Neha Koshti

205224009

M.Tech Data Analytics

Devops Homework 1

Planning Poker – Agile Estimation Technique

Planning Poker is a consensus-based estimation technique used in Agile software development to estimate the effort required for tasks or user stories. It combines expert opinion, analogy, and disaggregation into an interactive process.

Instead of endless debates, team members use numbered cards to vote on how complex a task is. If the votes don't match, they discuss their reasoning, revote, and keep going until they reach an agreement. It's a mix of expert judgment, discussion, and a bit of gut feeling all wrapped in a simple game-like format.

Working:

- The Scrum Master runs the session.
- Each team member gets a set of cards with numbers like 1, 2, 3, 5, 8, 13, 21 (Fibonacci sequence). These numbers represent story points, a way to measure effort, complexity, risk.
- The Product Owner explains what needs to be built.
- The team picks a task or "user story" to estimate.
- The Product Owner explains what's needed and answers any questions.
- Each team member privately selects a card that represents their estimate.
- Everyone reveals their cards at the same time.
- If the estimates vary a lot, the team discusses the differences.
- The team revotes until they reach an agreement.

Developing a Mobile App for Maha-Kumbh

A 10-member development team has been asked to build a mobile app for Maha-Kumbh in 3 months. This app needs to:

- Users register and log in

- Provide real-time crowd navigation
- Show event schedules and send notifications
- Offer an SOS feature for emergencies
- Work offline with preloaded maps
- Be optimized for high traffic and security

The team will use Planning Poker to estimate the effort for each feature.

How the Team Uses Planning Poker

Consider one feature: ***Live Crowd Navigation***.

The app needs to track user locations and suggest less crowded routes in real-time.It requires:

- GPS tracking
- Real-time data processing
- Backend support for handling thousands of users
- An offline fallback for areas with bad network coverage

Step 1: Individual Estimates : Each developer picks a card privately and then reveals their estimate at the same time -

The votes are 5, 8, 13, 8, 21, 13, 8, 8, 13, 8

Step 2: Discussion : Since some developers voted 8, and others voted as high as 21, they discuss why-

- Some argue it's just about integrating Google Maps, so not too hard.
- Others point out that handling live traffic, network failures, and load balancing makes it way more complex.

Step 3: Revote and Consensus : After discussion, they revote, and most agree on 13 story points.

Estimating the Remaining Features

Feature	Initial Estimates	Final Estimate (Story Points)
User Registration & Authentication	5, 8, 8, 5, 5, 8, 8, 5, 8, 5	8

Live Crowd Navigation	5, 8, 13, 8, 21, 13, 8, 8, 13, 8	13
Event Scheduling & Notifications	3, 5, 5, 8, 5, 5, 3, 5, 5, 5	5
Emergency Help & Alerts	8, 13, 8, 8, 13, 8, 8, 8, 13, 8	8
Offline Maps & Guide	13, 21, 13, 13, 21, 13, 13, 21, 21, 13	13
Performance Optimization & Security	8, 13, 8, 8, 8, 13, 8, 8, 8, 8	8

Final Sprint Planning for Maha-Kumbh App Development

Total Story Points Estimated = 8 + 13 + 5 + 8 + 13 + 8 = 55 Story Points

Project Duration = 3 Months (~12 Weeks, assuming 2-week sprints)

Assuming the team can complete 10-15 story points per sprint, the project timeline looks feasible.

Planning Poker is a super useful tool for Agile teams it makes estimating effort quick, fair, and accurate. By using it, the Maha-Kumbh app team can now plan their work better, avoid surprises, and stay on track to deliver a great product on time.

Analysis of Pull Request Rejection Rate (PRR)

Pull Request Rejection Rate (PRR) : a key metric that indicates what percentage of submitted PRs were not merged into the project.

