An Autonomous Discord Bot to Improve Online Course Experience and Engagement: Lessons Learned Amid the COVID-19 Pandemic

Devin Robert Wright
Utah Valley University
DWright@uvu.edu

Tim Severence
Utah Valley University
Tim.Severance@uvu.edu

Charles D. Knutson
Utah Valley University
cknutson@uvu.edu

Jonathan L. Krein
Crimson Vista, Inc.
jonathan@crimsonvista.com

Tyler D. Buchanan
Utah Valley University
tbuchanan@uvu.edu

Abstract

The COVID-19 pandemic pushed many educational institutions to adopt online learning models for most or all of their courses. As a result, the effectiveness of remote learning is more important now than ever before. In this paper, we report on work that was conducted in the Spring of 2021 at Utah Valley University. We explored the use of Discord as a delivery mechanism for online course content during the 2020-2021 school year. We also developed a Discord bot to autonomously track attendance. Based on our experience to date, the Discord bot appears to enhance remote learning. We describe the design, implementation, and deployment of our bot. We also discuss what worked well, as well as areas for improvement. In future semesters we plan to collect data by which we may begin to answer fundamental questions about the impact of such bots on remote learning.

1. Introduction

In early 2020, the COVID-19 pandemic significantly uprooted face-to-face university classes across the world, necessitating an unprecedented migration to online learning formats and platforms over a very short time frame. Not surprisingly, most institutions experienced some degree of disruption during this time and faced new challenges. Utah Valley University (UVU), in Orem, Utah, was no exception.

On March 6th, 2020, the state of Utah officially declared a state of emergency, and by March 23, 2020 UVU had officially transitioned to a 100% remote learning model. The announcement to teach all classes remotely set in motion a rapid and unanticipated transformation for many classes at UVU. Instructors and administrators alike were suddenly forced to locate and integrate new tools to facilitate virtual classrooms. In such an environment, ease of use and reliability became critical, albeit somewhat elusive, priorities.

UVU eventually designated Microsoft Teams as its official virtual classroom solution. However, during the transition to Teams, a number of limitations became apparent. First, the university-wide scale up to using Teams led to predictable outages and failures. Second, we faced a number of key weaknesses regarding feature support within Teams[1]. Given these limitations, the authors assessed a number of other virtual workspace platforms (including Slack, Zoom, Google Classroom, and Discord) and ultimately selected Discord.

1.1. Motivation for Discord

This paper is not strictly about Discord, but an understanding of Discord and the motivation behind our decision to use Discord is necessary to understand the context in which we developed our asynchronous and automated attendance tracking bot.

Discord is a platform that organizes participants into communities, called "guilds"[2] – i.e., hosted digital spaces in which users can congregate and interact via voice, text, video, screen, and file sharing. In Discord, guilds (or servers) enable the creation of both voice and text "channels," which represent the fundamental mechanism for communication between community members. Communication over voice channels includes optional overlaying of screen, video, and file sharing. Voice and text channels can also be overlayed onto one another, thus allowing the user to create an array of custom communication configurations. Such flexibility proved invaluable for creating an effective and engaging learning environment.

In addition to Discord's capability to overlay multiple communication methods in an array of custom configurations, an informal poll of students suggested that a majority of CS students at UVU were not only familiar with Discord, but already spent significant time (primarily socializing) on the platform. Based on this

[1] As an example, Teams provides limited administrative control to those conducting meetings, whereas Discord gives the creator of a server full administrative control over it. Other weaknesses relate to specific usability concerns in which Discord clearly outshines Teams.

[2] Also commonly referred to as "servers" in Discord.

URI: https://hdl.handle.net/10125/79442
978-0-9981331-5-7

and other anecdotal evidence available to date, we believe the popularity of Discord among CS students at UVU is due to four factors: 1) Discord was initially built as a social networking platform for the gaming community; 2) A significant percentage of CS students are online gamers; 3) Discord features an elegant design and intuitive usability, coupled with a rich set of communication features – i.e., it's just a really good product; and 4) The platform is available to users at no cost.

Since much of our student body already utilizes Discord for gaming and socializing, it appeared to be a good candidate for use in our remote classroom. Additionally, in practice, we found Discord's intuitive UX ("user experience") design facilitated a smooth transition for most students, including those who had not previously used Discord. Further, in contrast to the other platforms we considered, Discord provides full administrative control over a server (or guild) once it's created, allowing the admin to assign roles and permissions to teaching assistants (TAs) and students. This capability for custom permissions significantly improved server and channel management, since it was better suited to our distributed model of course management – professor and TAs working together to administer a course.

