Analytics and AI Service Line

<March 3rd, 2025>

< VoyaTrek: AI-Powered Step Tracking for Microsoft Teams>

Submitted By:

DAN HALES: Dan.Hales@voyatek.com

YIYI LUO: Yiyi.Luo@voyatek.com

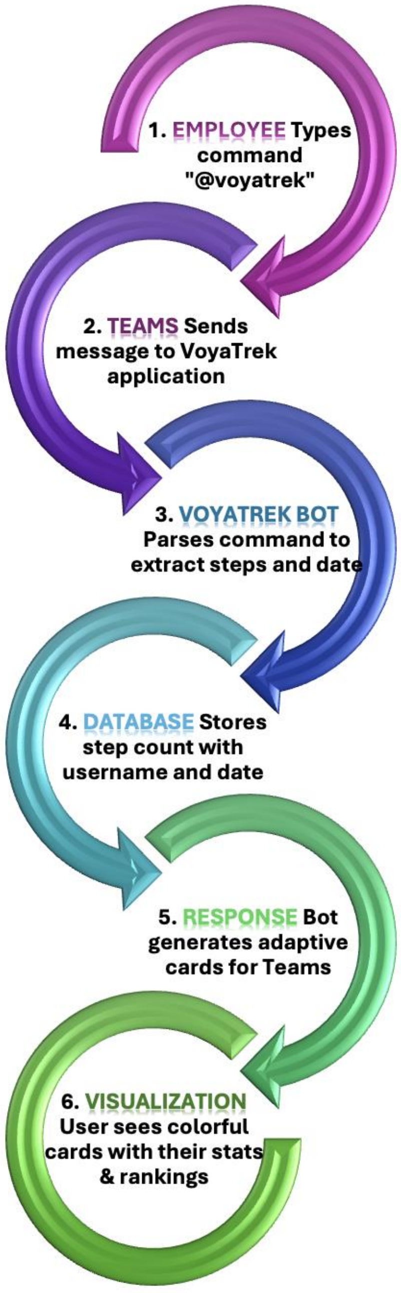
Supervisor:

ADAM ROY: Adam.Roy@voyatek.com

Voyatek

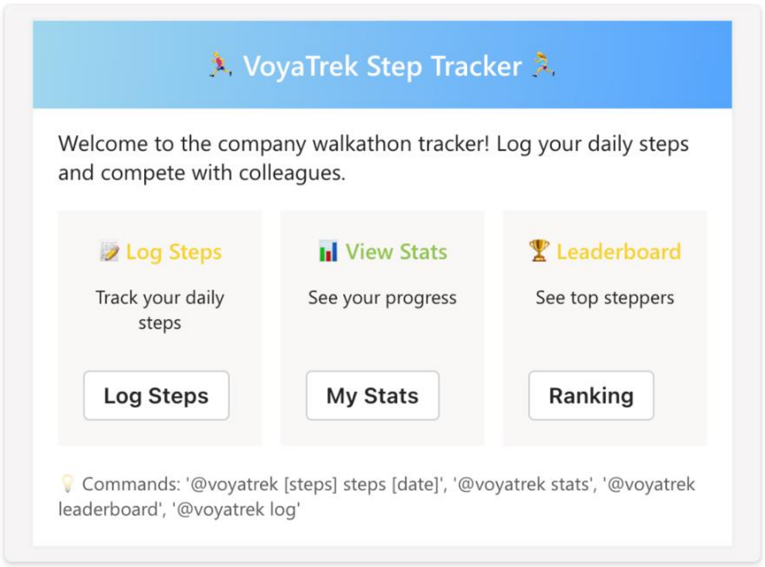
Table of Contents

VoyaTrek Workflow


1. EMPLOYEE Types command "@voyatrek"
2. TEAMS Sends message to VoyaTrek application
3. VOYATREK BOT Parses command to extract steps and date
4. DATABASE Stores step count with username and date
5. RESPONSE Bot generates adaptive cards for Teams
6. VISUALIZATION User sees colorful cards with their stats & rankings

