Exp-01

Create a kanban board to visualize the Tasks.

• Create a 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.

Fira or Trello software.

Procedure:

→ Open fira or Trello and create a new project or board.

→ Create three columns: To Do, in progress and domain, 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 solve & take a screenshot of the final kanban board.

Result:

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

Exp-02

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.

Ex-03

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 blw 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 refined based on stakeholder input.

Exp -04

Create a scrum project in jira

Aim:

To understand the scrum methadology by creating and managing a project in jira.

Procedure:

→ Open jira and create a new scrum project.

→ Add atleast 5 times and create a 1week sprint.

→ Move backlog items into the sprint & start it

→ Track progres 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.

Exp - 05.

Categorize library management System requirments using moscow method.

Aim:

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

Apparatus Required:

Computer with spreadsheet software, Documentation System requirments.

Procedure:

→ List all the provided requirments for the library management system in a spread sheet.

→ Define criteria for must-have, should-have, could-have & won't have features.

→ Categorize each requirments based on impact on stackholders & 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.

Apparatus required:

→ computer with internet access
→ jira and confluence accounts.

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 conference page and screenshot.

Result:

The project tasks were taskly, successfully, linked b/w pro and confluence; license enabling different tracking or monitoring.

Exp-07

Prioritize task management system features using moscow and kano models.

Aim:

The prioritize feature of a task management system using Moscow and kano models.

Apparatus required:

Computer with good internet access, jira software.

Procedure:

→ List the system requirements user login, task assignment, prioritization and progress tracking.

→ Categorize the features using the features using the moscow model.

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

→ Input the prioritized features into jira as tasks.

Result:

The task management system features were prioritized effectively using the moscow and kano models.

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 acces.

Procedure:

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

Categorize each feature using the Moscow method.

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

Create a sprint plan and assign tasks for implementation.

Result:

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

9. Demonstrate collaborative work using Git / Github.

Aim:

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

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.
* Respond 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 Gist.

Aim :

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

Appaeatus Required :

Git and Github accounts

computer with Git installed.

Procedure :

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 Docker.

Aim :

To containerize a static website and serve it using docker.

Apparatus Required :

Computer with Docker installed.

Basic knowledge of HTML & Docker file.

Procedure :

* create a simple static website with an file.
* Write a dockerfile to serve the website using Nginx. Build the docker image using the common docker build -t static website.
* Access the website in a brower by visiting http://local host:8080
* Test the container functionality & stop it after completion.

Result :

The static website was successfully containerized and served using Docker.

12. Deploy a flask API using Docker & kubernetes.

Aim :

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

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 nodepoint.

Result :

The flask API was successfully deployed and accessed via kubernetes.

13. Set up a CI/CD pipeline using jinkins.

Aim:

To automate the building, testing and deployment of a containerized application using Tenkins.

Apparatus Required:

Tenkins installed on a server or local Machine
Dockes 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 comitting code to the repository

Verify the deployment of application.

Result:

A CI/CD pipeline was successfully setup and verified using Tenkins

15. Implement continuous deployment using Github actions.

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.
* Wite 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 succesfully implemented and tested using Github actions.

Result:

Continuous deployment was succesfully 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 bit installed.

Procedure:

* Create a new github repository and initialize it with a readline nd 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.

19. Containerize a python flask application using for github with multiple branches and pull requests.

Aim:

To containerize a sample python flask application using Docker.

Apparatus Required:

→ Create required python and flask installed
→ Docker software.

Procedure:

* Create a simple flask application to "TO-DO" first.
* Write a docker file to containerize the flask application.
* Build the dockers image using docker build. Flask to do app.

Run the docker container locally with docker run -p 5000:5000 flask to do app.

Push the docker image to docker hub.

Result:

The python flask application was successfully containerized and deployed using docker.

81. Push and pull docker images using Dockerhub.

Aim:

To demonstrate the process of pushing and pulling docker images using docker hub.

Apparatus required: Docker installed, Docker hub. account.

Procedure:

* Create a docker image for a static HTML website
* Tag the image using docker tag static website latest | static-website
* Push the image to docker hub using docker push
* Push the image on another machine using docker pull.
* Run the pulled image to verify functionalities
* Document the steps and submit screenshots

Results:

Docker image were succesfully pushed to and pulled from docker hub.

23. Deploy a Multi- container application using kubernets.

Aim:

To deploy a multi-container application with frontend, backend components using kubernets.

Apparatus Required:

* Dockerized Multi-container application
* kubernetes CLI and cluster.

Procedure:

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

Write kubuctneter YAML files for deployments and services configuration

Deploys the application using kubectl apply -f deployment YAML.

Monitor pods and services with kubectl get pocks and kubetl gets services

Result:

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

24. 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 got add and git commit.

→ Push the changes to github using gitpush.

Result:

The github repository was successfully initialized, updated and documented.