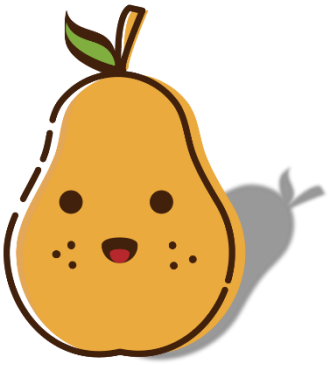
CST8333

Assignment 02



# Pear

A Discord User Pairing Bot

Kaitlyn Gatineau

July 19th 2022

# Table of Contents

# Introduction

As the community manager of a Discord group called the *Femmes and Thems in Tech*, I oversee, engage with and host events for thousands of spectacular women and nonbinary users who utilize our platform daily. Every week, it is my duty to manually pair individuals up who opt in to "break the ice" for those participating and allow for users to meet someone new every week.

Pear is a tool that was not only created for the sake of a school project but serves as the first iteration of a solution to a real-life problem in our community. Manually pairing users worked on a small scale, but the process needed automation and the type of tool necessary to make this happen did not exist at the time of writing for the Discord platform.

Automating the pairing process using a Discord bot not only saves time for the moderation team, but it allows for accuracy and consistency for when pairings would be posted.

While the bot will be originally utilized to pair people up for networking, it is intended to work as a pairing system for any scenario a Discord admin may deem necessary. Our goal is to create a versatile pairing bot that pairs users for any social/logistical need.

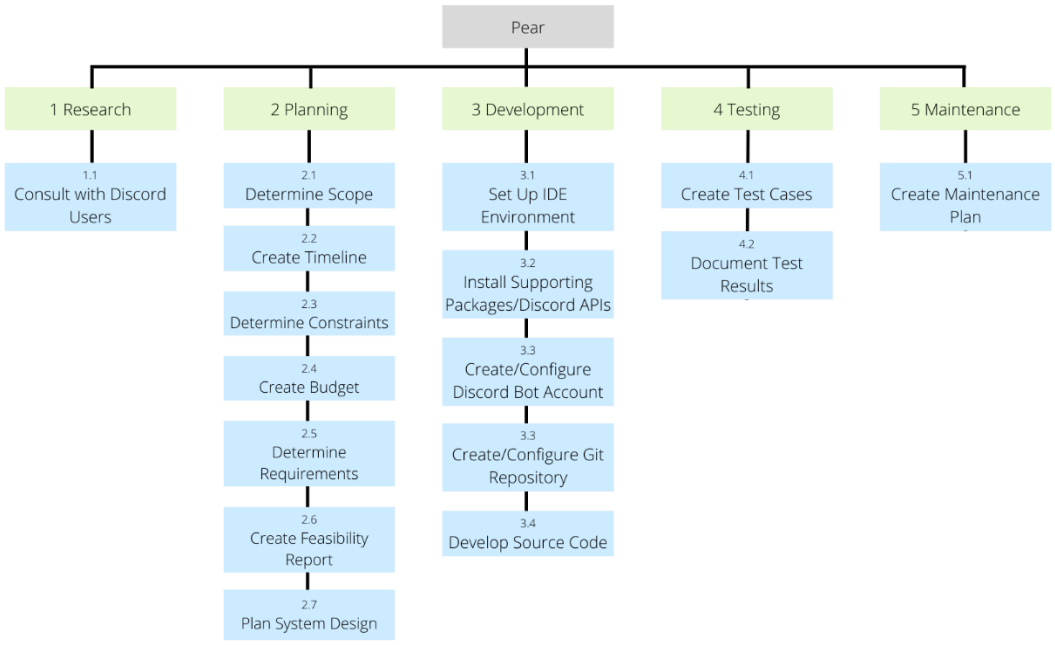
## Approach

This project (in respect to the version being created for the purpose of this course) is being conducted using a modified waterfall lifecycle. The project will be done in three phases, each in respect to the assignment/presentation due at the end of each phase.