1.2. Need for Automated Attendance Tracking

After a partial semester (Spring 2020) using Discord, followed by a full semester (Fall 2020) using the platform, we perceived a need for administrative automation within the platform to streamline certain aspects of our courses. Discord is, of course, not focused on academic applications or functionality. Our previous non-automated methods were tedious, time consuming, and error-prone – especially for tracking attendance. Inspired by these challenges, we conceptualized an autonomous attendance tracking system that could run as a background process on Discord, eliminate our previous cumbersome and inaccurate manual systems, and also provide a means for gathering data that could be used to analyze select student behaviors, thereby facilitating the future study of factors that might lead to increased student success (including classroom engagement).

1.3. Structure of this Paper

In the following section, we discuss literature related to classroom attendance within the context of online education and describe prior attempts to automate the tracking of attendance. Subsequently we provide additional background information regarding the Discord platform and our use of Discord to facilitate remote learning. We then describe the design and implementation of our automated attendance tracking solution, followed by a discussion of our deployment of that solution in the context of a computer science course at UVU (in the Spring of 2021). Finally, we discuss some of the limitations of our tool as well as ideas we are considering for future work and general conclusions.

2. Related Work

2.1. Motivation for Automated Attendance Tracking

Studies suggest that well-prepared and/or motivated university students generally perform similarly in both face-to-face and online classes. Conversely, the performance of average students tends to decrease in online settings, as compared to face-to-face classes. In addition, those students who are least prepared for higher (e.g., university) education (including low-income students and those with low high school grade point averages) demonstrate an even larger negative gap between their online class performance and their face-to-face class performance [1][2]. Jaggars and Xu suggest that this negative performance gap relates to a number of propensities and skills present (or more well-developed) in high-achieving students, including: (1) a high level of self-regulation and self-discipline, (2) a propensity to proactively seek help from teachers, and (3) a number of other metacognitive skills "which often fall under the broad rubric of self directed learning" [2]. In all of these areas, Jaggars found low-achieving students to be generally lacking or less well-developed than high-achieving students [1].

As an open enrollment university, UVU's 41,728 students include a significant percentage of non-traditional students (at least 32%[3]), as well as a large cohort of first-generation students (36%) [3]. While non-traditional community college students don't manifest as large a performance gap between online and face-to-face learning, they typically perform more poorly than the average university student in both online and face-to-face classes [2]. First-generation students are also more likely to be underprepared for a university education [4].

Given the inherent discrepancies between student performance in online courses vs. face-to-face courses, our research was driven by a desire to improve online courses to more closely mirror the face-to-face experience, thereby (hopefully) narrowing

[3]This figure may be higher, depending on the definition of "non-traditional" one applies.

the discrepancy. Having observed a steep decline in attendance when UVU transitioned to 100% remote learning (as a result of the pandemic), we considered class attendance to be an obvious area of remote learning where improvement was needed. Our current working hypothesis is that *the known use of an automated attendance tracking system improves remote learning outcomes* (e.g., it increases actual attendance, exam scores, and student satisfaction). However, at the very least (and from a practical standpoint), we believe such a tool enables the collection of accurate attendance information in an online environment, such that we can hold students accountable in courses where attendance is believed to be an important component of the learning experience. Further, gathering accurate data on student participation ultimately enables future studies concerning the impact of such tools on student performance (i.e., the tool will allow us to test our core hypothesis going forward).

Allensworth enumerates several aspects of student engagement, including behavioral, emotional, and cognitive elements. He points out, "Only behavioral engagement is observable by others, making it a crucial signal of overall engagement to which educators need to attend." [5]. One of the most evident forms of behavioral engagement is attendance. Consequently, behavioral engagement markers – and in particular, tracked attendance – are crucial to the evaluation of overall student engagement.

Becker supports the idea that the accuracy with which student involvement (in course material) is assessed and communicated impacts student learning outcomes [6]. Also, in the context of face-to-face courses, Gomis and Rodrigues suggest that absenteeism increases as access to online materials increases [7]. Consequently, as the online material provided to students expands to subsume all course material – essentially obviating (in the mind of the student) the necessity of in-class time as a means for information delivery – (1) class attendance drops significantly, (2) student happiness mid-semester increases marginally, (3) exam performance decreases, and (4) final grades decrease significantly (after which one might reasonably assume student happiness then decreases, though such an effect was not assessed by Gomis and Rodrigues) [7]. Additionally, and consistent with our earlier observations, committed and driven students tend to attend class regardless of other factors [8].