VoyaTrek workflow showing the complete process from user input to visualization

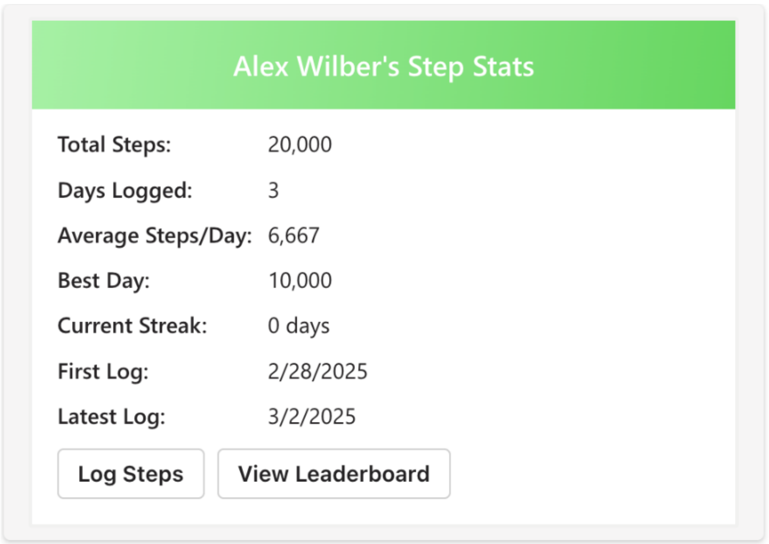
Application Interface


🏃 VoyaTrek Step Tracker 🏃

Welcome to the company walkathon tracker! Log your daily steps and compete with colleagues.

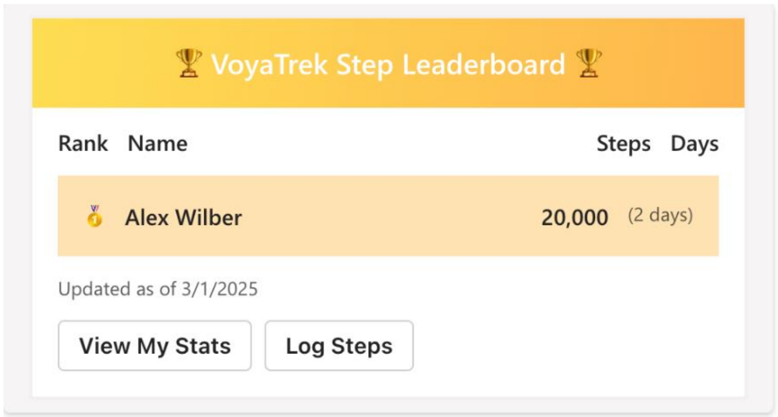
💡 Commands: '@voyatrek [steps] steps [date]', '@voyatrek stats', '@voyatrek leaderboard', '@voyatrek log'

Fig 1. Main Menu (Blue) - Central navigation hub providing access to all features


Alex Wilber's Step Stats

Total Steps:	20,000
Days Logged:	3
Average Steps/Day:	6,667
Best Day:	10,000
Current Streak:	0 days
First Log:	2/28/2025
Latest Log:	3/2/2025

Fig 2. Statistics View (Green) - Personal progress tracking with step totals and streaks



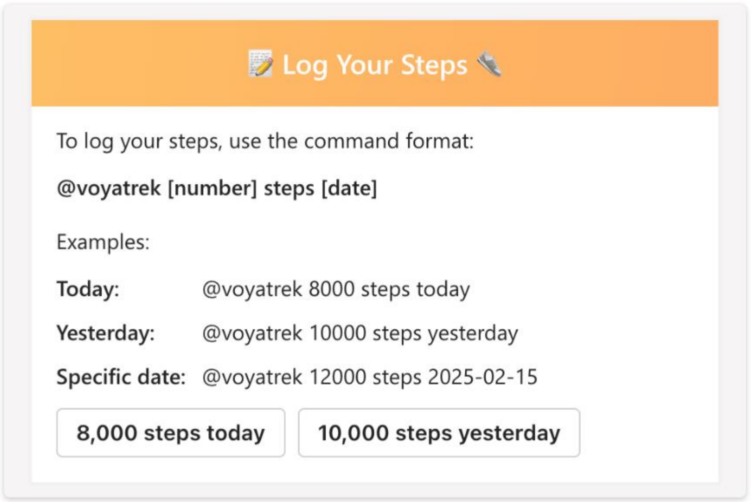
🏆 VoyaTrek Step Leaderboard 🏆

Rank	Name	Steps	Days
🥇	Alex Wilber	20,000	(2 days)

Updated as of 3/1/2025

Fig 3. Leaderboard (Gold) - Company-wide competition with medals for top performers

---



📝 Log Your Steps 👟

To log your steps, use the command format:

@voyatrek [number] steps [date]

Examples:

Today:	@voyatrek 8000 steps today
Yesterday:	@voyatrek 10000 steps yesterday
Specific date:	@voyatrek 12000 steps 2025-02-15

Fig 4. Step Logging (Orange) - Interface for recording daily step counts

---

This project was developed as a response to the "Request for Proposals on Generative AI Pilots" initiative. VoyaTrek demonstrates the capabilities of generative AI in creating real-world applications with minimal human intervention:

