13.12.24

Expt1: Kanban Board workflow Simulation

Aim :- To simulate task management for the Book store API Project using Kanban board to visualize and track task progress.

Procedure :-

Software used :- Kanban board tool - jira

Procedure :-

1. Create 3 Columns : To Do, In Progress and Done
2. Added 5 Tasks :
   - Define open API Specification
   - Render and Test API in Swagger UI.
   - Take Screen shots of API Testing
   - write YAML Specification file.
   - Submit Deliverable.
3. Moved task across Columns sequentially to simulate Progress.

Result :-

> Task were visualized clearly, ensuring efficient tracking
> Dependencies were resolved naturally by progressing sequentially.
> The workflow was streamlined without delays or overlaps.

13.12.24

Expt2: Simple Bus ticket Booking System Prototype

Aim:- Create simple bus ticket booking system prototype in figma.

Tool: Figma

Procedure:-

1. Design Screen: Create Screens for home, search results, seat selection, passenger details, payment and confirmation
2. Create Interaction: Connect Screens with transitions and button clicks
3. Test and iterate: Test the Prototype for usability and make improvements.

Result:-

A functional prototype to visualize user flow and gather feedback.

Expt 4: E-Commerce Mobile App Prototype Development.

Aim:-

To create an interactive prototype for an e-commerce mobile app to gather stakeholder feedback and achieve design consensus.

Tool used:- Figma

Procedure:-

1) Develop low-fidelity wireframes to outline the app structure.
2) Create detailed designs using Figma's components and styles.
3) Utilize Figma's interactive features to simulate user flows and transitions.
4) Share the prototype with stakeholders, allowing them to comment directly on the design.

Result:-

The prototype facilitated effective discussions with stakeholders, leading to valuable insights and refinements in the design process, ensuring alignment with user experience goals.

13.12.24

Expt 3:- implementing Scrum Project in Jira

Aim:- To demonstrate the use of jira to manage a Scrum Project, from backlog creation to sprint completion.

Tool used: Jira

Procedure:-

1. Create Scrum project
2. Add backlog items and prioritize them.
3. Create a 1-week sprint.
4. Start the sprint and assign tasks
5. Monitor Progress and hold daily stand-ups
6. End the sprint and conduct a retrospective.

Result:-

Successful Scrum project execution visualised in jira.

14.12.24

Expt : Requirement categorization for library Management System.

Aim:-

To categorize the requirement of the library management system using the MoSCow method for prioritization based on user impact and feasibility.

Tool used:- google sheet or excel.

Procedure:-

> Compile all system requirement.
> Assess each requirement based on user impact and feasibility.
> Classify each requirement into one of 4 categories: Must-have, Should-Have, Could-Have or won't-have.
> Provide brief justification for each categorization.
> Documentation: Input the categorized requirement and justification into google sheet or excel file.

Result:-

The categorized requirement provide clear prioritization framework, facilitating focused development effort while ensuring alignment with user needs and Project constraints.

Expt : Library Management System

Aim :- To Create "Library Management System Project overview" Page in Confluence that embeds jira tasks for efficient tracking and Progress monitoring.

Tool used :- Confluence and jira.

Procedure :-

> Set up a new Confluence Page title "Library Management System Project overview".

> use the jira macro to embed at least five relevant tasks (Eg: "Develop book search functionality", "Create user login Page")

> ensure each task display it status (To DO, in Progress, Done).

> Insert Progress bar to indicate the overall completion percentage of tasks.

Result :-
This integration provide clear visibility into Project Progress, enhancing communication and task management among team members and stake holders.

Task management System

Aim:- To Prioritize the features of Task management System using the moscow and kano model in jira.

Tool used:- Jira

Procedure:-

> Identify the features to be Prioritize :-

- User login and Role Assignment
- Task Creation and Assignment
- Task Prioritization and Deadlines
- Progress Tracking and Reporting

1. MoSCow Prioritization:-

- Must - Have :
  - * user login and Role assignment
  - * Task Creation and Assignment.
- Should - Have :-
  - * Task Prioritization and Deadline
- Could - Have :-
  - * Progress Tracking and Reporting
- Won't - Have :

  none for this phase

2. Kano Model Analysis :-

- Basic need :

  >user login and Role Assignment

  Performance needs :-

  > Task Creation and Assignment

  > Task Prioritization and Deadlines

encitement needs.