2.2. Non-Discord Attendance Tracking Methods

2.2.1. Manual Methods Traditionally, class attendance is tracked manually, either through a verbal roll call or by allowing students to mark themselves present on a class roll. These traditional methods for recording attendance present problems that are difficult to resolve. Depending on the size of a given class, roll call can consume significant time that would otherwise be dedicated to lecture or other in-class activities. As an example, for a typical three-credit course, forty to forty-five hours of in-class time is allotted for a semester. Masalha suggests that as much as eight hours of class time per semester (i.e., 17-20% of a typical three-credit course) may be lost to the traditional roll call method for attendance tracking [10].

A common substitute for roll call is to allow students to mark themselves present, typically on a sheet of paper. Not only does this create distractions during lecture, but for classes in which attendance is a major part of a student's grade, this approach presents students with the opportunity for a moral dilemma, such as enabling them to mark absent peers as present. Additionally, if an instructor includes multiple days on a roll sheet being passed around the class, students may avail themselves of the opportunity to retroactively mark themselves present for days on which they had actually been absent. The core problem here is that the instructor does not have an effective means by which to validate the data once collected, nor an efficient means for migrating it into a digital format [9].

2.2.2. Partially and Fully Automated Methods One study proposed marking attendance by requiring students to scan a unique daily QR code. This approach makes it more difficult for students to falsify attendance, since it requires the individual to be physically near the QR code in order to scan it with their mobile device [10]. However, it doesn't prevent a student from snapping a picture of the QR code and forwarding it to a friend who is not present. It also requires the professor (or TA) to create QR codes and display or circulate them in some fashion as part of each class period, which can be burdensome and time consuming.

Other innovative methods for attendance tracking include fingerprinting [11], RFID ("Radio Frequency Identification") [12], facial recognition [13], and NFC ("Near Field Communications") technology [13]. While many of these methods are technically elegant, all of them require the adoption of specific technologies (hardware as well as software) by both students and institutions. Additionally, each of these solutions has certain limitations. For example, QR and NFC methods do not account for students who are present

but forget to mark themselves as such. RFID can be unreliable in precisely detecting student location. Many of these solutions are also incapable of reliably tracking certain important data – such as time of arrival, time of departure, and amount of time present – without disrupting the classroom setting, wasting time, or adding unnecessary complexity to what should ideally be a simple process.

In contrast to these proposed solutions, we sought to create a fully automated attendance tracking system with a low barrier to entry, few points of failure, high resilience to user error or student inaction, that also facilitated comprehensive collection of attendance-related metrics.

2.2.3. Technologies for Recording Attendance

Academic institutions have relied on industry-leading educational platforms, such as Canvas and Blackboard, for many years. However, as Cacho suggests, even the most current and well-funded products tend to provide outdated functionality [14]. For example, these tools are often inflexible, lack video conferencing capabilities, and provide outdated forum style discussion boards that are not conducive to active participation by the contemporary student [15]. Additionally, such tools typically require manual input of attendance, which, as we discussed above, suffers from several key weaknesses [9].

As Olson asserts, "Technology firms are eager to build more functional tools for users but they need both guidance and coordination from the field to build solutions that work for more than a single use in a single school system. Tech providers should codesign such solutions with the ultimate end users: educators and school system leaders." [16]. To the extent educational technology providers have failed to be responsive to educators and school system leaders, or have failed to keep up with the expectations and needs of the contemporary student, some educators have seen success in deploying their own solutions [9] [17] [18].

2.3. Discord-based Methods for Attendance Tracking

As mentioned previously, in early 2020 UVU transitioned from traditional face-to-face classroom teaching to an entirely online-based approach. At that time, the authors of this paper made the decision to conduct classes using the Discord platform. While not a primary motivation for our initial decision to use Discord, the platform's support for third-party application development (such as plugins and bots) proved a significant advantage as we explored the possibility of automating the tracking of attendance.

A number of online activities on Discord can be managed by the creation of a particular type of automated software application referred to as a "bot" [19]. Discord bots look and act (to the system) like user accounts, but automatically perform actions that would otherwise require manual human intervention. In fact, a number of Discord bots have already been created to specifically address the issue of automating the tracking of virtual meeting attendance. Notable examples of these Discord attendance bots include:

- Suivix – capable of creating lists of absent and present users for a given activity [20]. It allows attendance to be recorded for up to eight voice channels and eight user roles. Additionally, it can create anonymous and flexible surveys (referred to as "polls"). In contrast to most Discord bots, Suivix provides a full-featured user interface, allowing a great deal of configuration control, independent of the Discord interface.

