Smart Assessment Hub (SAH) is a technology-driven platform that streamlines examination and evaluation processes. It enhances learning outcomes through automated grading, real-time feedback, and data-driven insights. The platform ensures transparency, scalability, and efficiency in assessments. By integrating advanced analytics, it empowers educators and institutions to make informed decisions for continuous improvement. Based on this scenario, answer the following:[10 Marks] a. Abstract - 4M b. Functional Requirements-2M c. Non-Functional Requirements-2M d. Identification of Users-2  
  
**a. Abstract (4M)**

The *Smart Assessment Hub (SAH)* is a digital examination and evaluation platform designed to automate grading and provide real-time feedback. It integrates advanced analytics to generate meaningful insights, helping educators improve teaching methods and students to enhance learning outcomes. The system ensures **transparency, scalability, and efficiency** in conducting assessments, while supporting data-driven decision-making for continuous academic improvement.

**b. Functional Requirements (2M)**

These are the features the system **must perform**:

1. **Automated Grading** – Evaluate objective-type questions instantly.
2. **Real-Time Feedback** – Provide students with performance results immediately after submission.
3. **Report Generation** – Generate detailed analytics reports for students and educators.
4. **User Management** – Manage student, teacher, and admin roles with proper authentication.

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

These define the **quality attributes** of the system:

1. **Scalability** – Must handle thousands of simultaneous users during peak exam times.
2. **Reliability & Accuracy** – Ensure accurate grading and consistent availability.
3. **Security** – Protect sensitive exam and student data through authentication and encryption.
4. **Performance** – Response time should be minimal, even under heavy load.

**d. Identification of Users (2M)**

The primary users of the SAH system are:

1. **Students** – Take assessments, receive feedback, and view reports.
2. **Educators/Faculty** – Create question papers, monitor exams, and analyze performance reports.
3. **Administrators** – Manage system operations, user accounts, and maintain overall platform integrity.

**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 for the above project. The template is provided as per the following link.

You are given the following GitHub repository link:

<https://github.com/savram674/Smart-Assessment-Hub.git>

1. Download the given GitHub repository. There are few errors in POM.XML correct them and build the project. 10M
2. The project requires **Servlet API** dependency. Add the proper Maven dependency for javax.servlet-api:4.0.1. – 5M
3. What happens if the <version> tag is omitted in a dependency? – 2M
4. Add a dependency for **MySQL Connector/J 8.0.33** to enable database connectivity. – 5M
5. Package the project as a WAR file and explain how to deploy it on Tomcat. – 3M
6. Where do you find the added dependency of Servlet in the maven local repository – 2M
7. Draw the folder structure of your project. Where you find packaged file? – 3M

**1. Download the GitHub repository & fix POM.XML errors (10M)**

Steps:

* Clone the repo:

git clone https://github.com/savram674/Smart-Assessment-Hub.git

cd Smart-Assessment-Hub

* Open pom.xml. Common errors usually include:
  + Wrong plugin/dependency names (like <dependecy> instead of <dependency>).
  + Missing <groupId>, <artifactId>, or <version> tags.
  + Incorrect <packaging> (should be war for web apps).

✅ Example corrected structure:

<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.kmit</groupId>

<artifactId>smart-assessment-hub</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>war</packaging>

<dependencies>

<!-- Servlet & MySQL will be added in next steps -->

</dependencies>

</project>

Build the project:

mvn clean install

**2. Add Servlet API Dependency (javax.servlet-api:4.0.1) (5M)**

Inside <dependencies>:

<dependency>

<groupId>javax.servlet</groupId>

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

<version>4.0.1</version>

<scope>provided</scope>

</dependency>

👉 scope=provided means Tomcat/Server provides it at runtime.

**3. What happens if <version> tag is omitted? (2M)**

* Maven tries to use the **latest version** available in repositories.
* This can lead to **build failures** if the latest version has breaking changes.
* Best practice → always specify <version>.

**4. Add MySQL Connector/J 8.0.33 Dependency (5M)**

<dependency>

<groupId>mysql</groupId>

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

<version>8.0.33</version>

</dependency>

This allows database connectivity with MySQL.