The measurement criteria for each phase will be the checklists/rubrics associated with the assignment/presentation. 60% of the source code must be completed by the end of phase two, and 100% completed by the end of phase three. The phase will be deemed successful should the majority of checklist/rubric goals for that phase be met.

## Work Breakdown Structure

Below is a first draft of the WBS for this project. This is subject to change should new needs arise.




## Version Control

Version control will be maintained using a Git repository. I plan to utilize the GitHub platform to maintain the code.

## Objective

The overall objective is to create a function bot that works as intended by pairing users upon request. This main objective can be broken down into the following:

**Table 1: Objectives and Business Outcomes**

| No. | OBJECTIVE | BUSINESS OUTCOME | MEASUREMENT CRITERIA |
|---|---|---|---|
| 1 | Bot only pairs users who opt into the pairing system versus all users/online users in the Discord. | Users who wish to be paired are paired, and those who do not wish to be paired are not bothered by the bot. | Users who are opted in are pinged (@ tagged); users who are not are left alone. |
| 2 | Bot is only responsive to commands made by Administrator ranks. | Users can enjoy the pairing system without unnecessary/duplicate pairings triggered by unauthorized users. | User with Administrator rank can trigger the bot via command;<br>Bot does not respond to general user commands. |

| No. | OBJECTIVE | BUSINESS OUTCOME | MEASUREMENT CRITERIA |
|---|---|---|---|
| | | Administrators can rely on the bot to not populate the server with unwanted posts. | |
| 3 | Bot activity is limited to the designated channel. | Users know where to look for the postings with ease. | Bot only triggers from commands and posts in a designated channel. |
| 4 | Bot fetches user IDs from one designated post only. | Users who reacted to unrelated posts in the Discord are not accidently considered in the pairings. | Bot only pairs users who react to a single post.<br><br>Users who react to other posts are not included. |

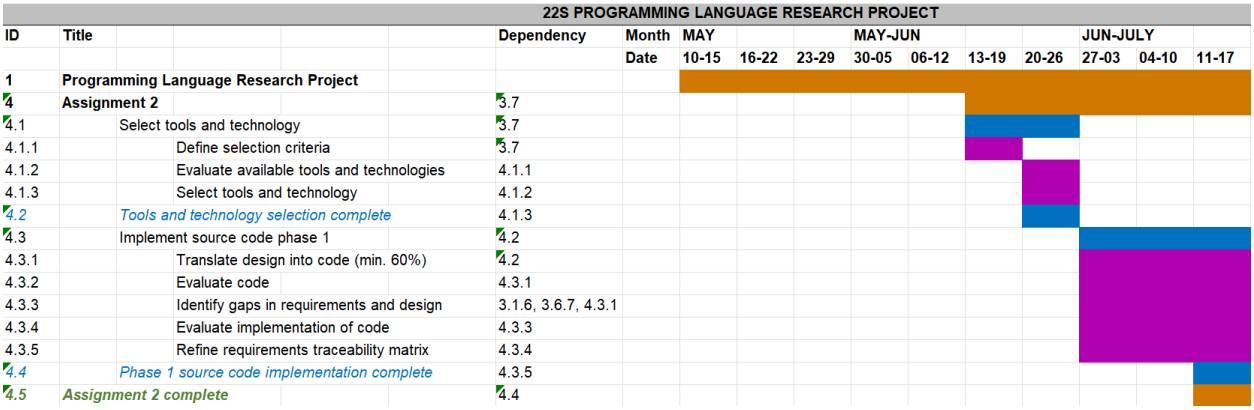
## Scope

This section describes the features of the bot that will be implemented for the scope of this course. Features beyond the scope are listed as “Activities Out of Scope”.