- AmtBot – manages attendance for online events related to a specific live action role playing game called Amtgard [21]. The bot is started by the admin of a Discord server and allows users to mark themselves present. It then tracks their status throughout a given meeting and then provides information to the admin (as well as to the attendee) related to the user's attendance.

- Integromat – a robust and powerful tool that provides integration with numerous platforms and automates a variety of tasks [22]. Integromat offers an attendance bot for Discord meetings, among a host of other features.

- Apollo – manages events and attendance, and pings users to remind them when an event is about to start [23]. It leverages Discord's permissions and roles capabilities, and offers a simple and intuitive user interface for Discord servers.

Our analysis of these bots revealed deficiencies with respect to specific features we desired in our automated solution (including granular tracking of student arrivals and departures, flexible calculation of attendance grades, and support for multiple courses, among others). Given these limitations, we determined to design and implement our own Discord bot to automate the tracking of course attendance. As an added benefit, implementing our own bot allowed us to maintain full control over the bot in real time, as well as giving us the ability to expand its functionality down the road.

3. Applying Discord to Remote Learning

3.1. The Discord Usage Model

Discord is a popular platform for group communications, with over 140 million active monthly users [24]. Its functionality is similar to Slack, Google Chat, and Microsoft Teams, but unlike those platforms, typical Discord users don't simply use the platform for scheduled meetings or events. Rather, they tend to integrate the platform as a regular part of their online life. As the Discord site says, "Imagine a place where you can belong to a school club, a gaming group, or a worldwide art community. Where just you and a handful of friends can spend time together. A place that makes it easy to talk every day and hang out more often" [25]. Typical Discord users really do tend to hang out on the platform in the same way that most digitally connected people stay in contact with others via text messaging or chat platforms (such as Messenger).

Discord users can join existing communities ("servers" or "guilds") or create their own servers in order to invite others. The members of a given server form an exclusive, invitation-only community within Discord. Within a server, one or more "text channels" (including a default channel called "general"), and one or more "voice channels" (including a default channel called "General"), facilitate communication between members. Once a user joins a voice channel, the effect is that of a conference call in which all participating users can talk to each other. Additionally, within a voice channel any individual on the channel can share their screen (or specific applications on their computer) and/or turn on their camera to facilitate an interactive video chat (or both). During video and voice chats, text channels remain easily accessible to users.

3.2. Using Discord as an Online Classroom

To set up Discord for use as an online classroom, we created a new Discord server for the course we were teaching. The course was titled "Global Social and Ethical Issues in Computing (CS 305G)." We cleverly named our new server "UVU CS 305G," and a custom logo was generated for it by the students during our first class period together online. We also established three administrative roles for the new server: 1) "@admin," the default administrator role, which we assigned to the instructor and teaching assistants; 2) "spring2021," which included all students currently enrolled in the course; and 3) "AssistantBot," to which we assigned the necessary permissions to enable our attendance bot to perform its duties of taking attendance.[4]

As we discuss below, Discord bots are applications that look and act like regular users within a Discord server. As functional users, bots can be assigned roles, thus enabling them to be granted whatever privileges are necessary for them to perform their function.

4. Design and Implementation of a Discord Attendance Bot

Discord bots are comprised of two key components: First, an infrastructural component that represents a Discord "application" of type "bot," which is configured per Discord's documentation to communicate with Discord servers in the cloud. This part of the attendance bot is created using the framework provided by the Discord Developer Portal.[5] Second, a business logic component. This part of the attendance bot represents the core application code that must be custom designed by the developer, and which constitutes the domain-specific behavior of the bot (See Figure 1). We discuss each of these two components in turn.

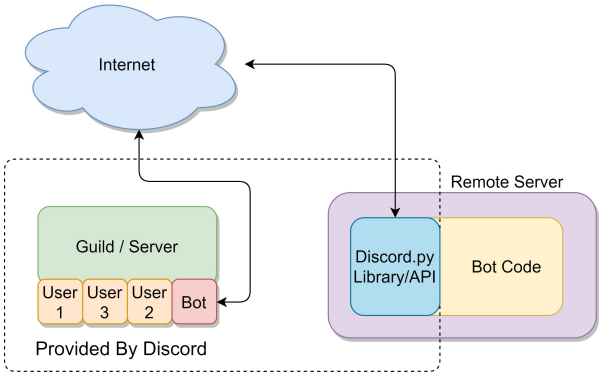

Figure 1. Discord Bot Infrastructure.

