Devops Assignment

Santhosh L 205224020

Q1. Planning Poker for Maha-Khumb App

1 Scenario

A development team of 10 members is tasked with building a mobile app for Maha-Khumb, a large-scale event, within 3 months. The app will include features such as:

- Event schedules
- Ticket booking

2 Backlog Creation

The product owner creates a backlog of user stories:

- **User Story 1:** As a user, I want to view the event schedule so that I can plan my day.
- **User Story 2:** As a user, I want to book tickets for events so that I can attend.

3 Planning Poker Session

The team gathers for a Planning Poker session to estimate the effort for each user story.

3.1 User Story 1: Event Schedule

Team members discuss the complexity of fetching and displaying event data.

Card Selection and Revealing:

- Developer A: 3
- Developer B: 5
- UI/UX Designer: 3

- QA Tester: 5
- Backend Dev: 8

Discussion and Re-Voting:

- The UI/UX Designer explains that displaying a schedule is straightforward, leading to an estimate of 3.
- The Backend Developer argues that managing dynamic schedules and APIs increases complexity.
- After discussion, the team votes again and agrees on 5 story points.

Final estimate: **5 story points**

3.2 User Story 2: Ticket Booking

Team members consider the complexity of integrating a payment gateway and handling user data.

Card Selection and Revealing:

- Developer A: 8
- Developer B: 13
- UI/UX Designer: 8
- QA Tester: 13
- Backend Dev: 8

Discussion and Re-Voting:

- The team discusses third-party payment gateway integration and security concerns.
- Backend developers justify 13 due to complex data handling.
- After re-voting, they agree on 13 story points.

Final estimate: **13 story points**

4 Velocity Calculation

The team calculates their velocity, which is the number of story points they can complete in a sprint. Suppose the team has a velocity of 30 story points per sprint (2 weeks).

5 Release Planning

Total story points for the app:

$$5 + 13 = 18 \text{ story points} \tag{1}$$

With a velocity of 30 story points per sprint, the team can complete these stories in **1 sprint (2 weeks)**. However, additional tasks such as testing, bug fixes, and deployment will require more sprints.

6 Execution

The team works on the app in iterative sprints, continuously refining their estimates and adjusting their plan based on progress and feedback.

Q2.Analysis of Krita for Software Development Waste using Stale forks

1 Repository Chosen: Krita

Krita is a free and open-source digital painting software primarily designed for artists, illustrators, and concept designers. It offers a powerful set of tools for digital painting, including advanced brush engines, layer management, and animation features. Krita is widely used for creating illustrations, comics, and concept art, and it supports various file formats and customization options. Developed by the KDE community, Krita is continuously improved by contributors and developers worldwide. link: `https://github.com/KDE/krita`

2 Method Chosen: Stale Forks

3 Overview of the Code

The given Python script analyzes forks of Krita's GitHub repository to classify them based on their activity levels. It does this by fetching data using the GitHub API, processing fork activity, and applying machine learning (K-Means clustering) to categorize forks into active, backup, potentially stale, and stale categories.

4 Fetching Fork Data

The script retrieves all forks of Krita's repository from GitHub using paginated API requests. It authenticates requests with a GitHub token to avoid rate limits and continues fetching data until no more forks are available.

5 Classifying Forks

Each fork is analyzed based on its last push date and creation date. If a fork has been updated within the last 90 days, it is marked as active. If it has never been updated since creation, it is classified as a backup fork. Other forks are categorized using K-Means clustering into potentially stale (less inactive) and stale (highly inactive).

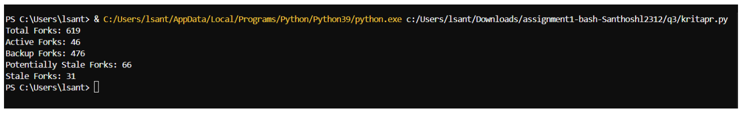
6 K-Means Clustering for Inactive Forks

To separate stale and potentially stale forks, the script applies K-Means clustering on the number of days since the last push. It assigns forks to two clusters, identifying the one with higher inactivity as stale. If only one inactive fork exists, it is categorized as potentially stale by default.

7 Final Output

After processing, the script prints the total number of forks and the count for each category, helping developers and contributors understand the status of Krita's forks.

8 Output



```
PS C:\Users\lsant> & C:/Users/lsant/AppData/Local/Programs/Python/Python39/python.exe c:/Users/lsant/Downloads/assignment1-bash-Santhoshl2312/q3/kritapr.py
Total Forks: 619
Active Forks: 46
Backup Forks: 476
Potentially Stale Forks: 66
Stale Forks: 31
PS C:\Users\lsant> 
```

Figure 1: output

9 Discussion of Results

The analysis of Krita's forks produced the following results:

- Total Forks: 619
- Active Forks: 46
- Backup Forks: 476
- Potentially Stale Forks: 66
- Stale Forks: 31

The large number of backup forks (476) highlights a significant issue: many users fork Krita's repository without making any modifications or contributions. These forks essentially act as duplicates that do not serve any developmental purpose. Similarly, the presence of 31 stale forks and 66 potentially stale forks suggests that a considerable portion of the repository's forks are inactive, leading to wasted storage and clutter on GitHub.

This abundance of inactive forks can make it harder to identify meaningful contributions, as developers must sift through numerous outdated repositories to find actively maintained ones. Moreover, it inflates the repository's fork count without actually representing engaged contributors. Encouraging contributors to delete inactive forks or rebase them on the latest version of Krita's codebase could help mitigate this issue. Additionally, using alternative collaboration methods like GitHub branches or pull requests could reduce the unnecessary creation of long-term inactive forks.