Ans. 1:

Planning Poker is an Agile technique that is used by the development teams in order to estimate the efforts that will be required to complete a given task. This technique follows certain steps:

Steps:

1. The Product Manager will present the task to his team.
2. The team will have discussion between them regarding the task in which they are free to ask questions and can clarify any doubts that they will have.
3. There will be a deck of cards with numbers written on it where numbers represent the story points that an individual thinks he will require(story points are directly proportional to effort). Each individual has to select a card from the deck privately.
4. Then each individual has to open their cards.
5. If there is a large difference in the minimum and the maximum value then there will be further discussions among the team members regarding why they have chosen the following story points.
6. The process will continue till the consensus is reached.

Applying the above steps to develop a mobile app for Maha-Kumbh in 3 months.

Let's assume the app has features which are event schedules, maps, crowd alerts, and ticket booking. The team uses Planning Poker during sprint planning to estimate the effort for user stories in their product backlog.

Team Members Composition:

The team consists of 10 people with the following roles and count that are:
1 Product Manager
1 Scrum Master
1 DevOps Engineer
3 Developers
2 UX Designers
2 QA Testers

Planning Poker Session

1. **Setup:**
    - The Scrum Master schedules the Scrum Meeting for the Team.
    - In the meeting the Product Managers share the Product that is to be developed (Maha-Kumbh Mobile App) with the team.

- Each team member has a deck of Planning Poker cards where the value represents the story points.

2. **Discussion:**
    - The group will sit collectively and discuss the steps they'll follow and the doubts they are having.
    - Whatever doubts the team members are having they discuss with each other until they get the entire clarification of the task that they have to complete.

3. **First Estimation Round:**
    - Team members privately select cards based on their understanding of effort.
    - Let's assume that the cards chosen were:
        - Developer 1: 5
        - Developer 2: 8
        - Developer 3: 13
        - UX Designer 1: 5
        - UX Designer 2: 3
        - QA Tester 1: 8
        - QA Tester 2: 13
        - DevOps Engineer: 8
        - Scrum Master (observing, not voting): N/A

4. **Discussion of Variance:**
    - UX Designer 2 gave the lowest estimate i.e. 3 and the QA Tester 2 and Developer 3 gave the highest estimate i.e. 13.
    - They will share the thoughts of why they have chosen these story points.
    - By sharing the thoughts others will figure out whether they have missed something while estimating the story points or whether the person sharing the thoughts has missed something or considered something extra.

5. **Second Estimation Round:**
    - After discussion, the team re-votes:
        - Developer 1: 8
        - Developer 2: 8
        - Developer 3: 8
        - UX Designer 1: 5
        - UX Designer 2: 5
        - QA Tester 1: 8
        - QA Tester 2: 8
        - DevOps Engineer: 8

- Most votes cluster around 8, with some 5s. The team discusses briefly, and the UX Designers agree that 8 accounts for integration effort they hadn't considered.

6. **Consensus:**
   - The team agrees on **8 story points** for this user story, reflecting moderate complexity due to API integration, UI design, and testing.

The team repeats this process for other user stories as well e.g., "View event schedule," "Receive crowd alerts" until there's no consensus.

Ans 2:

Pull Request Rejection Rate (PRR Rate):

Pull Request Rejection Rate (PRR) is a particular measure applied to determine waste in software development by examining the quality and efficiency of code contributions made through pull requests (PRs) in version control tools such as GitHub, GitLab etc. It calculates the ratio of rejected PRs (i.e., not merged) based on factors like low-quality code, test failures, incomplete requirements, or deviations from project standards. High PRR may signal waste in the shape of rework, defects, or miscommunication and is a useful lens to use in recognizing inefficiencies within the development process.

In software development, a pull request is a proposal for a change in the codebase. When a PR is declined, efforts like coding effort, time taken for reviewing the code, testing efforts etc. can be lost partially or entirely, and the code must be rewritten. PRR has a direct correlation with wastes like defects (defective code), overprocessing (too many revisions), and waiting (wasted time in loops of feedback). PRR tracking allows teams to detect process failure and improve collaboration, code quality, and delivery speed.

$$PRR = ((Number\ of\ Rejected\ PR's) / (Total\ Number\ of\ PR's\ submitted)) * 100$$

Steps to Measure PRR and Identify Waste:

1. Collect Data:
   a. Use version control system metrics (e.g., GitHub Insights, GitLab Metrics) or scripts to track PR success.
   b. Mark PRs as "Merged," "Rejected," or "Abandoned."

2. Calculate PRR:
   a. For a time or sprint, count total PRs and rejected PRs, and then use the formula mentioned above.

3. Examine Rejection Reasons:
    a. Review rejected PRs to find out why they failed (e.g., bugs, scope creep, poor documentation) and also look at PR comments, CI/CD logs, team feedback.

4. Correlate to Waste
    a. Try and figure out which waste can we map it to.

5. Take Action:
    a. Adjust processes (e.g., more code reviews, better requirements) to decrease PRR and waste.

Repo on which the measure was done: django/django

Result:

```
PS C:\Users\Acer\Desktop> whoami
laptop-gf3tvkpa\acer
PS C:\Users\Acer\Desktop> python DevOps_Homework.py
Total PRs: 338
Rejected PRs: 86
Pull Request Rejection Rate: 25.44%
```