The Discord Developer Portal provides an easy-to-use interface that allows any Discord user to create a Discord "application."[6] Having created the application, the Portal interface then allows the developer to add a "Bot" user to the application, effectively (and irreversibly) specializing the Discord application with the capability to represent an (automated) user within the system. A Discord application, having been specialized as a bot, now

[4] Since we had initially conducted class on Discord during the Fall 2020 semester, a number of students from the previous semester were still members of the server. By establishing a role for students registered during the current semester, we were able to limit access to certain resources to only those students. We were also able to message current students without spamming former students.

[5] See https://discord.com/developers/applications.

[6] See the Discord Developer Portal for documentation on how to do this.

acquires the capability of interacting with a given Discord server in (effectively) the same way as a human user might. Although a Discord bot can be thought of as running on the Discord platform, it actually runs on a computer of your choosing, from which it interacts with the Discord platform via standard network calls. Its ability to communicate with the Discord platform is based on the infrastructural component or functionality within its source code, whereas its behavior (as a user on the Discord platform) is determined by the business logic component.

As just explained, part of the process of creating a Discord bot requires establishing certain configurations within the Discord Developer Portal, which are necessary for the Discord platform to recognize the bot code created by the developer. As part of the portal configuration process, Discord generates a unique token, which is provided to the developer via the portal user interface. When setting up the infrastructural component of the bot's application code, the developer uses this token (as shown in Discord's documentation, cited previously) to establish a connection or session with the Discord platform. Communication with the Discord platform, including use of the token, is facilitated by a Discord-provided software library, which provides standard functions to the developer, by which the developer can interact with the Discord platform. This software library is simply imported into the bot's application code and runs as part of the bot on the same computer with the bot. Discord provides interface libraries for both the Python and JavaScript programming languages. We chose to develop our bot application code in Python, and thus we utilized the Python-based "discord.py" library.[7]

Our bot code is organized into two Python scripts (or program modules). The first script ("AssistantBot.py") encapsulates the infrastructural component or functionality of the bot. AssistantBot.py establishes and maintains the bot's connection with the Discord platform. It also initializes and launches the business logic or behavioral component of the bot. The portion of the source code that implements this latter component primarily resides in a structure that Discord refers to as a "cog" – Discord jargon for a base or parent class from which the developer's code must inherit (in a programming sense). The second script ("AdminFunctionality.py") defines the "AdminFunctionality" cog, which we implemented to provide the bot with the behavior and functionality to operate as an attendance tracker.

In addition to performing initialization and launching the AdminFunctionality cog, AssistantBot also defines three administrative commands:

- load
- unload
- reload

These commands are used to load, unload, and reload any cogs supported by the AssistantBot. While it is true that the AdminFunctionality cog is loaded by AssistsantBot when the bot joins a Discord server, by implementing these commands we preserve the ability to manually load and unload any present or future cogs without having to take down the bot, thus providing an extensible design.

Communication with a bot is achieved by typing commands in a text channel on a given server to which the bot is connected. Each bot on a Discord server defines a prefix string, which enables the bot to receive messages through the text channel. When the first portion of a typed string matches the string defined by the bot, the bot consumes the text chat and treats it as input. Therefore any text string typed by a user on a text channel that begins with the prefix string defined by a given bot will be passed to the bot as input (with the prefix string removed). Prefix strings can be a single character (such as '>') or an entire word (such as 'heybot!'). We tried a number of prefix strings during the semester, ultimately settling on ')'.

The AdminFunctionality cog defines the following commands:

- clear – Deletes messages associated with the bot from the text channel on which the command is run.
- change_prefix – Allows the user to change the prefix string used to communicate with AssistantBot.
- change_channel – Allows the user to change the voice channel on which attendance is taken by AssistantBot.
- attendance_start – Initiates the attendance sampling activities of AssistantBot.
- attendance_stop – Terminates the attendance sampling activities of AssistantBot.