- The project began with a single human prompt describing the desired functionality, all subsequent development was guided by Claude's instructions; 100% of the code and implementation scripts were generated by Claude AI
- Humans provided feedback by sharing error messages and testing results, Claude AI generated solutions to all technical challenges encountered
- The documentation, workflow diagrams, and feature summaries were also created with Claude's assistance

The Original Prompt That Started It All:

"*We want to use gen AI to create a full-stack plugin for Microsoft Teams that allows everyone to log their daily step count via messages in the chat window. We propose the following syntax: '@voyatrek 10000 steps yesterday'. The plugin should detect this message, parse '10000', create a valid date for 'yesterday', and make an entry (username, 10000, '2025-02-28') in a Postgres instance on Azure. We should use AI to generate both a secure application meeting this spec and full configuration/setup instructions, then get the plugin approved for use by the whole company to submit steps for the summer walkathon*."

From this single prompt, Claude AI guided the entire development process, from environment setup to database configuration to user interface design, culminating in the fully functional application documented in this presentation.

I. Project Overview

VoyaTrek is a Microsoft Teams plugin developed to support the company's summer walkathon initiative. The application allows employees to easily log their daily step counts directly within Microsoft Teams, view their personal statistics, and participate in friendly competition through a leaderboard system.

1. Purpose

- Enable employees to track their step counts for the company walkathon
- Provide a simple, intuitive interface integrated with Microsoft Teams
- Foster friendly competition and encourage physical activity

2. Core Features

- Natural language step logging through Teams chat
- Personal statistics tracking with streak monitoring
- Company-wide leaderboard
- Interactive, color-coded user interface

3. Technology Stack

- **Frontend**: Microsoft Teams Adaptive Cards
- **Backend**: Node.js with Bot Framework SDK
- **Database**: PostgreSQL on Azure
- **Hosting**: Azure App Service

4. Functionality

VoyaTrek allows users to input step counts using natural language within Microsoft Teams. The primary command format is:

```
@voyatrek [number] steps [date]
```

Example commands:

- @voyatrek 10000 steps yesterday
- @voyatrek 8000 steps today
- @voyatrek 12500 steps 2025-02-15

The plugin processes these messages by:

1. Parsing the step count (e.g., "10000")
2. Interpreting the date (e.g., calculating "yesterday" as the previous day)
3. Recording the entry in the Azure PostgreSQL database with the username, step count, and date

Additional commands provide access to statistics and leaderboards:

- @voyatrek stats - View personal step statistics
- @voyatrek leaderboard - View company-wide rankings
- @voyatrek log - Access the step logging interface

The application features an intuitive, color-coded interface with distinct visual themes for each function:

- Main menu (blue) - Central navigation hub
- Statistics (green) - Personal progress tracking
- Leaderboard (gold) - Company-wide competition
- Step logging (orange) - Data entry

5. Implementation Approach

The development leveraged AI assistance to:

1. Design the application architecture
2. Generate the core application code
3. Implement database connectivity and data processing
4. Create an intuitive and visually appealing user interface
5. Develop deployment and configuration instructions

The resulting application is secure, user-friendly, and ready for company-wide deployment to support the summer walkathon initiative.

II. Creating the Teams App Project

1. Setting Up the Development Environment on MacBook

First, let's set up the development environment:

1.1. Install prerequisites:

```
# Install Node.js (LTS version) and npm via Homebrew
brew install node

# Install Git if not already installed
brew install git

# Install Visual Studio Code
brew install --cask visual-studio-code
```

1.2. Install Teams Toolkit extension:

- Open VS Code
- Go to Extensions (Ctrl+Shift+X)
- Search for "Microsoft Teams Toolkit"
- Click Install

2. VoyaTrek Teams Plugin Setup Guide

2.1 Project Setup

- ***a. Create a new Teams app with Teams Toolkit***:
  - Open VS Code with Teams Toolkit installed
  - Press F1 or Ctrl+Shift+P to open the command palette
  - Type "Teams: Create a New Teams App" and select it
  - Choose "Bot" capability
  - Select "Chat Command Bot" as the bot type
  - Name your app "VoyaTrek"
  - Choose JavaScript as the programming language
  - Select a folder to save your project
- ***b. Install additional dependencies***:

```
cd voyatrek
npm install pg dotenv
```

2.2 Configuring the Bot

- ***a. Define command in your manifest***: Open ./appPackage/manifest.json and add the following command:

```
"commands": [
  {
    "title": "track",
    "description": "Track your steps for the walkathon"
  }
]
```

