Productivity Bot Final Presentation

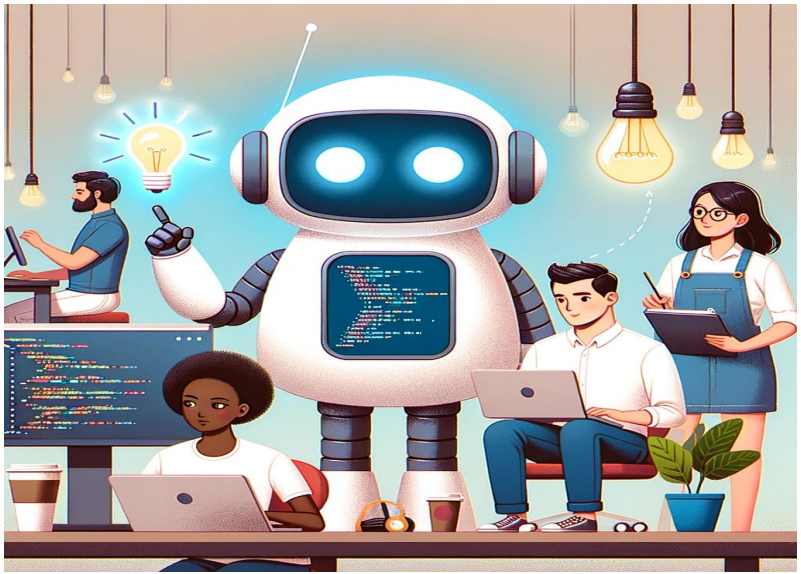
By Team BHADS:
Brett Noneman, Harry Leverone, Aaron Lambert, Danny Spatz, Sonny McMaster

Problem statement

How can we create a bot for Software Engineers that:

- **Boosts productivity** with features that:
  - Help the engineer stay focused
  - In a good mental state while working

"75% of a developer's time is spent on debugging (1500 hours a year!)"
-Coralogix study



"Friendly Robot Assistant with Software Engineers," created by OpenAI's DALL-E.

Explanation of Solution

We created a discord bot that has:

- Jokes for boosting morale
  - Gives bot personality and more interactivity
- Rubber duck debugging
  - Helps programmer solve issues
- Manages breaks
  - Similar to another tool, the Pomodoro timer

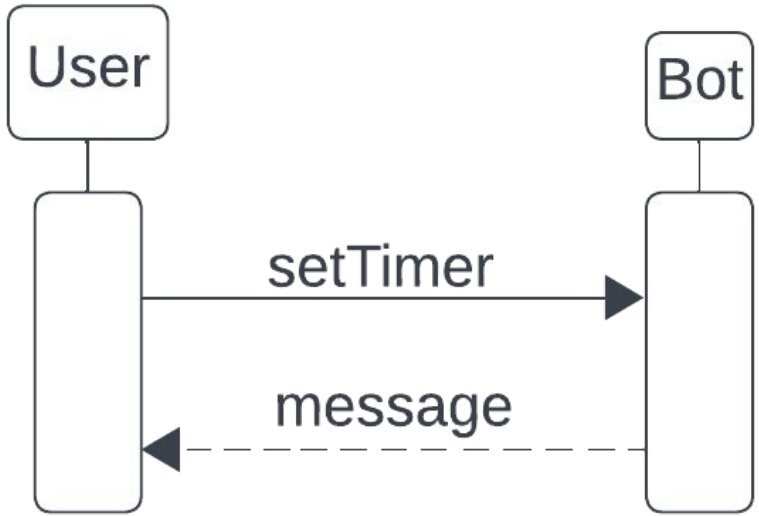
"Putting the effort to teach your rubber duck can help you recognize whether you actually understand a topic".
-University of Business Innovation and Sustainability (Switzerland)

"People who practice the Pomodoro Technique say it improves their concentration and that they feel less drained when the workday is over"
-builtin.com

Updated Use Cases

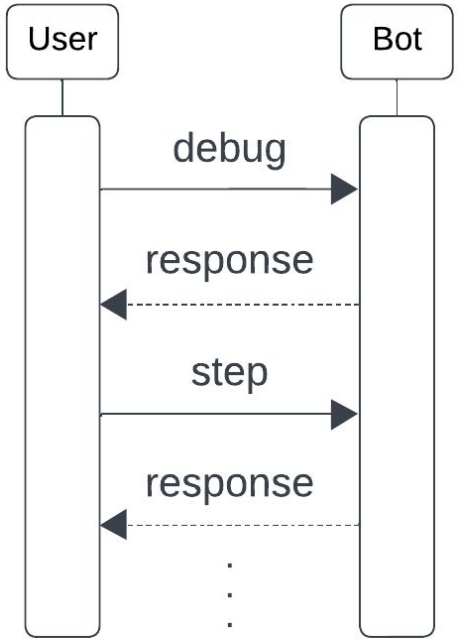
Use Case 1 - Setting a Timer

1. **Preconditions**
   User must have access to discord chat.
2. **Main Flow**
   User requests to set a timer [S1]. The bot confirms the request [S2]. The bot notifies the user when the timer completes [S3].
3. **Subflows**
   [S1] User provides command !timer
   [S2] Bot responds with confirmation message
   [S3] Discord notification is sent to user on completion
4. **Alternative Flows**
   [E1] Invalid command