**Table 2: Project Scope**

| ACTIVITIES IN SCOPE | ACTIVITIES OUT OF SCOPE |
|---|---|
| 1. Assignment 1 | 1. Apply for business financing |
| 2. Fetch only user IDs/usernames of users who “reacted” to a designated post to be entered into the pairing pool to ensure only those who opt in are paired. | 2. Store pairing data in a database schema |
| 3. Randomize the collection order to ensure users are not paired by their place on the “reaction” list. | 3. Ensure no two users are paired multiple weeks in a row |
| 4. Ensure the post can only be triggered by an “Administrator” rank of the Discord server. | 4. Create logic for the bot to post at a designated time without the need to be manually triggered by command. |
| 5. Create a mock Discord server for development with minimum six users for testing. | 5. Purchase a hosting service to maintain high availability uptime without the need for an admin to host the bot on a personal machine |
| 6. Create a GitHub repository to maintain the code | |
| 7. Remove reactions by users who have left the Discord. | |

## Timeline



| 22S PROGRAMMING LANGUAGE RESEARCH PROJECT | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ID | Title | | | Dependency | Month | MAY | | | | MAY-JUN | | | | JUN-JULY | | | |
| | | | | | Date | 10-15 | 16-22 | 23-29 | 30-05 | 06-12 | 13-19 | 20-26 | 27-03 | 04-10 | 11-17 | | |
| 1 | **Programming Language Research Project** | | | | | | | | | | | | | | | | |
| 4 | **Assignment 2** | | | 3.7 | | | | | | | | | | | | | |
| 4.1 | | Select tools and technology | | 3.7 | | | | | | | | | | | | | |
| 4.1.1 | | | Define selection criteria | 3.7 | | | | | | | | | | | | | |
| 4.1.2 | | | Evaluate available tools and technologies | 4.1.1 | | | | | | | | | | | | | |
| 4.1.3 | | | Select tools and technology | 4.1.2 | | | | | | | | | | | | | |
| *4.2* | | *Tools and technology selection complete* | | 4.1.3 | | | | | | | | | | | | | |
| 4.3 | | Implement source code phase 1 | | 4.2 | | | | | | | | | | | | | |
| 4.3.1 | | | Translate design into code (min. 60%) | 4.2 | | | | | | | | | | | | | |
| 4.3.2 | | | Evaluate code | 4.3.1 | | | | | | | | | | | | | |
| 4.3.3 | | | Identify gaps in requirements and design | 3.1.6, 3.6.7, 4.3.1 | | | | | | | | | | | | | |
| 4.3.4 | | | Evaluate implementation of code | 4.3.3 | | | | | | | | | | | | | |
| 4.3.5 | | | Refine requirements traceability matrix | 4.3.4 | | | | | | | | | | | | | |
| *4.4* | | *Phase 1 source code implementation complete* | | 4.3.5 | | | | | | | | | | | | | |
| *4.5* | ***Assignment 2 complete*** | | | 4.4 | | | | | | | | | | | | | |

Clicking the icon below will open the full version in Excel. Please note, this Gantt Chart was created using the template provided, and is cited as a source in the *References*.

CST8333 Gantt Chart - Kaitlyn Gatin

## Milestones and Deliverables

This section lists major milestones/deadlines required by this project.

**Table 4: Project Milestones and Deliverables**

| MILESTONE | DATE | DELIVERABLES |
|---|---|---|
| 1. Assignment 1 - Complete | June 9 2022 | Written report and slide presentation |
| 2. Assignment 2 - Complete | July 14 2022 | Written report, slide presentation and 60% of source code |
| 3. Assignment 3 - Complete | August 11 2022 | Written report, slide presentation and 100% of source code |

## Risks

This section mentions the risks that are associated with this project.

**Table 5: Project Risks** (example)

| No. | RISK DESCRIPTION | PROBABILITY (H/M/L) | IMPACT (H/M/L) | MITIGATION |
|---|---|---|---|---|
| 1. | Schedule slippage | M | H | Track project scope and timeline |

| No. | RISK DESCRIPTION | PROBABILITY (H/M/L) | IMPACT (H/M/L) | MITIGATION |
|---|---|---|---|---|
| 2. | Discord server outage | L | M | Wait until Discord resolves the issue(s) |
| 3. | Hardware Malfunction | L | H | Back up project to external source; have a spare machine ready |
| 4. | Lack of Knowledge | M | M | Reach out for help from the internet/experienced developers |

