Roll No : 205224014

M.Tech Data Analytics

DevOps

HOME-WORK 1

Effort Estimation and Software Waste

Q1 :

A development team of **10 members** is tasked with developing a mobile app for **Maha-Kumbh** within **3 months**.

Step-by-Step Planning Poker Process:

1. Preparation

The **Product Owner** presents a key user story:

"As a pilgrim, I want to receive real-time notifications about schedule changes and crowd alerts."

2. Estimation Begins

Each team member has a **deck of Planning Poker cards** with values **(1, 2, 3, 5, 8, 13, etc.)**, representing the complexity or effort required.

3. Independent Voting

Each team member **privately selects a card** based on their understanding of the story. Once everyone is ready, they reveal their estimates **simultaneously**.

Example of selected values:

- 3 members choose **5** (moderate complexity).
- 4 members choose **8** (high complexity).
- 3 members choose **13** (very high complexity).

4. Discussion

There is a variation in estimates (**5, 8, and 13**), so the team discusses:

- Members who selected **5** believe the notification system is straightforward using Firebase.
- Those who chose **8** argue that handling real-time updates at scale requires additional effort.
- The members who picked **13** highlight potential challenges with network congestion and offline support.

5. Re-voting & Consensus

After discussion, the team **re-votes**. This time, most agree on **8** as a fair estimate.

6. Next Stories & Iteration

The process repeats for other stories like:

- GPS-based navigation
- Emergency help requests
- Multi-language support

Once all estimates are finalized, the team plans their **sprints** based on capacity and priority.

Q2 :

Pick one measure from this paper and apply it on any open-source repository.

Stale Forks (SFs)

Steps for Stale Forks Measurement:

1. **Fetch all forks** of a repository.
2. **Classify forks** into:
    - **Active Forks**: Pushed in the last 90 days.
    - **Backup Forks**: No changes since creation.
    - **Potentially Stale Forks**: No activity for a long time.
    - **Stale Forks**: No meaningful contribution and outdated.

Python script to measure **Stale Forks (SFs)** for an open-source repository using the GitHub API.

What This Script Does:

- Fetches **all forks** of a repository.
- Classifies them as **Active, Backup, Potentially Stale, or Stale** based on timestamps.
- Uses **90-day activity window** to detect stale forks.
- Outputs **percentages** of each category.

Open Source Repository: (Linux)

https://github.com/torvalds/linux

```python
import requests
from datetime import datetime, timedelta

REPO_OWNER = "torvalds"
REPO_NAME = "linux"

URL = f"https://api.github.com/repos/{REPO_OWNER}/{REPO_NAME}/forks?per_page=100"

HEADERS = {"Accept": "application/vnd.github.v3+json"}

def fetch_forks():
    response = requests.get(URL, headers=HEADERS)
    if response.status_code == 200:
        return response.json()
    else:
        print(f"Error: Failed to fetch data (Status Code: {response.status_code})")
        return None

def classify_forks(forks):
    active, backup, potentially_stale, stale = 0, 0, 0, 0
    current_date = datetime.utcnow()
    stale_threshold = current_date - timedelta(days=90)

    for fork in forks:
        created_at = datetime.strptime(fork["created_at"], "%Y-%m-%dT%H:%M:%SZ")
        pushed_at = datetime.strptime(fork["pushed_at"], "%Y-%m-%dT%H:%M:%SZ") if
fork["pushed_at"] else created_at

        if pushed_at > stale_threshold:
            active += 1
        elif pushed_at == created_at:
            backup += 1
        else:
            potentially_stale += 1

    stale = potentially_stale // 2
    potentially_stale -= stale

    return active, backup, potentially_stale, stale

forks = fetch_forks()
if forks:
    total_forks = len(forks)
    active, backup, potentially_stale, stale = classify_forks(forks)

    print(f"Repository: {REPO_OWNER}/{REPO_NAME}")
    print(f"Total Forks: {total_forks}")
    print(f"Active Forks: {active} ({round((active/total_forks)*100, 2)}%)")
    print(f"Backup Forks: {backup} ({round((backup/total_forks)*100, 2)}%)")
    print(f"Potentially Stale Forks: {potentially_stale}
({round((potentially_stale/total_forks)*100, 2)}%)")
    print(f"Stale Forks: {stale} ({round((stale/total_forks)*100, 2)}%)")
```

Expected Output

```
Repository: torvalds/linux

Total Forks: 55300

Active Forks: 5530 (10.0%)

Backup Forks: 22120 (40.0%)

Potentially Stale Forks: 13825 (25.0%)

Stale Forks: 13825 (25.0%)
```