Before discussing the process by which AssistantBot actually takes attendance, we first have to address and resolve a critical data translation problem. When connected to a voice or text channel, the AssistantBot is capable of identifying the other users within the same channel, and is able to do so by retrieving the

[7] See https://pypi.org/project/discord.py.

persistent identifier for each user (assigned by Discord) as well as the username and nickname of the user (determined by the user for that specific server). An obvious solution would be to have each user set up their nickname in a consistent manner (for example, "Jones, Monica"). The challenge, however, is threefold. First, getting several dozen students to execute such a maneuver without error is a potential fool's errand, with dozens of points of failure. Second, and perhaps more importantly, nicknames can be changed by users without limit and Discord users regularly change their nicknames and usernames for a variety of reasons. Third, certain students in a given class may prefer, for personal reasons, to function in the course under an assumed nickname, and not under their given name (or the name listed on the records of the university). As a result of these factors, we resolved to track attendance based strictly on the persistent identifier of the students.

The solution to this dilemma was to create a mapping between each student's name and their persistent identifier. The mapping between names and persistent identifiers was stored (i.e., hardcoded) in a Python dictionary data structure within the file AdminFunctionality.py. Given the persistent nature of Discord identifiers, as long as we could obtain them and accurately represent them in the Python dictionary (mapping them to student names), we eliminated all other points of potential failure and assured accurate attendance taking going forward.

With the mapping between persistent identifiers and student names securely stored in the bot source code, the final step was to assess attendance of students during class times. All lectures were conducted in the "General" voice channel for the "UVU CS 305G" server, so attendance at lecture was deemed to constitute being present in the General voice channel on the server during the timeframe designated for each class period. There are certain threats to validity in this approach, which we discuss below.

Discord allows a user with administrative privilege for a given server to poll the system to ascertain the identifiers of all users currently present on that server. This lookup is performed with a simple query (see Figure 2 below). The responses to the query are indexed into the dictionary mentioned above, and student names are thereby obtained. Attendance for each student is then recorded in a secure database.

We were interested not only in a binary view of student attendance (present vs. absent), but also in patterns of absence and partial attendance. By polling Discord every minute, we were able to track arrival and departure times, including multiple disjoint periods of attendance at the same lecture. We were thereby able

```
@commands.command()
    @commands.has_permissions(administrator=True)
    async def poll_channel(self, ctx):
        #poll voice channel for present members
        attendance_channel = discord.utils.get(
            ctx.message.guilds.channels,
            name="General",
            type=discord.ChannelType.voice)

        students_present = attendance_channel.members
```

Figure 2. Query to poll a voice channel.

to acquire a robust set of data that revealed patterns of tardiness, premature departure, and even students who showed up at the beginning of class (presumably because they knew we were taking attendance) but then vanished. At the beginning and end of each class attendance tracking was manually started and stopped. Figure 3 below shows the commands to start and stop the bot typed into the general text channel on the "UVU CS 305G" server.

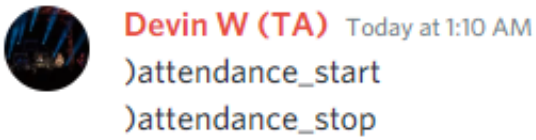

Figure 3. Commands to start and stop attendance tracking.

5. Deployment of the Discord Attendance Bot

The developer code that powers a Discord bot can be hosted essentially anywhere code can be run, as long as the hosting computer has internet connectivity. Some bot developers have chosen to run their code on Amazon Web Services, Microsoft Azure, or an other cloud hosting service. Others have chosen to run their code on a personal laptop or desktop machine.

We chose to run our code on a Raspberry Pi developer board dedicated to the task of running the AssistantBot. Data gathered by out bot was stored on the same Raspberry Pi device, but that data could easily be saved to a remote location, such as to a remote database. By leaving our server powered up and connected to the Internet, it was available to us any time, with no inconvenience and at very low cost. Figure 4 shows the bot running on the Raspberry Pi server, performing attendance tracking activities.

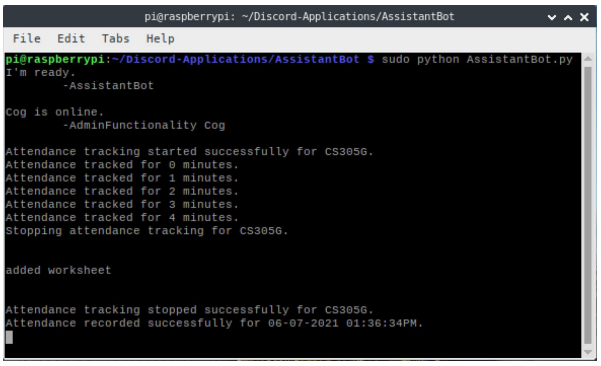

Figure 4. Attendance tracking running on Raspberry Pi server.