- ***b. Create PostgreSQL database on Azure*** **(Please see section below for more details)**:
  - Log in to Azure Portal
  - Create a new PostgreSQL flexible server
  - Create a database named "voyatrek"
  - Note the connection string for later use

(Specific instructions of creating PostgreSQL database on Azure)

A. Creating a PostgreSQL Flexible Server on Azure

1. **Resource Group**:
   - If you have an existing resource group, select it
   - If not, create a new one by clicking "Create new" and name it something like "voyatrek-resources"
2. **Server Name**:
   - Enter a unique name for your server, such as "voyatrek-db-server"
   - This name must be unique across Azure, so try adding your initials or a random number if needed
3. **Region**:
   - Select the region closest to you for best performance
4. **PostgreSQL Version**:
   - Choose version 14 or later
5. **Workload Type**:
   - Select "Development" for testing purposes
6. **Compute + Storage**:

- Click "Configure server"
- For development/testing, select the "Burstable" tier
- Choose "Standard_B1ms" (2 vCores) or smaller
- Storage: 32 GB is sufficient for testing

7. **Admin Username and Password**:
    - Create a secure admin username and password
    - Write these down securely - you'll need them for your .env file
8. **Network Settings** (on the next page):
    - For initial testing, select "Allow public access from any Azure service and current client IP address"
    - You can tighten security later when moving to production
9. **Database Settings** (next screen):
    - You can leave these at defaults
    - After the server is created, you'll create a database called "voyatrek"
10. **Review + Create**:
    - Review all settings
    - Click "Create"
11. **Wait for Deployment**:
    - Deployment usually takes 5-10 minutes
12. **Create the Database**:
    - Once the server is created, navigate to your server
    - Click on "Databases" in the left menu
    - Click "+ Add" to create a new database
    - Name it "voyatrek"
    - Leave other settings as default
    - Click "Save"

For your current purpose of building a Teams plugin on your personal laptop first, you should designate this as a development server. Tags are optional metadata you can attach to your Azure resources to help organize and manage them. They're completely optional for your development setup. Keep the default values for DB_PORT and DB_SSL if you're using Azure PostgreSQL Flexible Server

If you see ETIMEDOUT error indicates that your application can't connect to the PostgreSQL database server. This is typically caused by Azure's firewall rules blocking your connection. How to fix it:

1. **Log in to the Azure Portal**
2. **Navigate to your PostgreSQL server resource**
3. **Configure the firewall**:

- In the left menu, click on "Networking" or "Connection security"
- Under "Firewall rules", you need to add your current IP address
- Click "Add current client IP address"
- Also enable "Allow public access from any Azure service within Azure to this server"
- Click "Save"

If you're working from home or on a network with a dynamic IP address, your IP might change periodically, requiring you to update the firewall rules.

After updating the firewall rules, try running your database setup script again. The connection should go through if the firewall is properly configured to allow your current IP address.

B. Creating the PostgreSQL Database

1. **Access your PostgreSQL server in Azure Portal**:
    - Go to Azure Portal and navigate to your PostgreSQL flexible server
    - In the left menu, click on "Databases"
    - Click "+ Add" to create a new database
2. **Configure the database**:
    - Set "Name" to "voyatrek" (exactly this spelling and case)
    - Leave "Charset" and "Collation" as their default values
    - Click "Save" or "Create"

c. Configure environment variables: Create a .env file in the root of your project:

```
DB_HOST=your-postgres-server.postgres.database.azure.com
DB_USER=your_username
DB_PASSWORD=your_password
DB_NAME=voyatrek
DB_PORT=5432
DB_SSL=true
```

III. Core files for the project

Create a database setup script:

- **Please see file dbSetup.js (in the root of the project) for this part**
  - **Run the script to set up your database:**

```
node dbSetup.js
```

Create the core implementation for VoyaTrek Teams app:

- **Please see file teamsBot.js (in folder "src" of the project) for this part**
  - **Note**: The index.js file is responsible for initializing and setting up your bot, while teamsbot.js contains the actual bot implementation with all the message handling logic.

