

Exp - 01

Create a kanban board to visualize the Tasks.

- Create 3 columns for To do, in progress and done.

Aim:

To create and stimulate a kanban board for task visualization and workflow management.

Apparatus Required:

A computer with internet access.

Trello or Trello software.

Procedure :

- Open trello or Trello and create a new project or board.
- Create three column : To Do, in progress and done, add at least 5 sample tasks under the To Do column.
- Move tasks to the in progress and done column as they start.
- Shift completed tasks to the 'Done' column. Save & take a screenshot of the final kanban board.

Result:

A functional kanban board was created, stimulating task workflow effectively.

Exp-08

Aim :

To design a prototype of a bus ticket booking system to visualize its functionality.

Apparatus required : Figma tool

Procedure :

- open figma and create a new project design a simple interface with features like bus - selection, ticket booking and payment.
- Add buttons and text fields for data input, include a summary page displaying booking details.
- Save the design and export the prototype as a PNG or PDF file

Submit the file as proof of completion

Result :

A functional prototype of a bus ticket booking system was created using figma.

Prototype for an e-commerce mobile app

Aim:

To design a prototype of an e-commerce mobile app interface to gather stakeholder feedback.

Apparatus required : Figma

Procedure:

- open figma and create a new project. Design screen for login, product display, shopping cart and checkout.
- use interactive elements for navigation like screens.
- share the design with stakeholders for review & feedback.
- Make modifications based on feedback and finalize the prototype.

Result:

An interactive e-commerce app prototype was created and designed based on stakeholder input.

Create a scrum project in jira.

Aim: To understand the scrum methodology by creating and managing a project in jira.

Procedure:

- open jira and create a new scrum project.
- Add atleast 5 times and create a week sprint.
- Move backlog items into the sprint & start it
- Track progress during the sprint using the scrum board.
- Capture screenshots at the beginning and end of sprint

Result:

A scrum project was successfully created, managed and completed in jira.

Categorize library management system requirements using moscow method.

Aim:

To categorize the requirements of a library management system using the moscow method based on user impact and feasibility.

Apparatus Required:

computer with spreadsheet software,
Documentation system requirements.

Procedure:

- List all the provided requirements for the library management system in a spread sheet.
- Refine criteria for must-have, should-have, could-have & won't have features.
- Categorize each requirement based on impact on stakeholders by feasibility. Within time budget and resources constraints.

Result:

The library management system requirements were effectively categorized and justified using Moscow method.

Exp - 06

Link Jira tasks with confluence.

Aim: To streamline task tracking by linking Jira issues with confluence accounts for project management.

Goal

Object

Tool

Apparatus required:

- computer with internet access
- Jira and confluence account.

Procedure:

- Create a new page in confluence titled Library management system project overview.
- In Jira, create at least 5 issues related in the system.
- Use the Jira macro to embed these issues into the confluence page displaying their status.
- Save the confluence page and screenshot.

Result:

The project tasks were taskly, successfully, linked b/w Jira and confluence, hence enabling different tracking & monitoring.

Expt-04

Project task management system features using moscow and kano model.

The project feature of a task management

Aim:

System using moscow and kano model.

Applications required;

Computer with good internet access,

Ghra software.

Procedure:

→ Last the system requirements user login, task assign-

ment, projectization and program tracking.

→ Catalogize the features using the features. using

the moscow model. → Apply the kano model to access user satisfaction levels for each feature.

→ Input the projectized feature into gra as task.

Result:

The task management system features were projectized effectively using the moscow and kano

model.

Exp - 08

Categorize online learning platform features using moscow and kano models.

Aim:

To prioritize features of a online learning platform using the moscow and kano methods

Apparatus required:

1. Jira software
2. Computer with internet access.

Procedure :

List the required features : course enrolment video streaming quizzes, progress, tracking forums and certificates.

Categorize each feature using the moscow method.

use the kanomodel to access user satisfaction and desirability for each feature.

Create a sprint plan and assign tasks for implementation.

Result :

The features of the online learning platform were prioritized using Moscow and kanomodel & tasks were organised for development.

Q. Demonstrate collaborative work using Git/Github.

Aim:

To demonstrate collaboration using the fork and pull request workflow on github.

Tool

JDK

Apparatus Required :

Git and Github accounts
Computer with git installed.

Procedure :

- * Fork a public Github repository
- * Clone the forked repository to your local Machine
- * Create a new branch for your changes.
- * Make modifications or add a new feature.
- * Commit your changes & push the branch to Github.
- * Create a pull request to merge the branch into the main repository.
- * Response to feedback, if any and finalize the pull request.

Result :

Collaborative work was demonstrated using the git fork & pull request workflow

10. Resolve merge conflict using Git.

Aim : To learn how to handle and resolve merge conflict during collaboration in Git.

Procedure :

Git and Github accounts computer with Git installed.

Apparatus required :

clone and shared repository to your local machine

Create a new branch and switch to it
Make changes to a file.

Commit and push the changes to the repository
Pull the latest changes from the main branch
before merging

Merge the resolved branch to Github & create
a pull requested.

Result :

Merge conflict were resolved successfully
and changes were merged into the main
branch.