6. Limitations

A number of limitations of the Discord attendance bot represent threats to validity with respect to data collection:

- Our attendance bot determined the presence of a student based on whether the student was connected to the designated lecture voice channel. However, a student could obviously be counted as attending class without effectively attending (for example, by muting their speakers or leaving their computer after connecting).
- The attendance bot recognized attendees based on an internal hardcoded mapping between their persistent Discord identifiers and their name of record with the university. However, some students have multiple Discord accounts, and we encountered some students who repeatedly connected with an account for which we did not know the persistent identifier, despite having been instructed to use the same account each time they attended lecture. Some of these students were also unwilling to provide the necessary identifying information for their alternate account, which resulted in unnecessary difficulties correcting the attendance record.
- Although it did not happen during the semester in which we used the attendance bot, if a network connection were to fail for the device on which the attendance bot was running, the attendance bot would fail to collect attendance data.

Other limitations of the bot require manual effort by the professor or TA to overcome:

- The professor or TA must manually create the mapping of student names to persistent Discord identifiers for every student every semester. This process was the most time consuming aspect of deploying the bot once the bot itself was constructed.
- We also had to manually input the attendance data (collected by the bot) after each class period into the university's grading system, which is a tedious process. As a result, we are considering the possibility of integrating the bot with the university's grading system (see future work below).
- Tracking attendance with the bot required manually starting and stopping the bot's tracking functionality at the beginning and end of each class period. This task was not difficult or time consuming to perform, but did require diligence to remember that it be done consistently.

7. Future Work

As mentioned above, we had to manually import the attendance data from the bot into the university's grading system. Although not all grading or course management systems provide the necessary programming interfaces to automate that process, where such interfaces do exist, the bot could be enhanced to automatically transfer attendance data, either immediately after a class period ends, or upon receipt of a command from the bot administrator.

We are considering enhancing the bot to reference a course schedule and automatically start and stop its tracking of attendance to correspond with scheduled class times. This could be accomplished by creating a separate thread that runs on an interval (for example, every minute) while the bot is active, checking the schedule to determine whether to start or stop tracking attendance for a given class.

It would be convenient for the bot to flag any attendees who are not found in the mapping between student names and persistent Discord Identifiers. In doing so, the bot could also log the persistent identifier, username, and nickname of that person. The professor or TA could then use that information to determine the identity of that person. If the person is a student from a past semester, their information could be added to an ignore list maintained by the bot. If the person is a current student, the identifier could be added to the mapping of current students. Given such a strategy, the professor/TA would not have to rely on students to provide this information in the event they mistakenly use a secondary account.

We are considering adding additional bot commands

that can be executed by the students, such as commands to allow a student to retrieve their attendance record. The bot's scope could also be increased, or additional bots could be created, to implement student commands for things like retrieving the course schedule or other course information, requesting help, requesting lecture slides for a given day, etc.

In addition to further development work on the attendance bot, we also need to collect systematic data by which to empirically assess the actual impact the bot has on learning outcomes and student success. Does the bot, for example, actually narrow the gap between remote and in-person learning, and if so, for which segments of the student population?

8. Conclusions

In 2020, the world experienced what may be the first international wholesale move to virtual learning. During that time, many educators recognized a need to change the status quo with respect to remote learning and online education.

Faced with this transition, we made our own modest contribution by exploring new solutions to remote learning, including the use of Discord as an online classroom platform. We sought to bridge the gap between in-class and online learning by developing an autonomous attendance tracking bot for the Discord platform. In the Spring semester of 2021 we successfully developed and deployed a basic working version of an attendance tracking bot and then utilized our bot to track attendance in two courses during the semester.

We found that a Discord attendance bot can not only accurately track the arrival and departure times of students, but can also mitigate many of the pitfalls associated with traditional attendance taking methods. Given the rapid deployment of our attendance bot, we are well aware of a number of opportunities to further enhance the bot based on lessons learned, as we have outlined above in Limitations and Future Work.

References

[1] S. S. Jaggars and T. R. Bailey, "Effectiveness of fully online courses for college students: Response to a department of education meta-analysis," tech. rep., Community College Research Center, Teachers College, Columbia University, New York, New York, July 2010.

[2] D. Xu and S. S. Jaggars, "Performance gaps between online and face-to-face courses: Differences across types of students and academic subject areas," *The Journal of Higher Education*, vol. 85, no. 5, pp. 633–659, 2014.

[3] S. Trotter, "Enrollment numbers show UVU serves a diverse array of learners." https://www.uvu.edu/news/2019/10/10022019_enrollment.html, Oct. 2019. Accessed on 18-May-2021.

