**KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY**



**(AN AUTONOMOUS INSTITUTE)**

 **Accredited by NBA & NAAC, Approved by AICTE, Affiliated to JNTUH, Hyderabad**

**A.Y 2025-2026**

**Department of Computer Science & Engineering (DS)**

**Lab Internal I**

**Subject Name: Software Engineering Subject Code: 23CC501PC**

**Year and Semester: III / I Branch /Section: CSD-B Faculty:** Y Deepthi  **Lab Internal: 10/09/2025**

**Set-3: Hospital Appointment Management System (HAMS)**

**Part I – Software Requirement Specification (SRS) [10 Marks]**

A city hospital is facing challenges in handling patient appointments, doctor availability, billing, and reports. Patients often face long queues because booking is manual, while doctors find it difficult to track schedules and patient records. To solve these issues, the hospital wants to develop a digital **Hospital Appointment Management System (HAMS)**.

**Questions:**

1. Write a short abstract describing the purpose of HAMS. [2M]
2. List at least 5 functional requirements (features). [2M]
3. List at least 5 non-functional requirements. [2M]
4. Identify different types of users and how they will interact with the system. [2M]
5. Break the system into modules. [2M]

**Part II – Maven Web Application Development [25 Marks]**

You are developing the HAMS as a Maven-based Java Web Application.

<https://github.com/vam1207/HAMS.git>

1. The project fails to build due to missing dependencies. How will you fix pom.xml? [8M]
2. The hospital server is now using Java 17 instead of Java 8. Update Maven configuration. [2M]
3. A teammate added a new dependency, but Maven keeps using the old cached one. How will you refresh it? [2M]
4. You want to skip JUnit tests temporarily during build. Which Maven command will you use? [5M]
5. If two dependencies conflict in version, how will you resolve it in Maven? [5M]
6. To enable JSP with JSTL in your app, which dependencies will you add? [3M]

**Part III – Git & GitHub Integration [15 Marks]**

You are collaborating on the HAMS project with GitHub.

1. Initialize Git in your project and push it to GitHub. [5M]
2. You wrote “Added Doctor Modul” instead of “Added Doctor Module” in a commit message. Since you haven’t pushed, how do you fix it? [2M]
3. Which Git command shows modified but not committed files? [1M]
4. Show Git command to view commit history in a compact one-line format. [2M]
5. You accidentally deleted AppointmentServlet.java but haven’t committed. How do you recover it? [2M]
6. Clone your teammate’s GitHub repo and switch directly to feature/patient branch. [3M]

**Part IV – Git Collaboration, Patch & Merge Conflict Resolution [20 Marks]**

1. Create a new branch feature/billing and switch to it. [3M]
2. Show command to list both local and remote branches. [2M]
3. You mistakenly created feature/billing twice. Delete the extra local branch. [2M]
4. A bug is found in BillingService.java. Create a patch file and share it. [5M]
5. Your teammate sent you a patch. Apply it in your repo. [3M]
6. Combine multiple small commits into one before pushing. Which Git command will you use? [3M]
7. Clone the main hospital repo to your system. Which Git command will you use? [2M]

**Part V – Dockerization of Maven Application [15 Marks]**

1. Write a Dockerfile for HAMS (Tomcat + WAR deployment). [5M]
2. Build the Docker image and run the container. Verify app at http://localhost:8081. [5M]
3. Push the Docker image to your Docker Hub. [5M]

**Part VI – Docker Compose Multi-Container Setup [15 Marks]**

1. Write a docker-compose.yml with: [5M]
   * Service 1: HAMS app container (on port 8081).
   * Service 2: MySQL (username: hams\_user, password: hams\_pass, DB: hamsdb).
2. How will you run both containers? [2M]
3. If you update the app code, how do you rebuild services in Docker Compose? [3M]
4. How do you confirm patient records persist after restarting containers? [3M]
5. Which command stops all containers but keeps database data? [2M]

**Part I – SRS (10 Marks)**

**Q1. Abstract (2M)**  
The **Hospital Appointment Management System (HAMS)** is a web-based system that simplifies appointment booking, doctor availability, patient records, billing, and reporting. It reduces manual errors, avoids long queues, and provides an efficient way for patients, doctors, and staff to manage healthcare processes digitally.

**Q2. Functional Requirements (2M)**

1. Patient registration and login.
2. Appointment booking and cancellation.
3. Doctor schedule management.
4. Billing and payment tracking.
5. Report generation (daily/weekly/monthly).

