**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-1**

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

A restaurant wants to develop an **Online Food Ordering System (OFOS)** to handle menu browsing, food ordering, payments, and delivery tracking. Based on this case:

1. What should be the purpose of this project? Write a short abstract explaining it. [2M]
2. If you were designing this system, what key features (functional requirements) would you include to solve these problems? [2M]
3. How should the system behave in terms of non-functional requirements? [2M]
4. Who are the different types of users that would interact with the system, and how would each use it? [2M]
5. How would you break this system into modules? [2M]

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

You are given a GitHub repository link for the OFOS project. You are given the following GitHub repository link:  
<https://github.com/vam1207/OFOS.git>

You have imported it into Eclipse/IntelliJ.

1. The project fails to build due to dependency errors. Fix the errors in the pom.xml. [8M]
2. The app is running on Java 8, but your team has migrated to Java 17. Update Maven configuration accordingly. [2M]
3. A teammate updated pom.xml with a new dependency, but your local Maven cache still uses the old version. How will you force Maven to update it? [2M]
4. You want to skip tests during Maven build to save time. Which command or configuration will you use? [5M]
5. Maven shows a dependency conflict between two versions of the same library. How can you resolve it? [5M]
6. You need to add **JSTL** support for JSP pages. Which Maven dependencies will you include? [3M]

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

You are working with GitHub on menu.jsp of OFOS project. Using Git, handle these scenarios:

1. You start coding a new feature but realize the project is not under version control. Initialize a Git repo and push it to GitHub. [5M]
2. You accidentally wrote “Added Payement Module” instead of “Added Payment Module” in your commit message. Since you haven’t pushed yet, how do you fix it? [2M]
3. You need to check which files have been modified but not yet committed. Which Git command will you use? [1M]
4. You want to quickly see the history of commits in a compact view. Which command will you use? [2M]
5. You deleted menu.jsp by mistake but haven’t committed yet. How do you recover it from Git? [2M]
6. You want to clone your teammate’s repo and switch directly to the feature-payment branch. Which steps will you follow? [3M]

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

You and your teammate are coding OFOS from two different GitHub accounts. Handle these scenarios:

1. You are assigned to add a delivery tracking feature without affecting the main code. Create a branch named feature/delivery and switch to it. [3M]
2. You want to list all branches (local and remote) to check available branches. Which Git command will you use? [2M]
3. You accidentally created the branch in the wrong location. How do you delete the local branch feature/delivery safely? [2M]
4. A bug is found in OrderServlet.java. Create a patch file and share it with your teammate. [5M]
5. Your teammate sent you a patch file. How do you apply it to your local repository? [3M]
6. You want to combine multiple commits into one before pushing to the main repository. Which Git command helps? [3M]
7. You want to make a local copy of a remote repository to your machine so you can work on it. [2M]

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

Your OFOS application is complete, but deployment must be containerized:

1. Write a Dockerfile to build a Docker image for OFOS using Maven. [5M]
2. Build the image and run the container. Verify OFOS works at http://localhost:8080. [5M]
3. Your manager wants the same app to run on another developer’s machine. Push the image to Docker Hub. [5M]

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

Now the OFOS must work with a database backend using Docker Compose:

1. Write a docker-compose.yml file with two services: [5M]
   * Service 1: Your OFOS Docker image (from Docker Hub).
   * Service 2: A MySQL database with username ofos\_user, password ofos\_pass, and DB name ofos\_db.
2. How do you run both containers? [2M]
3. Suppose you make changes in OFOS Docker image, how do you rebuild the service using Docker Compose? [3M]
4. Verify orders data persists after restarting containers. [3M]
5. You want to stop all running containers but keep the data volume. Which command will you use? [2M]

**KEY**

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

1. **Abstract [2M]**  
   The purpose of the Student Course Management System (SCMS) is to automate course registrations, attendance, grading, and scheduling. It provides timely notifications to students, simplifies reporting for faculty, and ensures smooth academic management.
2. **Functional Requirements (any 5) [2M]**
   * Student course registration and enrollment.
   * Attendance tracking system.
   * Grade management (upload & view results).
   * Automated notifications for deadlines.
   * Schedule management (class timetable, exams).
3. **Non-Functional Requirements (any 5) [2M]**
   * **Usability**: User-friendly interface.
   * **Performance**: Quick response even with 1000+ users.
   * **Security**: Role-based login for students/faculty/admin.
   * **Scalability**: Should support more students in future.
   * **Availability**: 99% uptime during semesters.
4. **Users [2M]**
   * **Students**: Register, view courses, check grades.
   * **Faculty**: Mark attendance, upload grades, manage schedules.
   * **Admin**: Manage courses, users, system settings.
5. **Modules [2M]**
   * Registration Module
   * Attendance Module
   * Grade Management Module
   * Notifications Module
   * Schedule/Timetable Module

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

1. **Fix pom.xml errors [8M]**  
   Add correct <dependencies>, ensure <packaging>war</packaging>, use plugins for Java 17 (see pom.xml provided).
2. **Update Java 17 config [2M]**
3. <maven.compiler.source>17</maven.compiler.source>
4. <maven.compiler.target>17</maven.compiler.target>
5. **Force Maven update [2M]**
6. mvn clean install -U
7. **Skip tests [5M]**
8. mvn clean package -DskipTests

OR in pom.xml:

<skipTests>true</skipTests>

1. **Resolve dependency conflict [5M]**
   * Use <dependencyManagement> to specify version.
   * Or use <exclusions> to avoid duplicate jars.
2. **JSTL dependency [3M]**
3. <dependency>
4. <groupId>javax.servlet</groupId>
5. <artifactId>jstl</artifactId>
6. <version>1.2</version>
7. </dependency>

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

1. **Init repo & push [5M]**
2. git init
3. git add .
4. git commit -m "Initial commit"
5. git remote add origin <repo-URL>
6. git push -u origin main
7. **Fix commit message [2M]**
8. git commit --amend -m "Added Login Page"
9. **Check modified files [1M]**
10. git status
11. **Compact history [2M]**
12. git log --oneline
13. **Recover deleted index.jsp [2M]**
14. git checkout -- index.jsp
15. **Clone repo & switch branch [3M]**
16. git clone <repo-url>
17. cd <repo-folder>
18. git checkout feature-branch

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

1. **Create & switch branch [3M]**
2. git checkout -b feature/login
3. **List branches [2M]**
4. git branch -a
5. **Delete branch [2M]**
6. git branch -d feature/login
7. **Create patch [5M]**
8. git diff > bugfix.patch
9. **Apply patch [3M]**
10. git apply bugfix.patch
11. **Squash commits [3M]**
12. git rebase -i HEAD~n
13. **Clone repo [2M]**
14. git clone <repo-url>

**Part V – Dockerization [15 Marks]**

1. **Dockerfile [5M]**
2. **Build & Run [5M]**
3. docker build -t scms-app .
4. docker run -p 8080:8080 scms-app
5. **Push to Docker Hub [5M]**
6. docker tag scms-app username/scms-app:v1
7. docker push username/scms-app:v1

**Part VI – Docker Compose [15 Marks]**

1. **docker-compose.yml [5M]**  
   **Run both containers [2M]**
2. docker-compose up --build
3. **Rebuild after changes [3M]**
4. docker-compose up --build --force-recreate
5. **Verify persistence [3M]**  
   Insert data → restart containers → data still available (volume ensures persistence).
6. **Stop all containers but keep data [2M]**

docker-compose down