[4] A. R. Unverferth, C. Talbert-Johnson, and T. Bogard, "Perceived barriers for first-generation students: Reforms to level the terrain," *International Journal of Educational Reform*, vol. 21, pp. 238–252, Oct. 2012.

[5] E. M. Allensworth, C. A. Farrington, M. F. Gordon, D. W. Johnson, K. Klein, B. McDaniel, and J. Nagaoka, "Supporting social, emotional, & academic development: Research implications for educators," tech. rep., University of Chicago Consortium on School Research, Chicago, IL, Oct. 2018.

[6] W. E. Becker, "The educational process and student achievement given uncertainty in measurement," *The American Economic Review*, vol. 72, no. 1, pp. 229–236, 1982.

[7] P. Gomis-Porqueras and J. A. Rodrigues-Neto, "Teaching technologies, attendance, learning and the optimal level of access to online materials," *Economic Modelling*, vol. 73, pp. 329–342, 2018.

[8] R. J. Longhurst, "Why aren't they here? student absenteeism in a further education college," *Journal of Further and Higher Education*, vol. 23, no. 1, pp. 61–80, 1999.

[9] P. W. Mwangi, *Class attendance monitoring system using NFC technology*. PhD thesis, Strathmore University, Apr. 2018.

[10] F. Masalha and N. Hirzallah, "A students attendance system using QR code," *International Journal of Advanced Computer Science and Applications*, vol. 5, no. 3, pp. 75–79, 2014.

[11] S. Rao and K. J. Satoa, "An attendance monitoring system using biometrics authentication," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, no. 4, pp. 379–383, 2013.

[12] A. A. Kumbhar, K. S. Wanjara, D. H. Trivedi, A. U. Khairatkar, and D. Sharma, "Automated attendance monitoring system using android platform," *International Journal of Current Engineering and Technology*, vol. 4, no. 2, pp. 1096–1099, 2014.

[13] A. Bhise, R. Khichi, A. Korde3, and D. Lokare, "Attendance system using nfc technology with embedded camera on mobile device," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 4, no. 2, pp. 350–353, 2015.

[14] J. F. Cacho, "Using discord to improve student communication, engagement, and performance," in *Proc. of Best teaching practices expo 2020*, (Las Vegas NV), UNLV Office of Faculty Affairs, Jan. 2020.

[15] J. I. Olszewska, "The virtual classroom: A new cyber physical system," in *IEEE 19th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, pp. 187–192, IEEE, 2021.

[16] L. Olson, "How can learning management systems be used effectively to improve student engagement?," tech. rep., University of Washington Bothell Center on Reinventing Public Education, Bothell, WA, Jan. 2021.

[17] M. Vladoiu and Z. Constantinescu, "Learning during covid-19 pandemic: Online education community, based on discord," in *19th RoEduNet Conference: Networking in Education and Research (RoEduNet)*, pp. 1–6, IEEE, 2020.

[18] V. Kruglyk, D. Bukreiev, P. Chornyi, E. Kupchak, and A. Sender, "Discord platform as an online learning environment for emergencies," *Ukrainian Journal of Educational Studies and Information Technology*, vol. 8, no. 2, pp. 13–28, 2020.

[19] C. Lebeuf, M.-A. Storey, and A. Zagalsky, "Software bots," *IEEE Software*, vol. 35, no. 1, pp. 18–23, 2017.

[20] M. Espagnet, "Take attendance on discord." suivix.xyz/en. Accessed on 18-May-2021.

[21] discordamtbot, "AmtBot." www.facebook.com/discordamtbot/?ref=page_internal. Accessed on 18-May-2021.

[22] Integromat, "Attendance GIRITON, Discord integrations." www.integromat.com/en/integrations/discord/giriton. Accessed on 18-May-2021.

[23] "Apollo: Discord events made easy." apollo.fyi. Accessed on 18-May-2021.

[24] Nelly, "Discord Transparency Report: July – Dec 2020." blog.discord.com/discord-transparency-report-july-dec-2020-34087f9f45fb, Apr. 2021. Accessed on 10-June-2021.

[25] "Discord." discord.com. Accessed on 10-June-2021.

[26] Lucas, "Python: Making a Discord Bot (Rewrite / v1.x)." www.youtube.com/channel/UCR-zOCvDCayyYy1flR5qaAg/playlists, Apr. 2019. Accessed on 7-Jan-2021.