**Q3. Non-Functional Requirements (2M)**

1. Security (encrypted data and secure login).
2. High availability and reliability.
3. Scalability for more patients/doctors.
4. User-friendly interface.
5. Quick response time (<3 sec for major actions).

**Q4. Identification of Users (2M)**

* **Patients** → Book/cancel appointments, view bills.
* **Doctors** → Manage schedules, check patient records.
* **Admins/Staff** → Manage doctors, patients, and reports.

**Q5. Modules (2M)**

* Patient Module
* Doctor Module
* Appointment Module
* Billing Module
* Reports Module

**Part II – Maven Web App (25 Marks)**

**Q1. Fix missing dependencies in pom.xml (8M)**  
Add dependencies inside <dependencies>:

<dependency>

<groupId>javax.servlet</groupId>

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

<version>4.0.1</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>jstl</groupId>

<artifactId>jstl</artifactId>

<version>1.2</version>

</dependency>

**Q2. Configure Java 17 (2M)**

<properties>

<maven.compiler.source>17</maven.compiler.source>

<maven.compiler.target>17</maven.compiler.target>

</properties>

**Q3. Refresh Maven cache (2M)**

mvn clean install -U

**Q4. Skip JUnit tests (5M)**

mvn clean install -DskipTests

**Q5. Resolve version conflicts (5M)**  
Use <dependencyManagement> in pom.xml to force a version.

**Q6. Add JSP + JSTL dependencies (3M)**

<dependency>

<groupId>javax.servlet.jsp.jstl</groupId>

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

<version>1.2.1</version>

</dependency>

<dependency>

<groupId>org.glassfish.web</groupId>

<artifactId>jstl-impl</artifactId>

<version>1.2</version>

</dependency>

**Part III – Git Basics (15 Marks)**

**Q1. Initialize + push project (5M)**

git init

git remote add origin https://github.com/user/HAMS.git

git add .

git commit -m "Initial Commit"

git push -u origin main

**Q2. Fix commit message (2M)**

git commit --amend -m "Added Doctor Module"

**Q3. Show modified files (1M)**

git status

**Q4. Compact history (2M)**

git log --oneline

**Q5. Recover deleted file (2M)**

git checkout -- AppointmentServlet.java

**Q6. Clone + switch to branch (3M)**

git clone -b feature/patient https://github.com/teammate/HAMS.git

**Part IV – Git Collaboration (20 Marks)**

**Q1. Create + switch branch (3M)**

git checkout -b feature/billing

**Q2. List branches (2M)**

git branch -a

**Q3. Delete duplicate branch (2M)**

git branch -d feature/billing

**Q4. Create patch file (5M)**

git diff > billing\_fix.patch

**Q5. Apply teammate’s patch (3M)**

git apply billing\_fix.patch

**Q6. Combine commits (3M)**

git rebase -i HEAD~3

**Q7. Clone main repo (2M)**

git clone https://github.com/hospital/HAMS.git

**Part V – Dockerization (15 Marks)**

**Q1. Dockerfile (5M)**

FROM tomcat:9-jdk17

COPY target/HAMS.war /usr/local/tomcat/webapps/

EXPOSE 8081

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

**Q2. Build + run (5M)**

docker build -t hams-app .

docker run -d -p 8081:8080 hams-app

**Q3. Push to Docker Hub (5M)**

docker tag hams-app deepthisagar7/hams-app

docker push deepthisagar7/hams-app

**Part VI – Docker Compose (15 Marks)**

**Q1. docker-compose.yml (5M)**

version: '3'

services:

hams-app:

image: deepthisagar7/hams-app

ports:

- "8081:8080"

depends\_on:

- mysql-db

mysql-db:

image: mysql:8

environment:

MYSQL\_ROOT\_PASSWORD: root

MYSQL\_USER: hams\_user

MYSQL\_PASSWORD: hams\_pass

MYSQL\_DATABASE: hamsdb

volumes:

- db\_data:/var/lib/mysql

volumes:

db\_data:

**Q2. Run containers (2M)**

docker-compose up -d

**Q3. Rebuild services after code update (3M)**

docker-compose up -d --build

**Q4. Confirm persistence (3M)**  
Restart containers → check MySQL table data remains.

**Q5. Stop all containers but keep data (2M)**

docker-compose down