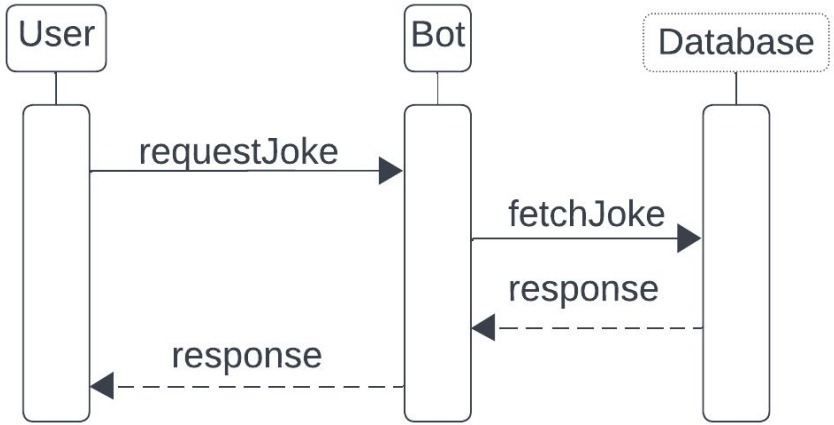
Use Case 2 - Rubber Duck Debugging

1. **Preconditions**
   User must have access to discord chat. User needs help debugging.
2. **Main Flow**
   User activates debug mode [S1]. User and chat bot walk through code step by step until issue is found [S2].
3. **Subflows**
   [S1] User enters command !debug
   [S2] User repeatedly enters command !step
4. **Alternative Flows**
   [E2] Invalid command



Use Case 3 - Jokes System

1. **Preconditions**
   User must have access to discord chat.
2. **Main Flow**
   User requests a joke from the discord bot [S1]. Discord bot responds with a joke [S2].
3. **Subflows**
   [S1] User provides command !joke
   [S2] Discord bot pulls joke from database and then responds through discord chat
4. **Alternative Flows**
   [E1] Invalid command



Demo

Limitations

- Command Scope
  - *“Invalid Command”* issue
- Discord API dependency
  - Changes with discord -> potential changes with bot
- Finite responses
  - ex: limited pre-programmed jokes for SE

Future Plans

- Timer functions
    - Changing alarm sound, pause feature
- Adding more jokes
    - Wider variety to shuffle through
- Rubber duck debugging
    - AI integration for better feedback

Processes Used | Requirements Elicitation Process

- Interviews
  - **Purpose:** Gain in-depth understanding of user needs for our Discord bot project.
  - **Process:** Direct discussions with potential users to capture detailed requirements.
  - **Advantage:** Clarifies ambiguities, ensuring our software (Productivity Timer, Joke Bot, Rubber Duck Debugging) meets specific user needs.
- Surveys
  - **Purpose:** Reach a broader audience to understand common needs.
  - **Process:** Collect large amounts of data efficiently to identify prevalent user requirements.
  - **Advantage:** Helps in prioritizing features for our Discord bot project.

Processes Used | High-Level Design: Event-Driven Architecture

- **Responsiveness to Events:** EDA is crucial for the Discord bot to respond to user commands efficiently.
- **Decoupling of Components:** Separates event producers (users) from event consumers (bot logic), enhancing flexibility.
- **Scalability and Flexibility:** Facilitates adding new commands and functionalities easily.
- **Asynchronous Processing:** Maintains bot responsiveness in high-latency environments.
- **Maintainability and Testability:** Modular nature allows for isolated testing and maintenance of distinct functionalities.
- **Application:** Distinct event handlers for commands like !joke, !debug, and !step, integrated with Discord API.

Processes Used | Low-Level Design/Prototyping

- **Structural Design Pattern:**
  - Utilizing an Adapter class to streamline interactions with the Discord API.
  - This approach abstracts relevant functionality from the Discord API, simplifying the addition of new bot features.
- **Design Sketch and Wireframes:**
  - Created three wireframes for key functionalities: Joke System, Rubber Duck Debugging, and Break Period Timer.
  - Each wireframe demonstrates user interaction with the bot commands like !joke, !debug, and timer setup.
- **Prototyping Process:**
  - Development of initial prototypes to test individual functionalities.
  - Iterative improvement based on user feedback and testing results.
- **Testing and Feedback:**
  - Regular testing phases to ensure functionality and responsiveness.
  - Incorporating feedback from test users to refine the bot features and user experience.

What we learned

- **Technical Skills and Design Patterns:**
  - Enhanced understanding of Event-Driven Architecture (EDA) and Adapter design patterns.
  - Learned the value of modular design for future scalability and easier maintenance.
- **Requirements Elicitation and User Feedback:**
  - Recognized the importance of combining interviews and surveys for comprehensive requirements gathering.
  - Valued iterative feedback for refining and improving product features.
- **Collaboration and Team Dynamics:**
  - Developed stronger team collaboration skills, including effective communication and remote coordination.
  - Overcame challenges in merging diverse viewpoints and skills to achieve project goals.
- **Problem-Solving and Creativity:**
  - Improved problem-solving abilities through innovative solutions to technical challenges.
  - Fostered creativity by integrating diverse functionalities (joke system, debugging, timer) into one bot.
- **Project Management Skills:**
  - Enhanced project planning and management abilities, including task prioritization and deadline management.
  - Learned the significance of balancing resource allocation with project demands.