Q1

Estimating a Mobile App for Maha-kumbh

A team is assigned the task of developing a mobile app for the **Maha-Kumbh** event, with features such as:

a. Event schedules

b. Live updates and notifications

c. Crowd management insights

d. Emergency alerts

e. Location-based services

The team consists of 10 members, including developers, designers, testers, and a Scrum Master. The deadline is 3 months.

Conducting Planning Poker:

USER STORY 1: *"As a visitor, I want to view the event schedule and receive notifications for upcoming events so that I don't miss important ceremonies."*

Planning Poker Estimates:

- Developer 1: 5
- Developer 2: 3
- Tester: 5
- Designer: 3
- Scrum Master: 3

Discussion:

- Developer 1 believes integrating a real-time notification system may be tricky.
- The designer argues that most of the work is UI-based and should be simple.
- The Scrum Master suggests reusing an existing notification framework to reduce effort.

- **Consensus: 3 story points.** The team agrees on assigning 3 story points to this user story.

USER STORY 2: *"As a visitor, I should be able to view real-time crowd density updates at different ghats to plan my visit accordingly."*

Planning Poker Estimates:

- Developer 1: 8
- Developer 2: 5
- Designer: 8

- Tester: 5
- Scrum Master: 3
- Other team members: Mixed values between 3 and 8

Discussion:

- Developer 1 argues that integrating live crowd data from sensors and APIs will take time.
- Tester believes it's complex because of potential data inconsistencies.
- Scrum Master suggests breaking it into smaller parts: API integration, UI, and testing separately.
- **Consensus: 8 story points.** The complexity of handling real-time data and API limitations make this a larger effort.

USER STORY 3: *"As a visitor, I want to see a map of the Kumbh Mela grounds with key locations like restrooms, medical tents, and water stations so that I can find essential services easily."*

Planning Poker Estimates:

- Developer 1: 5
- Developer 2: 3
- Tester: 5
- Designer: 3
- Scrum Master: 5

Discussion:

- Developer 1 suggests using Google Maps API for a faster implementation.
- Tester mentions that offline support might add extra effort.
- Designer points out that icons and user-friendly UI will be critical.
- **Consensus: 5 story points.** Using Google Maps API simplifies implementation but still requires effort for customization.

USER STORY 4: *"As an event organizer, I want to send emergency alerts to all users in case of a stampede or weather issue so that they can take necessary precautions."*

Planning Poker Estimates:

- Developer 1: 8
- Developer 2: 5
- Tester: 8
- Designer: 5
- Scrum Master: 5

Discussion:

- Developers highlight that implementing push notifications + SMS alerts needs backend work.

- Tester points out challenges in ensuring timely delivery of alerts under heavy load.
- The Scrum Master suggests using cloud-based messaging services
- **Consensus: 8 story points.** Critical feature requiring robust backend and real-time performance tuning.

Q2. **PR Rejection Rate**

Repository - https://github.com/scikit-learn/scikit-learn

# get data using api calls

$prData = Invoke-RestMethod -Uri [https://api.github.com/repos/ scikit-learn/scikit-learn/pulls?state=all](https://api.github.com/repos/scikit-learn/scikit-learn/pulls?state=all)

#Count of rejected pull requests

$rejectedPRs = ($prData | Where-Object { $_.state -eq "closed" -and -not $_.merged_at }).Count

Output : 0

#total pull requests

$totalPRs = $prData.Count

Output : 30

#rejection rate = rejectedPRs / totalPRs

$rejectionRate = [math]::Round(($rejectedPRs / $totalPRs) * 100, 2)

Output: 0

PRR = 0