0.a Come up with a team name for your group.

BHADS Team

0.b Please list the names and PIDs of the team members who are present today (or knowingly absent)

Aaron Lambert (aaronlambert)

Sonny McMaster (sonnymc)

Harrison Leverone (lharry)

Danny Spatz (spatzd) [not here but let us know beforehand]

Brett Noneman (brettn) [Absent for medical reasons]

0.c Provide your preliminary project idea (or set of ideas). This is not a commitment to a project.

- Productivity timer bot to help software engineers know when to take breaks.
- Programming Joke Discord bot to improve morale of software engineers.
- Rubber duck debugging discord bot.

MAYBE A COMBINATION OF ALL 3?

Using the approved idea for your group's course project, complete the following activities related to requirements analysis.

1. Provide an example of five hypothetical non-functional requirements for this system. Be sure to include the specific type of requirement discussed in class, with each requirement coming from a unique category.

1. Help documentation would be well written : usability
2. The bot shouldn’t fail to respond less than 99% of the time: reliability
3. Response time would be immediate : performance
4. Can be used in multiple projects/servers/situations : supportability (adaptability
5. Must use in discord : implementation/constraints

2. Provide an example of five hypothetical functional requirements for this system.

1. Allow user input for time of work/break and track time elapsed
2. Ping user upon timer ending/starting
3. Should have at least 20 distinct jokes that are randomly chosen
4. Provide a generic response to help the user think deeper when debugging
5. Should be able to do all three (timer, jokes, rubber duck debug) in one bot session.

3. Think of a specific task required to complete each of the functional requirements and non-functional requirements mentioned above. Estimate the amount of effort needed to complete this task using function points (i.e., using the values here). Briefly explain your answer.

1. Allow user input for time of work/break and track time elapsed
    a. Set up a timer that is accessible for time checks (1 point)
2. Ping user upon timer ending/starting
    a. Create a method that pings a user and ensure it pings correct user (2 point)
3. Should have at least 20 distinct jokes that are randomly chosen
    a. Set up random joke selector that cycles through the jokes (2 points)
4. Provide a generic response to help the user think deeper when debugging
    a. Come up with helpful generic responses that will be used (3 points)
5. Should be able to do all three (timer, jokes, rubber duck debug) in one bot session.
    a. Ensure that it is possible to use each in the one bot (8 points)

6. Help documentation would be well written : usability
    a. Proofread each others' additions to the help documentation until perfect (3 points)
7. The bot shouldn't fail to respond less than 90% of the time: reliability
    a. Debug any errors encountered to decrease failure rate (2 points)

8. Response time would be almost immediate : performance
    a. Remove any hurdles that could slow the bot’s performance (2 points)
9. Can be used in multiple projects/servers/situations : supportability (adaptability)
    a. Write the bot non-specific so it can be used in many cases (3 points)
10. Must use in discord : implementation/constraints
    a. Hardcode the bot so that it is discord only (1 point)

4. Write three user stories from the perspective of at least two different actors. Provide the acceptance criteria for these stories.

Actor: Software Engineer (User)

Story 1:

As a Software Engineer, I want to be reminded to take breaks, so I can stay productive and prevent burnout.

Acceptance Criteria:

1. The user can set the timer's duration for both work and breaks.
2. The bot sends a notification to the user when it's time to take a break or resume work.
3. The bot provides a clear indication that the timer has started or stopped.

Story 2:

As a Software Engineer, when I'm stuck on a problem, I want the rubber duck debugging feature to help me think about the problem differently.

Acceptance Criteria:

1. The user can activate the rubber duck debugging mode.
2. The bot provides a generic response prompting the user to think or approach the problem in a certain way.
3. The bot doesn't provide direct solutions but instead guides the thinking process.

Actor: Discord Server Admin

Story 3:

As a Discord Server Admin, I want to be able to set the bot permissions so that it doesn't disrupt the natural flow of discussions.

Acceptance Criteria:

1. The server admin can easily manage the bot's permissions.
2. The bot respects the permissions set by the server admin.
3. The bot only interacts/responds when called upon by a server member.

5. Provide two examples of risk that could potentially impact this project. Explain how you would mitigate these risks if you were implementing your project as a software system.

Risk 1: Overuse of the bot can lead to spam in the server, filling up chat logs and disrupting conversations.

Mitigation:

1. Limit the bot's responses if called repeatedly in a short span of time.
2. Allow server admins to configure rate limits on bot interactions.

Risk 2: The bot's timer or notifications might malfunction, leading to inaccurate reminders.

Mitigation:

1. Regularly test the bot to ensure the timer feature is functioning correctly.
2. Allow users to report issues and provide updates to fix known bugs.

6. Describe which process your team would use for requirements elicitation from clients or customers, and explain why.

Our team would use Interviews and Surveys as the primary methods for requirements elicitation.

Interviews: They provide an in-depth understanding of the user's needs, allowing us to capture detailed requirements. By discussing directly with potential users, we can clarify any ambiguities and ensure that our software will truly meet their needs.

Surveys: They enable us to reach a wider audience and gather a large amount of data in a short amount of time. Surveys can help identify the most common needs and desires among potential users, helping us prioritize features.

By combining these two methods, we can ensure that our software is both tailored to individual needs and generally applicable to a broader audience.