- Progress Tracking and Reporting

Result:-

The Prioritization helps focus development efforts on essential features first, ensuring functional system that meets user needs while allowing for future enhancements.

Online learning Platform Requirement.

Aim:-

To categorize and prioritize functionalities of online learning Platform using MosCow and Kano models in Jira.

Tool used:- Jira.

Procedure:-

> Identify key functionalities:-

2. MosCow Prioritization:-

Must-Have:

> Course enrollment and registration
> video lecture streaming

Should-Have:-

> Interactive Quizzes and Assignment.
> Progress Tracking Dashboard

could-Have:

> Peer-to-Peer Discussion Forums

Won't Have:-

> Certificate generation (for initial Launch)

3. Kano Model Analysis:-

Basic needs:-

> Course enrollment and registration

Performance needs:-

> video lecture streaming
> Interactive Quizzes and Assignment
> Progress Tracking Dashboard.

Excitement needs:-

> Peer-to-Peer Discussion Forum
> Certificate Generation

Result:-

This Prioritization framework ensure that essential feature are developed first, aligning with user expections and enhancing overall satisfaction with platform.

Fork - and - Pull request workflow in Github.

Aim:-

To demonstrate the Process of Collaborative development in git/github by utilizing fork - and - pull request workflow.

Tool used:- git and github

Procedure:-

> Create personal copy of public repository on github.
> use git to clone the forked repository to your local machine.
> Develop new feature or fix by creating seperate branch.
> implement change or add feature in codebase
> Stage and commit your modification with descriptive message.
> Push new branch with change back to your fork on github
> open pull request to propose merging changes into original repository.

Result:-

This workflow enables effective collaboration among developers, facilitate code review and enhancing Project management through structured ~~contribu~~ contribution.

Git Branches and resolving Merge Conflicts.

Aim:-

To demonstrate collaboration using git branches and how to resolve merge conflicts.

Tool used:- Git and github

Procedure:-

1. Clone repository
2. Create and Switch Branch:
3. Make change : Update file (README)
4. Commit Changes
5. Push Changes
6. Pull latest change from Main
7. Switch Back to features Branch
8. Merge Main into Feature Branch
9. Resolve conflicts
10. Commit Resolved Merge
11. Push merged change

Result:-

Successfully demonstrated branch management and conflict resolution in git, facilitating effective collaboration in team environment.

Static website with Docker

Aim: -

To create simple static website and serve it using Docker and Nginx.

Tool used: Docker, Nginx

Procedure: -

> Create an 'index.html' file with basic HTML content.
> Create 'Dockerfile' to serve the website using Nginx.
> Run the following command to build the Docker image
> start container from the image
> open a web browser and navigate to 'http://localhost:8080' to view the website.

Result: -

Successfully created and containerized static website using Docker, enabling easy deployment and access through a web browser.

Flask API

Aim:- To create simple Flask API, containerize it with Docker, Push the image to Docker Hub and display it using Kubernetes.

Tool used:- Python, Flask, Docker, Kubernetes.

Procedure:-

1. Write a Python file ('app.py') with basic end points
2. Create 'Dockerfile'.
3. Build the image by Running the Command.
4. Push the image to Docker Hub.
5. Write 'deployment.yaml' and 'service.yaml'.
6. Use Kubectl to deploy.
7. Open browser and navigate to 'http://<node_IP>:30000/api'.

Results:-

Successfully created and deployed a Flask app i using Docker and Kubernetes, enabling scalable access to the API via a NodePort service.

Build and Deploy a Dockerized Application.

Aim:- To create a Dockerfile for sample application, automate the build and deployment process using a Jenkinsfile, and configure Jenkins for continuous integration.

Tool used:- Docker, Jenkins

Procedure:-

1. Write 'Dockerfile' to containerize the application
2. Create 'Jenkinsfile' to automate building, testing and deploying
3. Set up Jenkins to trigger builds on code commits or pull request by configuring webhooks in your git respository setting.

Results:-

Successfully created a Dockerized application with an automated CI/CD pipeline using Jenkins, allowing for efficient builds, tests and deployments to cloud platforms.

Continuous Deployment of Dockerized.

Aim:-

To implement continuous deployment for Dockerized application using github Action, automating the build and deployment process.

