: .

image: foodsystem-local:latest

Then run:

docker-compose up -d --build

4) Verify the app and DB

Check containers are running

docker-compose ps

Check your app

Open in browser: <http://localhost:7078>

Or test with curl:

curl -I <http://localhost:7078>

Connect to PostgreSQL and check inserted data

# find db container name

docker-compose ps

# assume it is named projectname\_db\_1. To connect:

docker exec -it <db-container-name> psql -U foodadmin -d fooddb

# inside psql prompt:

SELECT \* FROM orders;

You should see the rows inserted from 01-init-data.sql.

If init script didn't run: likely because the DB volume already exists (Postgres only runs init scripts on a fresh data directory). To force re-run, stop containers and remove the DB volume:

docker-compose down

docker volume rm <project>\_db-data # careful: this deletes DB data

docker-compose up -d

Extra helpful bits (child-friendly reminders)

How to push your app image to Docker Hub (so docker-compose pulls it):

Build image locally:

docker build -t your-dockerhub-username/foodsystem:latest .

Login:

docker login

(Enter Docker Hub username & password)

Push:

docker push your-dockerhub-username/foodsystem:latest

If your app needs environment variables to connect to DB, set them in docker-compose.yml (I showed SPRING\_DATASOURCE\_\* as an example for Spring apps).

To stop everything:

docker-compose down

Add -v to also remove volumes (data): docker-compose down -v

Quick checklists you can copy-paste

Apply patch and commit (fast)

# apply

git apply fix-style.patch

# inspect changes

git status

git diff

# commit

git add .

git commit -m "Apply CSS fix from patch"

git push origin your-branch

Build Docker image and run locally (fast)

docker build -t foodsystem-local:latest .

docker run -d --name foodsystem -p 8080:8080 foodsystem-local:latest

docker logs -f foodsystem

# visit <http://localhost:8080>

Start app + postgres with docker-compose

# in folder with docker-compose.yml and db-init

docker-compose up -d --build

docker-compose ps

# visit <http://localhost:7078>