Saveetha School of Engineering

Saveetha Institute of Medical and Technical Sciences

Department of Computer Science Engineering

Engineer to Excel

CSA10 - Software Engineering

List of Lab Experiments

Sl. No	Question	Tools Used	CO
1	Create a Kanban Board to Visualize the Tasks. • Create Columns for To Do, In-Progress and Done. • Add Atleast 5 Sample Tasks • Move the Tasks across the Columns to Simulate the Workflow.	Jira or Trello	CO1
2	Sketch a Simple Prototype of a Bus Ticket Booking System using Figma Tool	Figma	CO1
3	The stakeholders have conflicting views on the user interface design for an E-Commerce mobile app. Create a prototype using Figma tool to discuss with the stakeholders to get their feedback and approval.	Figma	CO1
4	Create a Scrum Project in Jira. • Add a backlog with at least 5 items (e.g., "Create user registration page", "Develop API for login"). • Prioritize the backlog and create a 1-week sprint. • Move backlog items into the sprint and start the sprint. • Finally show the Screenshot of the sprint board at the start and end of the sprint.	Jira	CO2
5	Use the following requirements for a Library Management System: • Add a feature to search books by title and author. • Implement an online book reservation system. • Generate monthly reports on borrowed books for administrators. • Enable email notifications for overdue books. • Add support for QR code scanning for borrowing and returning books. • Create a user-friendly dashboard for librarians. • Allow users to review and rate books. • Integrate a chatbot for user assistance. • Develop a mobile app version of the system. • Provide multi-language support. Categorize each requirement using MOSCOW Method (Must-Have, Should-Have, Could-Have, or Won't-Have) based on the following criteria: • Impact on the users and stakeholders. • Feasibility considering time, budget, and resource constraints. Finally Submit the completed Google Sheet or Excel file with all requirements categorized and justified.	Google Sheet or Excel	CO2

6	Link Jira tasks with Confluence to streamline task tracking and progress monitoring for the Library Management System development. • Create a new page in Confluence titled "Library Management System Project Overview." • Embed at least 5 Jira issues related to the development of the Library Management System (e.g., tasks from the sprint like "Develop book search functionality," "Create user login page," etc.). • Use the Jira macro to display issues with status (e.g., "To Do," "In Progress," "Done"). • Add a progress bar in the Confluence page to visually track the completion of each embedded Jira task (e.g., percentage of tasks completed in the sprint). • Submit a screenshot of the Confluence page showing the embedded Jira tasks and the progress bar.	Jira & Confluence	CO2
7	You are designing a Task Management System for a small team. The system should include the following features: 1.User Login and Role Assignment 2.Task Creation and Assignment 3.Task Prioritization and Deadlines 4.Progress Tracking and Reporting Prioritize these requirements using the MoSCoW and Kano models in Jira.	Jira	CO2
8	You are tasked with developing an Online Learning Platform. The platform should include the following functionalities: 1. Course Enrollment and Registration 2. Video Lecture Streaming 3. Interactive Quizzes and Assignments 4. Progress Tracking Dashboard 5. Peer-to-Peer Discussion Forums 6. Certificate Generation Use Jira to categorize and prioritize these requirements using MoSCoW and Kano techniques.	Jira	CO2
9	Demonstrate how to work collaboratively in Git/GitHub on a project using the fork-and-pull request workflow. Tasks: 1. Fork an existing public GitHub repository (e.g., a sample JavaScript or Python project). 2. Clone the forked repository to your local machine using Git. 3. Create a new branch for the feature or change you want to work on. 4. Make modifications or add new features (e.g., add a function, fix a bug, or update the README). 5. Commit your changes and push the branch to GitHub. 6. Go to the GitHub repository and create a pull request to merge your feature branch into the main branch. 7. Review the pull request and provide feedback on the changes. 8. Respond to feedback by making additional commits to the	Git/GitHub	CO3

	feature branch if necessary. 9. Once the pull request is approved, merge it into the main branch. 10. Finally submit the link to the pull request along with a summary of the changes you made and how you collaborated.		
10	Demonstrate how to work with Git branches and resolve merge conflicts when collaborating with others. Tasks: 1. Clone a shared repository to your local machine. 2. Create a new branch and switch to it. 3. Make changes to a file (e.g., update a README or modify code in a specific function). 4. Commit your changes. 5. Push the changes to the remote repository. 6. Before merging, pull the latest changes from the main branch. 7. Switch back to your feature branch. 8. Merge the main branch into your feature branch. 9. If there are conflicts, resolve them manually by editing the conflicting files. After resolving conflicts, mark the conflicts as resolved. 10. Commit the resolved merge. 11. Push your feature branch with the merged changes to GitHub and create a pull request. 12. Submit a summary of the steps you performed, the conflicts you encountered, and how you resolved them.	Git/GitHub	CO3
11	Create a Static Website and Containerize, Build & Serve it using Docker. Tasks: 1. Create a Simple Static Website (index.html file) with basic HTML content. 2. Write/create a Dockerfile to serve the website using Nginx. 3. Build the Docker Image 4. Run the container: 5. Access the Website using a Browser	Docker	CO3
12	Create a Simple Python Flask API, Containerize the Application, Build & Push the Image using Docker and Deploy the Application using Kubernetes. Tasks: 1. Create a Simple Flask API by writing a Python file (app.py) with basic endpoints. 2. Containerize the Flask App using Dockerfile. 3. Build the Image using Docker. 4. Push the Image to Docker Hub. 5. Create Kubernetes manifests (Deployment YAML & Service YAML) to deploy the application. 6. Apply the Manifests and Access the API via NodePort.	Docker & Kubernetes	CO3
13	Set up a CI/CD pipeline to automate the building, testing, and deployment of a containerized application. Tasks: 1. Set up Jenkins on a local machine or server.	Docker	CO4