Tool used:- github, Docker, github Actions.

Procedure:-

> Create new github repository and push simple Dockerized application

> Add 'github / workflow / main.yml' file to automate building and pushing the Docker image.

> Extend the workflow to deploy the application to Cloud platform (eg AWS ECS) after the image is built.

> Push new code changes to main branch and verify that the deployment occurs automatically.

Results:-

Successfully implemented CI/CD pipeline using github Actions for Dockerized application, enabling automated builds and deployments

implement version control.

Aim:-

To set up github repository for team Project, manage branches and document the work flow.

Tool used: git hub

Procedure:-

1. Create new repository on github and initialize it with README file.
2. Each team member create individual branches for their respective modules
3. After completing their work, team members submit pull request for code reviews.
4. If there are merge conflicts, resolve them manually by editing the conflicting files and commiting the changes.
5. Update the README file to include the version control workflow and guidelines for collaboration.

Result:-

Successfully established version control system using github, enabling efficient collaboration among team members with clear documentation of the workflow.

To-Do Application.

Aim:- To create simple Python Flask application for "To-Do" list, containerize it using Docker and push the image to Docker Hub.

Tool used : Docker, Flask.

Procedure:-

> write simple Flask app ('app.py') for "To-Do" list.

> write 'Dockerfile' to containerize the application.

> Run the following command in the terminal.

> Store the container.

> Access the API using Postman or curl.

> Push the Docker image to Docker Hub.

Result:-

Successfully created and containerized Flask To-Do application, tested its functionality within the container and Pushed the Docker image to Docker Hub.

Pushing and Pulling Docker images.

Aim :-

To create Docker image for static HTML website, push it to Docker Hub and pull it from Docker Hub on another machine.

Tool used :- Docker , Docker hub

Procedure :-

1. Create a directory with an 'index.html' file.
2. Write 'Dockerfile' to serve the static website using Nginx.
3. Then Build Docker image, Run the Command ~~to be~~
4. Tag the image and Push it to Docker hub.
5. On another machine, pull image from Docker Hub.
6. Start Container from the pulled image.

Result :-

Successfully Created Docker image for static HTML website Pushed it to Docker Hub and pulled it on another machine, demonstrating effective image management using Docker.

Deploy Multi-Container Application.

Aim:-

To create and deploy multi-container application consisting of frontend, backend and database using Kubernetes.

Tool used:- Kubernetes, Docker.

Procedure:-

* Develop simple application with 3 components.
  Frontend: React or Angular app.
  Backend: Node.js or Flask server
  Database: Mongo DB or Postgre SQL database.
* Create 'deployment.yaml' file for deployment and 'service.yaml' file for services.
* Use 'Kubectl' commands to apply the YAML files.
* Check the status of the deployed pods and services.
* Increase the no. of replicas for the frontend to handle increased load.

Result:-

Successfully deployed a multi-container application on Kubernetes monitored its status and scaled components as needed, demonstrating effective management of containerized application in cloud environment.

Create CI/CD pipeline for Flask application

Aim:- To set up CI/CD pipeline for containerized Flask application using github Action, automating testing and deployment.

Tool used:- Github, Docker, Github Action.

Procedure:-

* Create new repository on github for the flask application and push the code.
* Create '.github/workflow/main.yml' file to automate the testing and deployment process.
* Add steps in the workflow to run test before deployment.
* Configure the workflow to run test before deployment.
* Provide the link to github repository and the deployed application on Heroku.

Result:-

Successfully implemented CI/CD pipeline using github Action for containerized Flask application, allowing for automated testing and deployment upon code changes.

Set up github Repository and implement version control

Aim:- To create new github repository, initialize it with README.md file and demonstrate version control by making changes and pushing them back to github.

Tool used:- github.

Procedure:-

1. go to github, click on new repository, enter name ("my-Personal-Project") add description and select initialize this repository with README. Click create repository.
2. Clone Repository by opening Terminal and run.
3. open 'README.md' in text editor and add project description, then save the changes.
4. i) Stage the change.
   ii) commit the change
   iii) Push the changes back to github.

Result:-

Successfully created a github repository, cloned it to the local machine, modified the README.md file and pushed the changes back to github, ~~dem~~ demonstrating effective use of version control.