**5. Package project as WAR & deploy on Tomcat (3M)**

* In pom.xml, ensure:

<packaging>war</packaging>

* Build WAR file:

mvn clean package

* Output file:  
  target/smart-assessment-hub-1.0-SNAPSHOT.war

**Deployment:**

1. Copy WAR → $CATALINA\_HOME/webapps/
2. Start Tomcat → bin/startup.sh (Linux) or startup.bat (Windows).
3. Access → http://localhost:8080/smart-assessment-hub

**6. Where is the Servlet dependency stored? (2M)**

* Local Maven Repository → ~/.m2/repository (Linux/Mac) OR C:\Users\<username>\.m2\repository (Windows).
* Exact path:

~/.m2/repository/javax/servlet/javax.servlet-api/4.0.1/

**7. Folder Structure & Packaged File Location (3M)**

📂 **Maven Web App Structure**

Smart-Assessment-Hub/

│── src/

│ ├── main/

│ │ ├── java/ # Java source files

│ │ ├── resources/ # Config files

│ │ └── webapp/ # JSPs, HTML, CSS, WEB-INF

│ │ └── WEB-INF/

│ │ └── web.xml

│ └── test/ # Test cases

│

│── pom.xml

│── target/

│ └── smart-assessment-hub-1.0-SNAPSHOT.war ✅

👉 Packaged WAR file is found in target/.

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

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

1. Introduce yourself to git - Configure -- **2 M**
2. Initialize the Smart Assessment Hub project in Git. – 2M
3. Commit with message “Added automated grading module”. – 2M
4. Check the status of modified and staged files. – 2M
5. View the commit history with author details. – 2M
6. Create a new branch feedback-module. – 2M
7. Merge feedback-module into main without conflicts [resolve the intentionally created conflict]. – 3M
8. Resolve a merge conflict when report-module branch conflicts with main. – 3M
9. Create a patch file for bug fixing. Write step to Apply a .patch file. – 3M
10. Push the branch analytics-dashboard to GitHub. – 2M
11. Use a fork-and-pull request workflow to contribute a new feature for the git repo https://github.com/octocat/Spoon-Knife . – 5M
12. Check the differences between your local branch and remote main branch. – 2M

**1. Introduce yourself to git (Configure) — 2M**

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

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

👉 This sets your identity for commits.

**2. Initialize the Smart Assessment Hub project in Git — 2M**

cd Smart-Assessment-Hub

git init

👉 Creates a new .git repo in your project folder.

**3. Commit with message — 2M**

git add .

git commit -m "Added automated grading module"

**4. Check status of modified and staged files — 2M**

git status

👉 Shows untracked, staged, and modified files.

**5. View commit history with author details — 2M**

git log --oneline --author="Your Name"

👉 Use git log for full details (hash, author, date, message).

**6. Create a new branch feedback-module — 2M**

git checkout -b feedback-module

👉 Creates & switches to feedback-module.

**7. Merge feedback-module into main without conflicts — 3M**

git checkout main

git merge feedback-module

* If conflict happens → open files, resolve manually, then:

git add .

git commit -m "Resolved merge conflict in feedback-module"

**8. Resolve a merge conflict with report-module — 3M**

* Switch & merge:

git checkout main

git merge report-module

* If conflicts:
  + Edit files → keep correct changes.
  + Mark resolution:

git add <conflicted-files>

git commit -m "Resolved merge conflict with report-module"

**9. Create & apply a patch file — 3M**

Create patch:

git diff > bugfix.patch

Apply patch:

git apply bugfix.patch

**10. Push analytics-dashboard branch to GitHub — 2M**

git checkout -b analytics-dashboard

git push origin analytics-dashboard

**11. Fork-and-Pull Request workflow (Spoon-Knife repo) — 5M**

1. Fork https://github.com/octocat/Spoon-Knife to your GitHub.
2. Clone fork:
3. git clone https://github.com/<your-username>/Spoon-Knife.git
4. Add feature, commit, push to your fork.
5. Go to GitHub → click **New Pull Request** → submit to main repo.

**12. Check differences between local & remote main — 2M**

git fetch origin

git diff main origin/main

👉 Shows differences between your local main and remote main.

✅ That’s a **complete answer for QIII (30 Marks)**.

Do you also want me to make a **diagram/flowchart of the Git branching workflow** (main → feedback-module → report-module → analytics-dashboard → fork → pull request)? It could score you brownie points in the exam.

**You said:**

QIV. Docker containerization for Maven Web Application - 20M Task: Containerize your Maven project using Docker. 1. Check Docker version and list all images. – 2M 2. Pull and run the hello-world image. – 2M 3. Run python image in an interactive mode. – 2M 4. Write a Dockerfile to build and run the Smart Assessment Hub application using Tomcat base image. 3M 5. Build an image named smarthub-image and run it as a container. – 2M 6. List all containers (running and stopped). – 2M 7. Start and stop a running container. – 2M 8. Push your custom image to Docker Hub. – 3M 9. Inspect the logs of a Docker container to troubleshoot errors. – 2M

**ChatGPT said:**

Nice ⚡ Let’s break down **QIV – Docker containerization for Maven Web App (20M)** step by step.

