DevOps Assignment

Q1: Read about “Planning Poker” - Agile estimation technique and illustrate an example with a Development Team of 10 who are tasked to develop a mobile app for Maha-Khumb in 3 months.

Answer:

Planning Poker – Agile Estimation for Maha-Khumb Mobile App Development

Project Overview

Objective: Develop a mobile app for Maha-Khumb in 3 months
Team Size: 10 members (Product Owner, Scrum Master, 8 Developers)
Key Features:

- User Registration
- Event Schedule
- Crowd Density Alerts
- Interactive Map
- Emergency SOS Button

Step 1: Setting Up the Estimation Session

- Facilitator: The Scrum Master leads the session.
- Estimators: 8 Developers + Product Owner participate.
- Cards: Each developer holds Planning Poker cards (1, 2, 3, 5, 8, 13, 20, 40, 100).
- Baseline Selection: The team agrees that a simple user registration feature should be estimated as 2 points for reference.

Step 2: Discussing User Stories & Estimating

User Story 1: User Registration

"As a user, I want to register using my phone number or email so that I can access event details."

- Discussion: Developers clarify requirements (OTP verification? Social logins?)
- Estimation Round 1: Votes: **3, 5, 5, 5, 8, 5, 5, 5**
- Outlier Discussion: A developer who voted 8 explains concerns about security issues.
- Re-estimation: After discussion, all agree on 5 story points.

User Story 2: Event Schedule

"As a user, I want to see the event schedule with real-time updates."

- Discussion: Needs admin panel updates, push notifications.
- Estimation Round 1: Votes: **5, 8, 8, 5, 8, 5, 5, 5**
- Outlier Discussion: The developers who voted 8 explain concerns about real-time updates.
- Re-estimation: The team agrees on 6 story points (average).

User Story 3: Crowd Density Alerts

"As a user, I want to receive alerts when a location is overcrowded to avoid congestion."

- Discussion: Needs GPS tracking, live heatmaps.
- Estimation Round 1: Votes: **13, 20, 13, 20, 20, 13, 13, 13**
- Outlier Discussion: Developers voting 20 highlight integration complexity.
- Decision: Team agrees on 15 story points after adjusting scope.

User Story 4: Interactive Map

"As a user, I want a real-time map to navigate Maha-Khumb."

- Discussion: Needs live tracking, event locations, crowd zones.
- Estimation Round 1: Votes: **20, 40, 20, 40, 20, 40, 20, 40**
- Outlier Discussion: Those voting 40 raise concerns about real-time GPS accuracy.
- Decision: Since it is a critical feature, they settle on 30 story points.

User Story 5: Emergency SOS Button

"As a user, I want a quick SOS button to call for help."

- Discussion: Needs GPS location sharing, emergency contacts.
- Estimation Round 1: Votes: 5, 5, 3, 5, 5, 3, 3, 5
- Consensus: 4 story points.

Step 3: Handling Discrepancies in Estimates

- Example: If one developer consistently gives very high estimates, the team discusses if they need more clarity or research.
- Action: If needed, a feature is put on hold for further exploration (e.g., Crowd Alerts need more research).

Step 4: Finalizing the Sprint Plan

Total Effort Estimated: 60+ Story Points
Sprint Distribution:

- Sprint 1: User Registration, Event Schedule, SOS Button (~15 points)
- Sprint 2: Crowd Density Alerts (~15 points)
- Sprint 3: Interactive Map (~30 points)

Key Takeaways from Planning Poker for Maha-Khumb App

Encourages team discussion before committing to estimates.
Outliers help identify risks early.
Provides realistic effort estimation for better sprint planning.
Ensures consensus so all developers understand the workload.

Q2: Read Paper – Measuring Software Development Waste in OSS Projects - https://arxiv.org/pdf/2409.19107. Pick one measure from this paper and apply it on any open-source repository. Share results.

```
PS C:\Users\knave\OneDrive\Documents\Devops_Assignment\assignment1-bash-sandeep-006> python -u "c:\Users\knave\OneDrive\Documents\De
vops_Assignment\assignment1-bash-sandeep-006\assignment2.py"
Total Closed PRs: 100
Merged PRs: 86
Unmerged PRs: 14
PR Rejection Rate: 0.14
```

Implementation of PR Rejection Rate in DevOps Assignment

Paper Reference: *Measuring Software Development Waste in OSS Projects* (arXiv:2409.19107)

1. Selected Measure: PR Rejection Rate

The research paper highlights Pull Request (PR) Rejection Rate as a crucial metric for assessing software development efficiency in open-source projects.

Formula for PR Rejection Rate:

PR Rejection Rate=Unmerged Closed PRsTotal Closed PRsPR \ Rejection \ Rate = \frac{\text{Unmerged Closed PRs}}{\text{Total Closed PRs}}PR Rejection Rate=Total Closed PRsUnmerged Closed PRs

Where:

- **Unmerged Closed PRs:** Number of PRs that were closed but not merged.
- **Total Closed PRs:** Sum of merged and unmerged closed PRs.

A high PR rejection rate might indicate inefficiencies such as:
Misalignment between contributors and maintainers
Poor code quality
Redundant or unnecessary contributions

2. Application on DevOps Assignment

From the provided Python script execution, the computed PR statistics are:

- Total Closed PRs: 100
- Merged PRs: 86
- Unmerged PRs: 14

PR Rejection Rate Calculation:

PR Rejection Rate=14100=0.14PR \ Rejection \ Rate = \frac{14}{100} = 0.14PR Rejection Rate=10014=0.14

This 14% rejection rate suggests that out of every 100 closed PRs, 14 were rejected (not merged).

3. Insights & Observations

A low PR rejection rate (14%) generally indicates a healthy development process.
If the rejection rate were significantly higher, it could signal issues in contribution quality, review process, or maintainers' decision-making.
The team can further analyze rejection reasons (e.g., lack of tests, duplicate features, or outdated PRs) to optimize development efficiency.

Would you like further analysis or improvements in the script? 🚀

Interpretation of PR Rejection Rate in DevOps Assignment

Based on the computed PR statistics from your output:

- 86% of closed PRs were merged, showing a high acceptance rate.
- 14% of PRs were closed without merging, meaning some contributions were either rejected or abandoned.
- A PR rejection rate of 0.14 suggests that the project maintainers are actively reviewing and merging PRs, ensuring that most contributions meet the required standards.

3. Key Takeaways

High Acceptance Rate: 86% of PRs were merged, indicating an efficient development workflow.
Low Rejection Rate (14%): Most contributions were accepted, implying that PRs were generally well-aligned with project needs.
Reasons for Unmerged PRs: Some PRs might be rejected due to outdated code, conflicts, or not meeting quality standards.

Potential Next Steps

Analyze PR rejections: Check why certain PRs were not merged—were they duplicate contributions, lacking documentation, or conflicting with existing code?

Compare with other repositories: Evaluate the rejection rate across multiple projects to identify trends in contribution quality and review efficiency.

Optimize Contribution Guidelines: If needed, improve PR submission guidelines to minimize unnecessary rejections and enhance collaboration.