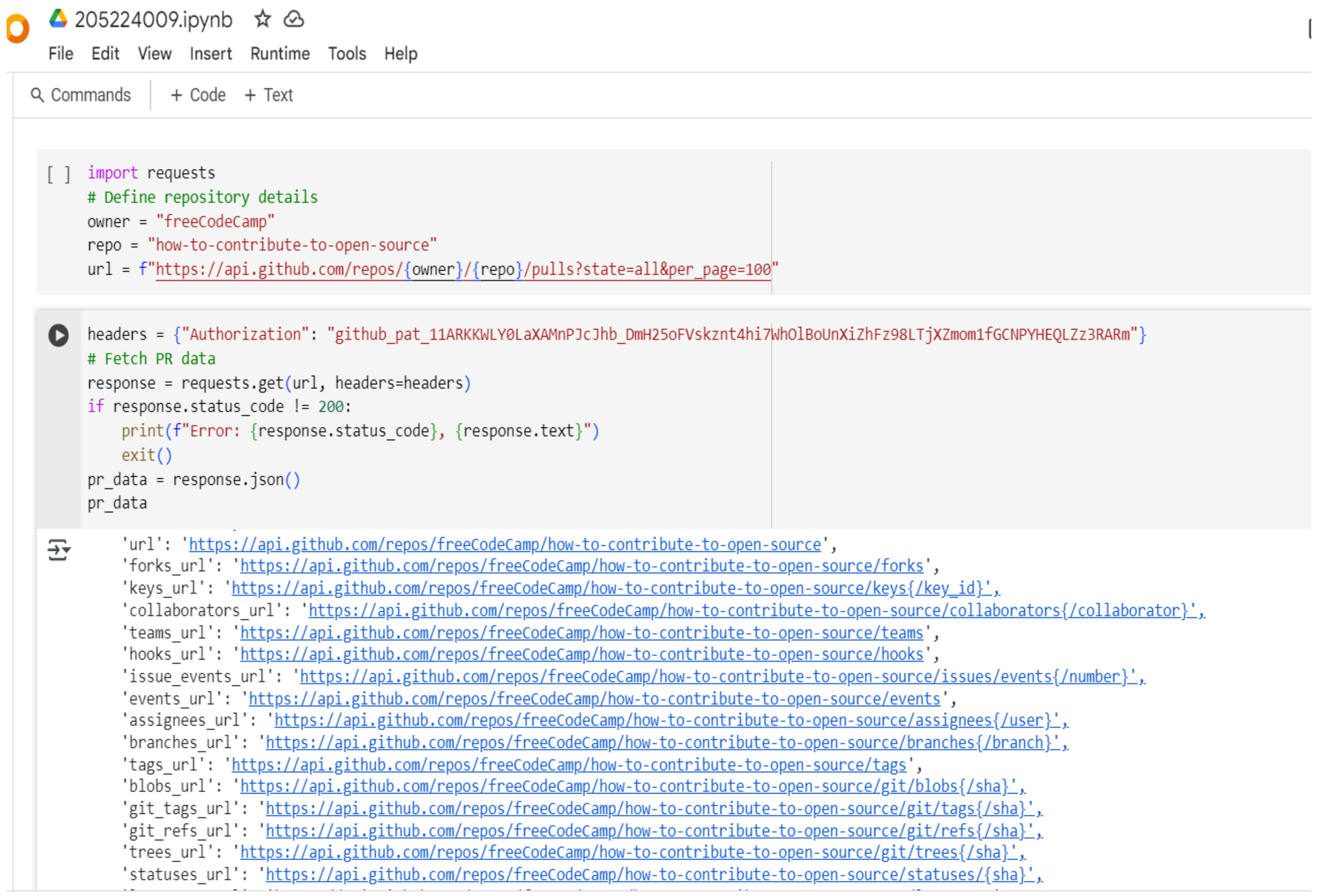
To calculate PRR, I used GitHub's REST API to fetch all PRs from the repository. The PRR is computed as:

PRR=(Unmerged PRs / Total PRs)×100

- Unmerged PRs: Pull requests that were either closed without merging or are still open.
- Total PRs: The sum of merged and unmerged PRs.

freeCodeCamp GitHub Repository:
https://github.com/freeCodeCamp/how-to-contribute-to-open-source