**1. Check Docker version & list images (2M)**

docker --version

docker images

👉 Confirms installation & shows downloaded images.

**2. Pull and run hello-world image (2M)**

docker pull hello-world

docker run hello-world

👉 Simple test that Docker is working.

**3. Run python image in interactive mode (2M)**

docker pull python

docker run -it python

👉 Drops you into a Python REPL inside the container.

**4. Dockerfile for Smart Assessment Hub (3M)**

Your Maven project (WAR) should be deployed on Tomcat.

**Dockerfile**:

# Use official Tomcat base image

FROM tomcat:9.0

# Maintainer info (optional)

LABEL maintainer="yourname@example.com"

# Copy WAR file to Tomcat webapps directory

COPY target/smart-assessment-hub-1.0-SNAPSHOT.war /usr/local/tomcat/webapps/smart-assessment-hub.war

# Expose Tomcat default port

EXPOSE 8080

# Start Tomcat

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

**5. Build image & run container (2M)**

docker build -t smarthub-image .

docker run -d -p 8080:8080 --name smarthub-container smarthub-image

👉 -d = detached, -p = port mapping.  
👉 Access app → http://localhost:8080/smart-assessment-hub.

**6. List all containers (2M)**

docker ps -a

👉 Shows running & stopped containers.

**7. Start & stop container (2M)**

docker stop smarthub-container

docker start smarthub-container

**8. Push custom image to Docker Hub (3M)**

1. Login to Docker Hub:

docker login

1. Tag image:

docker tag smarthub-image yourdockerhubusername/smarthub-image:v1

1. Push image:

docker push yourdockerhubusername/smarthub-image:v1

**9. Inspect logs of a container (2M)**

docker logs smarthub-container

👉 Useful for debugging Tomcat startup or app errors.

✅ That’s a **full solution for QIV (20M)**.

Do you also want me to prepare a **ready-to-run docker-compose.yml** (Tomcat + MySQL) for QV? That way you’ll have both QIV and QV covered smoothly.

**You said:**

QV. DOCKER COMPOSE -10M Your have successfully built Smart Assessment Hub using Maven and pushed the application image to Docker Hub. The next step is to run the application along with a Tomcat and run the containers. Also include a database for future use (MYSQL) 1. create a multi folder docker-compose.yml for your Smart Assessment Hub Project with Tomcat Server and include a database mysql to be used at port 3306. Add volumes for persistency. Show both the containers running using docker compose up command -- 5M 2. Container 1 – Access Tomcat at port 8080 -- 2M 3. Container 2 – Execute your database to add a new table -- 3M

**ChatGPT said:**

Perfect ✅ Let’s wrap up with **QV – Docker Compose (10M)**. This will combine your **Smart Assessment Hub app (Tomcat WAR)** and a **MySQL database** into a multi-container setup.

**1. Create a multi-folder docker-compose.yml (5M)**

📂 Project structure:

Smart-Assessment-Hub/

│── docker-compose.yml

│── app/ # For Tomcat + SmartHub image

│── db/ # For MySQL persistence

📌 **docker-compose.yml**

version: '3.8'

services:

sah-app:

image: yourdockerhubusername/smarthub-image:v1 # custom image from QIV

container\_name: sah-app

ports:

- "8080:8080"

depends\_on:

- sah-db

networks:

- sah-network

sah-db:

image: mysql:8.0

container\_name: sah-db

restart: always

environment:

MYSQL\_ROOT\_PASSWORD: root123

MYSQL\_DATABASE: sahdb

MYSQL\_USER: sahuser

MYSQL\_PASSWORD: sahpass

ports:

- "3306:3306"

volumes:

- db\_data:/var/lib/mysql

networks:

- sah-network

volumes:

db\_data:

networks:

sah-network:

👉 This defines **two containers**:

* sah-app → Tomcat running Smart Assessment Hub (port 8080).
* sah-db → MySQL database with persistence (volume).

Run them:

docker-compose up -d

Check containers:

docker ps

**2. Container 1 – Access Tomcat at port 8080 (2M)**

Open browser:

http://localhost:8080/smart-assessment-hub

**3. Container 2 – Execute database & add table (3M)**

Connect to MySQL container:

docker exec -it sah-db mysql -u root -p

(password = root123)

Inside MySQL:

USE sahdb;

CREATE TABLE students (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100),

marks INT

);

✅ Now you have **Tomcat + MySQL running together** with persistence.