Slack Bot Implementation

Date: February 2, 2025
Prepared by: Ayush Sheta
Team: TEAM-18

1. Overview

This document provides an in-depth analysis of two different implementations of a Slack bot:

1. **Command Bot –** Uses Slack slash commands (`/command`) to trigger bot actions.
2. **Chat Bot** – Listens to user messages in Slack channels and responds accordingly.

Both implementations utilize **Node.js**, **Express**, and **Slack Bolt SDK**. This documentation covers the preparation, development process, differences between the two approaches, challenges faced, and potential improvements.

The implementation codes are available for download as a ZIP file via the GitHub repository of **TEAM-18DASS2025**.

2. Preparation & Resources

Technologies Used:

- **Node.js**: JavaScript runtime environment for backend development.
- **Express**: Lightweight web framework for handling HTTP requests.
- **Slack Bolt SDK**: Official Slack framework for bot interactions.
- **dotenv**: Library for managing environment variables.
- **body-parser**: Middleware for handling request bodies.
- **crypto**: Node.js module for generating unique session IDs.

Resources Referred:

- Slack API Documentation
- Slack Bolt Framework
- Express.js Official Documentation
- Node.js Official Docs
- Various tutorials and GitHub repositories on Slack bot development.
- Debug and syntax support from chatgpt 4o.

3. Two Implementation Approaches

A. Command Bot Implementation

How It Works:

- Users interact with the bot by typing **slash commands** (e.g., `/searchcoach`, `/addgoal`).
- Each command is registered in Slack and handled by an **Express API route**.
- The bot processes the request and responds with structured messages using Slack's **Block Kit UI**.

Example Commands Implemented:

- `/coach` – Retrieves available coaches based on expertise.
- `/addgoal` – Creates a user-specific goal.
- `/addsession` – Schedules coaching sessions.
- `/listsessions` – Displays upcoming sessions.
- `/deletegoal` – Removes a specified goal.

Key Features:

- **Explicit interaction** – Users must know and type commands.
- **Fast processing** – API-based response with structured replies.
- **Security control** – Commands can be restricted to specific users.

B. Chat Bot Implementation

How It Works:

- The bot listens to **Slack messages** using event subscriptions.
- When a message matches a keyword (e.g., "search coach"), the bot responds.
- The bot uses **natural conversation flow** rather than structured commands.

Example Messages Handled:

- "search coach" – Returns a list of available coaches.
- "create action" – Creates an action task for the user.
- "create goal" – Registers a goal for tracking.
- "schedule session" – Schedules a coaching session.
- "help" – Provides guidance on how to interact with the bot.

Key Features:

- **More intuitive** – Users can phrase requests naturally.
- **Dynamic responses** – Bot reacts to general conversations.
- **Context-aware** – Potential for future AI enhancements.

4. Comparison: Command Bot vs. Chat Bot

Feature	Command Bot	Chat Bot
Trigger Mechanism	Slash commands (`/command`)	Listens to chat messages
Ease of Use	Requires knowledge of commands	More user-friendly, natural language
Response Format	Predefined, structured responses	More flexible, conversational replies
Implementation	API-based via Express routes	Event-based using Slack message listeners

Best Use Case	When clear actions are needed (e.g., task creation)	When interaction should be conversational

5. Challenges Faced & Solutions

Challenge	Solution
Limited knowledge of Slack API	Referred to official documentation and tutorials
Debugging issues in bot responses	Used extensive logging and tested responses in Slack sandbox
Formatting messages properly	Learned and applied Slack Block Kit effectively
Handling multiple user inputs	Implemented structured validation and error handling

6. Future Improvements(on Client feedback)

Enhancements:

- Add **interactive features** like dropdowns, buttons, and modals.
- Implement **multi-step interactions** for better user engagement.
- Improve UI with **formatted cards and structured responses**.
- Store data in a **database** instead of in-memory arrays.
- Add **authentication and permissions** for secure command usage.
- Implement **AI-based recommendations** for better coach matching.

7. Conclusion

Both implementations of the Slack bot offer unique advantages. The **Command Bot** is best suited for structured interactions requiring predefined actions, while the **Chat Bot** provides a more natural, conversational experience. Depending on user needs, either approach can be extended with additional features for better usability and engagement. Future improvements can enhance both implementations by incorporating AI, databases, and more interactive UI elements.