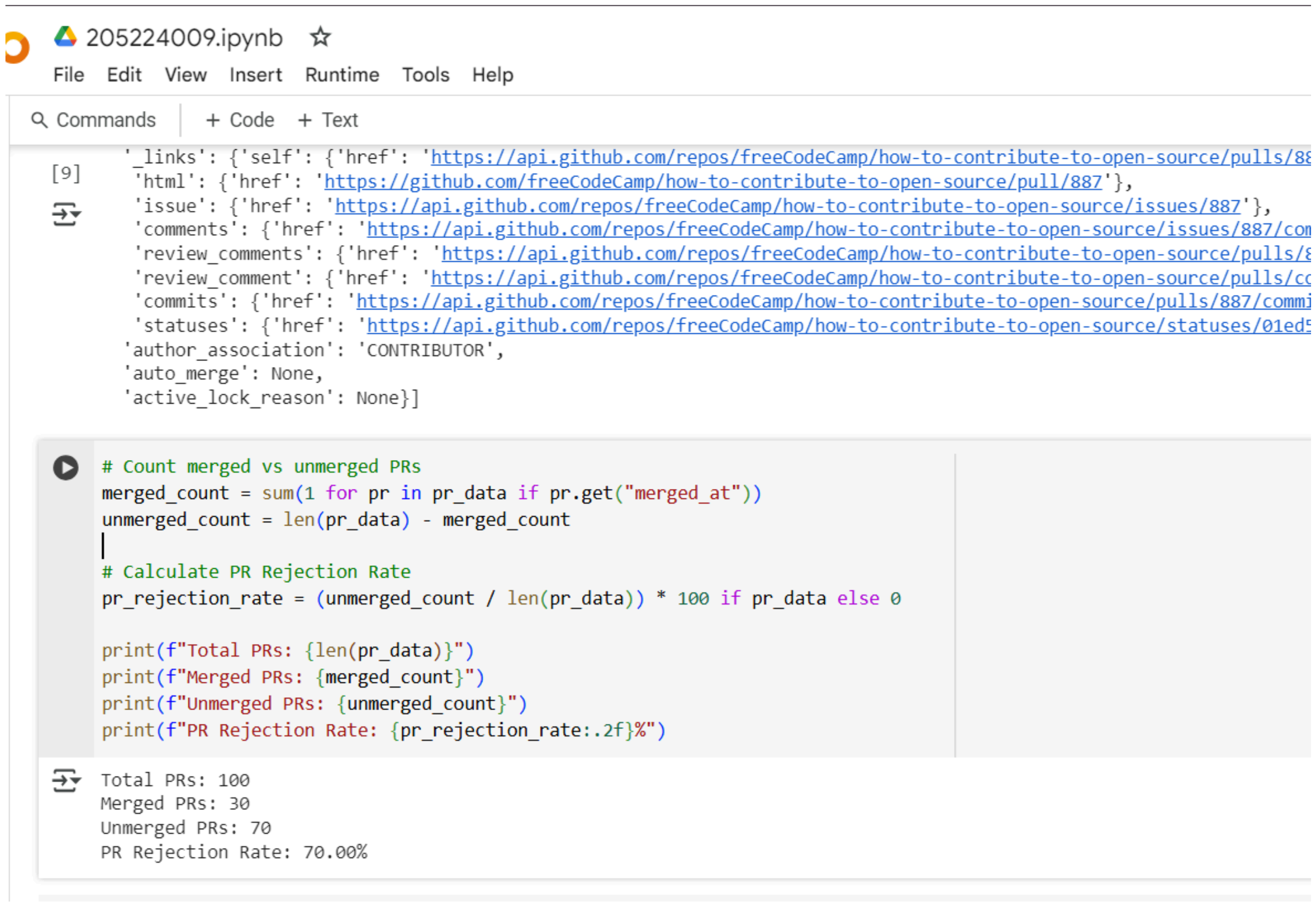
I wrote a Python script to extract and process thedata.



```
[ ] import requests
    # Define repository details
    owner = "freeCodeCamp"
    repo = "how-to-contribute-to-open-source"
    url = f"https://api.github.com/repos/{owner}/{repo}/pulls?state=all&per_page=100"
```

```
headers = {"Authorization": "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"}
# Fetch PR data
response = requests.get(url, headers=headers)
if response.status_code != 200:
    print(f"Error: {response.status_code}, {response.text}")
    exit()
pr_data = response.json()
pr_data
```

```
    'url': 'https://api.github.com/repos/freeCodeCamp/how-to-contribute-to-open-source',
    'forks_url': 'https://api.github.com/repos/freeCodeCamp/how-to-contribute-to-open-source/forks',
    'keys_url': 'https://api.github.com/repos/freeCodeCamp/how-to-contribute-to-open-source/keys{/key_id}',
    'collaborators_url': 'https://api.github.com/repos/freeCodeCamp/how-to-contribute-to-open-source/collaborators{/collaborator}',
    'teams_url': 'https://api.github.com/repos/freeCodeCamp/how-to-contribute-to-open-source/teams',
    'hooks_url': 'https://api.github.com/repos/freeCodeCamp/how-to-contribute-to-open-source/hooks',
    'issue_events_url': 'https://api.github.com/repos/freeCodeCamp/how-to-contribute-to-open-source/issues/events{/number}',
    'events_url': 'https://api.github.com/repos/freeCodeCamp/how-to-contribute-to-open-source/events',
    'assignees_url': 'https://api.github.com/repos/freeCodeCamp/how-to-contribute-to-open-source/assignees{/user}',
    'branches_url': 'https://api.github.com/repos/freeCodeCamp/how-to-contribute-to-open-source/branches{/branch}',
    'tags_url': 'https://api.github.com/repos/freeCodeCamp/how-to-contribute-to-open-source/tags',
    'blobs_url': 'https://api.github.com/repos/freeCodeCamp/how-to-contribute-to-open-source/git/blobs{/sha}',
    'git_tags_url': 'https://api.github.com/repos/freeCodeCamp/how-to-contribute-to-open-source/git/tags{/sha}',
    'git_refs_url': 'https://api.github.com/repos/freeCodeCamp/how-to-contribute-to-open-source/git/refs{/sha}',
    'trees_url': 'https://api.github.com/repos/freeCodeCamp/how-to-contribute-to-open-source/git/trees{/sha}',