11. Containerize and serve a static website using

Dockee.

To containe a static website and serve it using docker.

Apparatus Required:

Procedure:

Basic knowledge of HTML & Docker file.

Procedure:

- * Create a simple static website with an file.
- * Write a dockerfile to save the website using Migrx. Build the docker image using the common docker build -t static website.
- * Access the website in a browser by visiting `http:// local host :9000`

Test the container functionality & stop it after completion.

Result:

The static website was successfully centralized and served using Docker.

QUESTION

18. Deploy a flask API using Docker & kubernetes.

Aim : To develop a simple flask API, containerizing it using Docker and deploy it with kubernetes.

AIMS

Apparatus Required :

Python & flask, Docker & kubernetes

CLI, kubernetes cluster .

Procedure :

Write a basic flask API with at least one endpoint.

Create a dockerfile to containerize the flask application build and push the docker image to docker hub .

Create kubernetes YAML manifest for deployment and services. Apply the Manifests using the kubernetes apply deployment YAML. Access the API using the assigned endpoint.

Result :

The flask API was successfully deployed and accessed via kubernetes .

13. set up a ci/cd pipeline using jenkins.

Aim: To automate the building, testing and deployment of a containerized application using Jenkins.

Apparatus Required :

Jenkins installed on a server or local machine Docker and sample application.

Procedure :

Install and configure Jenkins on a server or local machine

Write a dockerfile for a sample application.

Create a jenkins file to define the build in tasks, and deployment stages.

Configure Jenkins to trigger builds on code commits or pull requests.

Test the pipeline by committing code to the git repository

Verify the deployment of application.

Result :

A ci/cd pipeline was successfully setup and verified using Jenkins.

15.

Implement continuous deployment using Github actions.

Time
Date

9/11

10/11

11/11

12/11

13/11

14/11

15/11

Aim: To automate the deployment of a dockerized application using Github actions.

Apparatus required: Github repository with a Dockerized application Github action.

Procedure:

- * Create a new Github repository and push the dockerized application code.
- * Write a Github actions workflow file to automate build and deployment.
- * Configure the workflow to build the docker image and push it to docker hub.
- * Automate deployment was successfully implemented and tested using Github actions.

Result:

Continuous deployment was successfully implemented & testing using Github actions.

16. Create a GitHub repository and implement version control.

Aim: To demonstrate version control in GitHub by setting up a repository and managing branches.

Apparatus Required:

Github account, computer with Git installed.

Procedure:

- * Create a new GitHub repository and initialize it with a `readme.md` file.
- Clone separate branches for individual project modules.
- Open pull request and merge branches after code reviews.

Document the workflow in the `readme.md` file.

Result:

Version control was successfully implemented using GitHub with multiple branches and pull requests.

docke

successfully

centralize and deployed using

The python flask application was

Result:

push the docker image to docker hub.

docker sum -p 5000:5000 flask to do app

Run the docker container locally with

to do app

* Build the docker image using docker build. Flash action.

* Write a docker file to centralize the flask application.

* Create a simple flask application of "To-Do" first.

Procedure:

→ Docker software.

→ Create required python and flask installed

Application Required:

action using Docker

To centralize a sample python flask application

file:

Containerize a python flask application using for

gitHub with multiple branches and pull requests.

19.

and pull a from Docker hub.

Docker image were successfully pushed to

Results:

* Document the steps and submit somewhere

* Run the pull image to verify functionality

pull

* Push the image on another machine using docker

* Push the image to docker hub using docker push

latest/static - website

* Tag the image using docker tag static website

* Create a docker image for a static HTML website

Procedure:

account -

application required: Docker installed, Docker hub.

pulling docker image using Docker hub.

To demonstrate the process of pushing and

ARM:

Push and pull docker image using Dockerhub.

81.

Ques : Deploy a multi - container application using kubernetes.

Aim : To deploy a multi - container application with frontend, backend component using kubernetes.

Apparatus Required :

* Dockerized Multi - container application

* Kubernetes CLI and cluster.

Procedure :

* Create docker image for the frontend, backend and data base.

Write kubernetes YAML files for deployment and service configuration

Deploys the application using kubectl apply -f deployment YAML.

Monitor pods and services with kubectl get pods and kubectl get services

Result :

The Multi - container application was deployed and scaled successfully using kubernetes.

Q4. Create a CI/CD pipeline using Github Actions.

Aim: To automate the testing and deployment of a containerized application using Github actions.

Apparatus Required:

Github repository, github actions enabled

Procedure :

- * Set up a github repository for containerized flask application
- * Create a github actions workflow file.
- * Define steps to build test and push the docker hub .
- Trigger the workflow by pushing changes to the repository .
- Verify the deployment and submit workflow logs .

Results :

The CI/CD pipeline was successfully implemented & tested using github actions

Initialize and update a github repository

Aim: To setup and update a github repository a personal project.

Apparatus Required:

- Github account computer with git installed
- Created a github repository and initialize with a `read me.md` file.
- Clone the repository to your local machine using git clone.
- Edit the `readme.md` file to add project details.
- Stage and commit changes with git add and git commit.
- Push the changes to github using git push.

Result:

The github repository was successfully initialized, updated and documented.