After replacing the contents of teamsbot.js with the VoyaTrek Bot Implementation code, we may need to make small adjustments to match the export/import structure used in your project. For example, we might need to:

1. Check if the class name matches what's being imported in index.js
2. Ensure the exports at the bottom of the file match what's expected

Create a simple database utility file to help manage database operations:

- **Please see file db-utils.js (in folder "src" of the project) for this part**
  - **Note**: After creating this file, you may need to update the import statements in your teamsbot.js file to correctly import the database utilities. If your file is in the same directory, the import would look like:

```
const { getDbClient, getUserStats, getLeaderboard } = require('./db-utils');
```

Create a deployment script to help you deploy the application easily:

- **Please see file deploy.sh (in the root of the project)for this part**
  - **Note**: You need to make the deployment script executable before you can run it. This is a common security feature in Unix-based systems like macOS. Run this command in your terminal to make the file executable:

```
chmod +x deploy.sh
```

This command changes the file permissions to allow it to be executed as a script. After running this command, try executing the script again:

```
./deploy.sh
```

If you continue to have permission issues, you can alternatively run the script by explicitly calling bash:

```
bash deploy.sh
```

This second approach will work even if the file doesn't have executable permissions.

IV. Testing Locally

1. **Run your app locally**:
    - Select "Debug > Start Debugging" in VS Code
    - Teams Toolkit will launch your app in a browser
    - Sign in with your Microsoft account
    - Add the bot to a chat or channel
2. **Test the command**: Type "@VoyaTrek" in the chat

V. VoyaTrek Development Journey: Progress and Issue Resolution

Initial Implementation

- Created basic Teams bot structure with TeamsActivityHandler
- Implemented step logging with natural language parsing
- Set up PostgreSQL database connectivity
- Added date parsing for "today," "yesterday," and specific dates
- Created basic response handling for user commands

Feature Additions

1. **User Statistics Functionality**

- Added database queries to calculate total steps, averages, and streaks
- Created statistical summary with formatting for readability
- Implemented streak calculation logic to reward consistent activity

2. **Leaderboard Implementation**

   - Added company-wide ranking system based on total steps
   - Created a leaderboard display with sorting and formatting
   - Added visual indicators for top performers

3. **Interactive Card Interfaces**

   - Developed a central main menu card for navigation
   - Created specialized cards for each feature
   - Implemented button-based navigation between features

Issue Resolutions

1. **Command Recognition Problems**

   - Fixed the "command unknown" error by handling commands more gracefully
   - Implemented proper regex patterns for different command formats
   - Added safety checks for text processing and input validation

2. **Button Functionality Issues**

   - Initially attempted to use onTeamsCardAction which wasn't available
   - Switched to a custom card action handler using activity values
   - Implemented proper action handling for button clicks
   - Added error handling for action processing

3. **UI/UX Improvements**

   - Added color coding for different feature cards
   - Implemented gradient backgrounds for visual appeal
   - Added emoji icons for better visual cues
   - Fixed column height discrepancies in the main menu
   - Resolved button text truncation by adjusting layout and text

4. **Error Handling Enhancements**

   - Added safety checks for undefined activities and properties
   - Implemented graceful error handling for database operations
   - Added user-friendly error messages for various failure scenarios
   - Ensured proper database connection handling (connect/disconnect)

5. **Technical Debt Resolution**

   - Improved code organization with clear method structures

  - Enhanced comments for maintainability
  - Standardized UI component styles
  - Ensured consistent data handling throughout the application

Final Enhancements

- Added colorful themed cards for different functions:
  - Blue main menu for navigation
  - Green stats card for personal progress
  - Gold leaderboard with medal indicators for rankings
  - Orange step logging for data entry
- Implemented quick-action buttons for common tasks
- Fixed layout issues for consistent visual presentation
- Ensured all interactive elements work properly in the Teams environment

The development progressed from a basic command parser to a full-featured, visually appealing application with statistics, leaderboards, and intuitive UI—all while maintaining a robust backend for secure data storage and retrieval.

VI. Deploying to Azure

1. Provision cloud resources

- In VS Code, press F1 and select "Teams: Provision in the cloud"
- Follow the wizard to create Azure resources
- This will create an Azure Bot Service and App Service

2. Deploy your app

- Press F1 and select "Teams: Deploy to the cloud"
- Wait for the deployment to complete

3. Submitting for Approval

- **Prepare for submission**:
  - Update your app's privacy policy and terms of use
  - Complete all required fields in the manifest
- **Submit for admin approval**:

    - In Teams Toolkit, select "Teams: Publish to Teams"
    - This will generate an app package
    - Share the package with your IT admin for approval and distribution

4. Troubleshooting

- **Database connection issues**: Ensure your firewall rules on Azure allow connections from your IP
- **Bot not responding**: Check App Service logs in Azure Portal
- **Command parsing errors**: Review the regex pattern in your code

5. Security Considerations

- Use Azure Key Vault for storing secrets in production
- Implement proper authentication for the bot
- Sanitize all user inputs before storing in the database
- Implement rate limiting to prevent abuse