Tools For Teams

Declan Baker, Joseph Cote, Owen Davidson, Zihan Pan
Department of Computer Science, University of New Hampshire, Durham, NH 03824

Introduction

UNH Enterprise Technology & Services (ET&S) has raised concerns regarding internet security and compliance risks associated with using Discord as the primary communication platform within the Computer Science Department. To address these concerns, the department is transitioning to Microsoft Teams as its official communication platform. Our team is responsible for migrating the existing Discord bots' functionality to Microsoft Teams while expanding their capabilities to better support the Programming Assistance Center (PAC).

Success Criteria:

- Develop a Microsoft Teams bot that preserves core Discord bot functionality
- Expand functionality to better support the Programming Assistance Center (PAC)
- Ensure compliance with university IT security policies

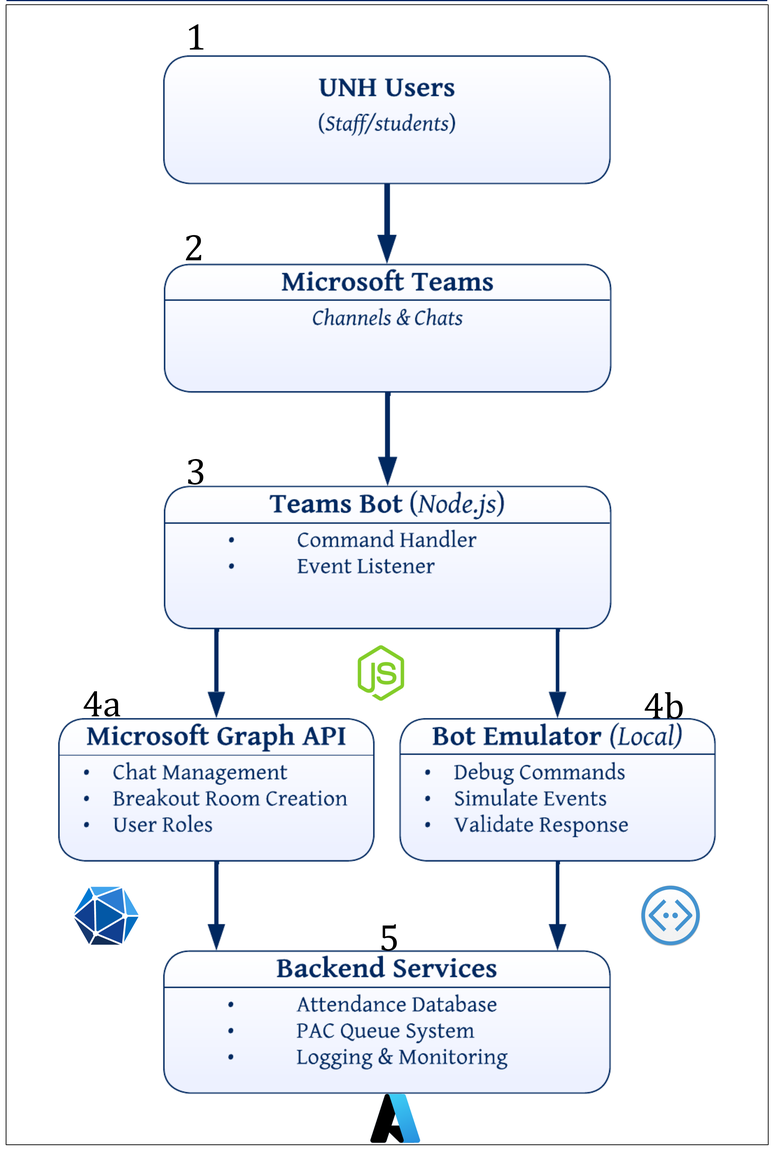
Requirements

- Functional
  - Students and faculty must be able to create and breakout rooms
  - Instructors must be able to run attendance polls
  - Students must be able to add themselves to PAC queue
- Nonfunctional
  - Bot must have minimal downtime
  - Bot must be easy enough to use to justify its use

Bot Commands

- **hello:** triggers a "Hello World!" response from the bot for testing purposes
- **poll:** start/stops an attendance action. User can click the interactive button to mark themself present. Stopping the attendance will automatically generate a text file of present students.
- **breakout**: create/close a breakout room with a list of users. Closing the breakout room will automatically remove all participants.
- **queue**: add a student to the PAC queue. PAC consultants can use assist to remove a student from the queue
- **availability**: a PAC consultant can mark themselves available or not, and students can check the availability of all PAC consultants
- E.G., "@TeamsChatBot breakout @Zihan Pan @Joseph Cote"

Architecture Diagram I



Architecture Diagram II

1. **UNH Users:** Students or staff type a command mentioning @TeamsChatBot in a Teams channel or chat.
2. **Microsoft Teams:** Teams captures the message and packages it into an activity payload, routing it through the Bot Framework Service to the bot.
3. **Teams Bot:** The command handler parses the command and dispatches it, while the event listener watches for passive events like members joining.
4a. **Microsoft Graph API:** The bot calls Graph to perform real Teams operations like creating a breakout room chat or fetching user role data.
4b. **Bot Emulator:** During development, this stands in for real Teams so the team can fire commands and inspect payloads locally without touching the live tenant.
5. **Backend Services:** Both paths land here, where attendance records get written, the PAC queue is managed, and logs are kept for monitoring.

Implementation

- **Node.js** serves as the primary JavaScript runtime environment, handling asynchronous request/response processing and managing communication between the bot, Azure services, and Microsoft Teams.
- **Microsoft's Bot Framework Emulator** provides a local testing environment that simulates bot interactions, allowing for debugging and validation before deployment to Azure.
- **Microsoft Azure (Azure Bot Service / App Service)** is used to deploy, host, and scale the bot within a cloud environment, enabling secure integration with UNH's live Microsoft 365 tenant.
- **Microsoft Graph API** is integrated with Azure Active Directory authentication to securely access Microsoft Teams chat endpoints, enabling message retrieval and response handling within Teams channels.

Testing

- **Hands-On Testing**
  - Most of the testing was performed by the team on either Microsoft Teams or the Bot Framework Emulator for more basic local testing
- **Unit Testing with Jest**
  - Unit tests were written using Jest to ensure correctness in basic bot response functionality
- **Testing in UNH Tenant**
  - Sponsor allowed bots to be deployed to UNH owned courses Teams(CS419 & CS619) and tested with real students

Challenges

- Tenant Permission
  - The bot did not have the correct permissions because it was created in a separate tenant than the testing environment, solved by redeploying the bot in a newly created tenant
- Navigating / Configuring Azure
  - Hosting the bot requires many different components on Azure and it was difficult understanding what everything does

Evaluation

- Developer testing in sandbox tenant, more in UNH tenant planned.
- If we can get a beta test, we will evaluate usability and utility of the bot – whether commands are intuitive, whether help text is informative, whether breakout and attendance functionalities are helpful.

Next Steps

- Bot documentation and getting started guides written for next development team / maintainer
- Synchronize Violet Reynold with how the bot works and communications with UNH IT as she will be primarily responsible for maintaining the service in the future

Acknowledgements

Project Sponsor: Matthew Plumlee
Project Advisor: David Benedetto
UNH IT Staff: Kirk Francis, Ted Wisniewski, Dianna Dobe, Shari Starkey