    'statuses_url': 'https://api.github.com/repos/freeCodeCamp/how-to-contribute-to-open-source/statuses/{sha}',
```



```
 '_links': {'self': {'href': 'https://api.github.com/repos/freeCodeCamp/how-to-contribute-to-open-source/pulls/88
  'html': {'href': 'https://github.com/freeCodeCamp/how-to-contribute-to-open-source/pull/887'},
  'issue': {'href': 'https://api.github.com/repos/freeCodeCamp/how-to-contribute-to-open-source/issues/887'},
  'comments': {'href': 'https://api.github.com/repos/freeCodeCamp/how-to-contribute-to-open-source/issues/887/com
  'review_comments': {'href': 'https://api.github.com/repos/freeCodeCamp/how-to-contribute-to-open-source/pulls/8
  'review_comment': {'href': 'https://api.github.com/repos/freeCodeCamp/how-to-contribute-to-open-source/pulls/co
  'commits': {'href': 'https://api.github.com/repos/freeCodeCamp/how-to-contribute-to-open-source/pulls/887/commi
  'statuses': {'href': 'https://api.github.com/repos/freeCodeCamp/how-to-contribute-to-open-source/statuses/01ed5
 'author_association': 'CONTRIBUTOR',
 'auto_merge': None,
 'active_lock_reason': None}]
```

```python
# Count merged vs unmerged PRs
merged_count = sum(1 for pr in pr_data if pr.get("merged_at"))
unmerged_count = len(pr_data) - merged_count

# Calculate PR Rejection Rate
pr_rejection_rate = (unmerged_count / len(pr_data)) * 100 if pr_data else 0

print(f"Total PRs: {len(pr_data)}")
print(f"Merged PRs: {merged_count}")
print(f"Unmerged PRs: {unmerged_count}")
print(f"PR Rejection Rate: {pr_rejection_rate:.2f}%")
```

```
Total PRs: 100
Merged PRs: 30
Unmerged PRs: 70
PR Rejection Rate: 70.00%
```

The script fetched the latest 100 PRs. The results are :

- Total PRs analyzed: 100
- Merged PRs: 30
- Unmerged PRs: 70
- Pull Request Rejection Rate (PRR): 70.00%

Findings and Observations

If 70% of PRs get rejected, it means 7 out of 10 don't make it in. That's a lot! It could be because of code quality, duplicate work, or just not following guidelines.

A high rejection rate can sometimes discourage new contributors. But in a well-maintained project like freeCodeCamp, some rejections are normal, it helps keep the codebase clean. Still, if rejection rates are this high