	2. Create a Dockerfile to containerize a sample application. 3. Write a Jenkinsfile to automate the process of building the Docker container, running tests, and deploying to a cloud platform (e.g., AWS or GCP). 4. Configure Jenkins to trigger builds upon code commits or pull requests.		
14	Implement Continuous Deployment using GitHub Actions to deploy a Dockerized application. Tasks: 1. Set up a GitHub repository and push a simple Dockerized application. 2. Create a GitHub Actions workflow to automatically build and push the Docker image to Docker Hub or GitHub Container Registry. 3. Automate deployment to a cloud platform (e.g., AWS ECS, Azure Kubernetes Service, or Google Kubernetes Engine). 4. Test the CI/CD pipeline by pushing new code changes and verifying that the deployment occurs automatically.	Docker	CO4
15	Create a GitHub Repository and Implement Version Control Tasks: • Set up a repository for a team project. • Create branches for individual modules. • Merge branches with pull requests after code reviews. • Resolve conflicts during branch merging. • Document the workflow using the README file. • Submit the repository link	Git/GitHub	CO3
16	Containerize a Python Flask Application Using Docker Tasks: • Write a simple Python Flask application for a "To-Do" list. • Create a Dockerfile to containerize the application. • Build and run the Docker container locally. • Test the application's functionality inside the container. • Push the Docker image to Docker Hub. • Submit the Dockerfile and screenshots of testing results	Docker, Flask	CO3
17	Push and Pull Docker Images Using Docker Hub Tasks: • Create a Docker image for a static HTML website. • Tag and push the image to Docker Hub. • Pull the image from Docker Hub and run it on another machine. • Submit Docker commands and screenshots for each step.	Docker Hub, Docker CLI	CO3
18	Deploy a Multi-Container Application Using Kubernetes Tasks: • Create a multi-container application with a frontend, backend,	Docker, Kubernetes	CO3

	and database. • Define a Kubernetes YAML file for deployment and services. • Deploy the application using kubectl commands. • Monitor the pods and services status. • Scale the frontend container to handle increased load. • Submit YAML files and screenshots of the Kubernetes dashboard.		
19	Create a CI/CD Pipeline Using GitHub Actions Tasks: • Set up a GitHub repository for a containerized Flask application. • Define a GitHub Actions workflow file to automate testing and deployment. • Add steps to build, test, and push Docker images to Docker Hub. • Deploy the application to a cloud platform (Heroku/AWS). • Submit the workflow file and deployment link.	GitHub Actions, Docker	CO3
20	Create a new GitHub repository for a personal project. Initialize the repository with a README.md file. Then, clone the repository to your local machine, make some changes to the README.md (e.g., add a project description), and push the changes back to GitHub. Tasks: • Create a GitHub repository. • Clone the repository to your local machine. • Edit the README.md file. • Stage, commit, and push your changes to GitHub.	Git/GitHub	CO3
21	Clone an existing GitHub repository to your local machine and make a small change in one of the files. Tasks: • Choose an existing public GitHub repository (e.g., your own or a sample repository). • Clone the repository to your local machine using git clone <repository_url>. • Navigate into the cloned repository directory and open one of the files. • Make a small edit, such as modifying a line of text or fixing a typo. • Save your changes.	Git/GitHub	CO3
22	After modifying a file in a cloned repository, commit the changes locally and push them to GitHub. Tasks: • Clone a repository (from question 1). • Modify a file (e.g., README.md). • Stage the changes with git add. • Commit the changes with git commit -m "Updated README". • Push the changes to GitHub using git push.	Git/GitHub	CO4

23	You cloned a repository earlier, and now another collaborator has made changes to it. You need to pull the latest changes from GitHub to keep your local copy up to date. Tasks: • Clone a repository (from previous questions). • Ask a collaborator to make a change and push it to the GitHub repository. • Use git pull origin main to pull the latest changes. • Verify that the changes made by your collaborator appear in your local repository.	Git/GitHub	CO4
24	In your GitHub repository, create a new branch called feature-login. Implement a simple login function in a login.py file. After completing the function, create a pull request to merge feature-login into the main branch. Tasks: • Create a new branch feature-login. • Add a login.py file with a simple login function. • Commit the changes and push them to GitHub. • Create a pull request from feature-login to main and merge it.	Git/GitHub	CO4
25	You and your team are working on a project. One team member adds a new feature to the project. You need to fetch the latest changes, create a new branch, and implement your feature without affecting the existing code. Tasks: • Fork a repository (you can use an existing repository or a partner's repository). • Clone the repository to your local machine. • Create a new branch for your feature. • Implement your changes and push them to your forked repository. • Submit a pull request to the original repository.	Git/GitHub	CO4