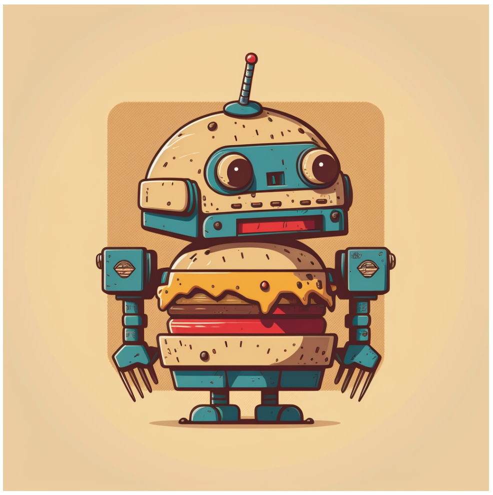
Report

Team: G511
Team members: Daniel Wei, Ken Chiem, Ryan Lee, Anthony Huang
Project Name: Bocaml
Github: github.com/epicdragon44/g511



Vision

Our vision for Bocaml was to create a "companion-like" Telegram bot that caters to the average users' diverse needs. The scope of its functionality spans a few key tiers: first, that of basic bot functionality, which includes health checks to ensure the bot is running; secondly, expected bot behavior, which covers a fairly diverse range of basic functionality expected of most standard companion bots such as helping a user with unit conversions, checking the weather, getting the current time in a global city; and lastly, extra bot behavior – a challenge we took upon ourselves to really build out intelligence in an industry currently dominated by AI news – in the form of two main features: an AI-powered Tic-Tac-Toe game that can be played against our bot (and which our bot never loses), and a GPT-powered "general chat" feature that lets our bot flexibly respond to whatever the user wishes to talk about, and thus completely fulfill the companion bot role. It is this last tier that was added on over time as our project grew in scope and we became more ambitious – apart from that, we've stayed true to our original vision, and executed it to the end.

Summary of Progress

Between MS2 and MS3, our team has made considerable progress. In the server codebase, we've rounded out the AI-powered features, including the OpenAI API-powered general discussion feature, as well as the Tic Tac Toe game, which utilizes a min-max function to literally never lose, rounding out another 250 lines of code or so. On the bot side, having finished the infrastructure during MS2, we now set to work implementing all of our functionality in a wrapper fashion that translates between the Telegram API and our own local server using all of the asynchronous material we learned in lecture. The result was another 500 or so lines of code. That is, of course, not to mention the infrastructural changes associated with massive development in a monorepo with two independent and self-contained OCaml projects: documentation was heavily revamped to be clearer and more well-organized, additional docs were written specifically for the perusal of 3110 graders, and scripts, such as for testing, code generation, and line counting, were refined, edited, and made more robust. We hope the effort we put in is apparent!

Activity Breakdown

Daniel:

Responsibilities:

- Maintained, developed, and expanded upon all repo-wide documentation.
- Redesigned and redeveloped many automation scripts.
- Implemented documentation and testing generation scripts.
- Integrated code from all other members between sprints.
- Coordinated sprint timelines.

Activities:

- Participated and occasionally led biweekly work sessions.

Features:

- Documentation that's readable, comprehensive, and clean.
- Makefile scripts that work.
- Simple random number generator, health checks, coin flips, and other "Tier 1" bot commands and server endpoints.

Hours worked: 20

Anthony:

Responsibilities:

- Broken down the unit conversion, currency conversion, and current time functionalities into multiple helper functions to make it more concise.

- Designed multiple unit tests to test the integrity of helper functions used for the three functions mentioned above.
- In addition to implementing the server-side function, I also implemented it on the bot-side, where the chat-bot is called whenever the user uses the correct command.
- Tended to any TODOs that were assigned in each sprint.

Activities:
- Participated in biweekly work sessions.
- Communicated with the team about any concerns/ideas the three functionalities that I was responsible for.

Features:
- My three features are:
    - Unit conversion: Converts a given amount from one unit of measurement to another; used pattern-matching as implementation
    - Currency conversion: Converts a given amount from one currency to another; used an external currency exchange API to implement
    - Current time query: Returns the current time in a given timezone; used an external time API to implement

Hours worked: 12

Ryan:

Responsibilities:
- Implemented server-side AI tic tac toe game handler as well as a weather fetcher and translator
- For AI handler:
    - Implemented interface, complete with reference game board and mutable textual representation of current game board, showing available/valid spots to place markers
- Extrapolated functionality from the aforementioned functions, creating a test suite for each helper
- Implemented bot-side functions that communicate with endpoints, processing user queries and outputting appropriate information
- Helped bulk testing suite, ensuring coverage is satisfactory

Activities:
- Participation in biweekly work sessions and meetings

- Regularly communicated with team regarding logistics and design of server-side/bot-side features

Features:

- AI Tic Tac Toe opponent: Leverages Minimax algorithm to choose the "best possible move in the worst possible situation" (i.e. you cannot *beat* it!)
- Textual interface, detailing the structure of the board, and the keys that map to specific positions on the board
- Weather handler: fetches the weather given a location, communicating with an API to accomplish this
- Translator: translates given body of text from the current language to another specified language, likewise uses an API to do so

Hours worked: 15

Ken:

Responsibilities:

- Broken down the main chat-bot functionality into multiple helper functions to make it more concise.
- Designed multiple unit tests to test the integrity of helper functions used for the general-chat-bot.
- In addition to implementing the server-side function, I also implemented it on the bot-side, where the chat-bot is called whenever the user uses the correct command.
- Tended to any TODOs that were assigned in each sprint.

Activities:

- Participated in biweekly work sessions.
- Communicated with the team about any concerns/ideas regarding the chat-bot feature.

Features:

- BoCaml has a chat-bot feature, where users can give it any question or general message that they want to send to the chat bot, and it will return a response to the user.
- Done by integrating the OpenAI API in the server-side folder, where the users questions/inputs will be sent to ChatGPT, and the response will be parsed and given as a response back to the user.

Hours worked: 13

Productivity Analysis

As a team, we've really just continued to be incredibly productive throughout the development process. Despite seasonal midterms and exams, we've put out code on a biweekly basis quite consistently, and at each turn, integrating our code together into main has resulted in few conflict errors and less problems than anticipated. In each sprint, we accomplished what we set out to do, whether that was the complete implementation of AI-powered features in the server, or the complete implementation of all linked functionality in the bot. As a whole, the organization of task distribution we aimed to create has been wildly successful – we've never run into the case where we're all bottlenecked by a single developer unable to complete a task. With the completion now of our holistic MVP, we're confident in saying that as a whole, our productivity was really quite above our own expectations.