## Assumptions

**Table 6: Project Assumptions**

| No. | THE FOLLOWING IS ASSUMNED |
|---|---|
| 1. | Fundamentals of new programming language will be learned and put to use, timely, to complete project |
| 2. | Bot will only run in a single server, via a single host machine at any given time |
| 3. | Tracking previous pairings is not necessary for the scope of this project |

## Technical Constraints

**Table 7: Technical Constraints**

| No. | TECHNICAL CONSTRAINTS |
|---|---|
| 1. | Program requires a user to execute on a host computer for it to function |

## Budget

Below is the proposed budget for the project. The project is measured in “Level of Effort” instead of currency as no money will be spent on this project.

Please note, the Project Budget was created using the template provided, and is cited as a source in the *References*.

**Table 8: Project Budget**

| ID | Title | Level of Effort (Days) |
|---|---|---|
| **1** | **Programming Language Research Project** | 57.20 |
| **2** | **Planning** | 5.00 |
| 2.1 | Review project approach | 1.00 |

| | | | |
|---|---|---|---|
| 2.2 | | Define project scope | 1.00 |
| 2.3 | | Identify project risks, assumptions and constraints | 1.00 |
| 2.4 | | Draft project budget | 1.00 |
| 2.5 | | Document project schedule (timeline) | 1.00 |
| ***2.6*** | ***Planning Complete*** | | 0.00 |
| **3** | **Assignment 1** | | 12.50 |
| 3.1 | | Gather requirements | 5.00 |
| 3.1.1 | | Define requirements gathering process | 1.00 |
| 3.1.2 | | Develop use cases | 1.00 |
| 3.1.3 | | Describe data requirements | 0.50 |
| 3.1.4 | | Document technical requirements | 0.50 |
| 3.1.5 | | Illustrate user interface | 1.00 |
| 3.1.6 | | Create requirements traceability matrix | 1.00 |
| *3.2* | | *Requirements gathering complete* | 0.00 |
| 3.3 | | Complete feasibility study | 4.00 |
| 3.3.1 | | Define evaluation criteria | 1.00 |
| 3.3.2 | | Analyze development options | 1.00 |
| 3.3.3 | | Complete cost/benefit analysis | 1.00 |
| 3.3.4 | | Document recommended solution | 1.00 |
| *3.4* | | *Feasibility study complete* | 0.00 |
| 3.5 | | Provide design description | 3.50 |
| 3.5.1 | | Create high level design | 1.00 |
| 3.5.2 | | Develop prototype | 1.00 |
| 3.5.3 | | Refine requirements traceability matrix | 0.50 |
| 3.5.4 | | Draft deployment plan | 0.50 |
| 3.5.5 | | Draft support plan | 0.50 |
| *3.6* | | *Design description complete* | 0.00 |
| ***3.7*** | ***Assignment 1 complete*** | | 0.00 |
| **4** | **Assignment 2** | | 19.00 |
| 4.1 | | Select tools and technology | 7.00 |
| 4.1.1 | | Define selection criteria | 3.00 |
| 4.1.2 | | Evaluate available tools and technologies | 3.00 |
| 4.1.3 | | Select tools and technology | 1.00 |
| *4.2* | | *Tools and technology selection complete* | 0.00 |
| 4.3 | | Implement source code phase 1 | 12.00 |
| 4.3.1 | | Translate design into code (min. 60%) | 3.00 |
| 4.3.2 | | Evaluate code | 2.00 |
| 4.3.3 | | Identify gaps in requirements and design | 2.00 |
| 4.3.4 | | Evaluate implementation of code | 3.00 |
| 4.3.5 | | Refine requirements traceability matrix | 2.00 |
| *4.4* | | *Phase 1 source code implementation complete* | 0.00 |
| ***4.5*** | ***Assignment 2 complete*** | | 0.00 |
| **5** | **Assignment 3** | | 20.70 |
| 5.1 | | Implement source code phase 2 | 5.00 |
| 5.1.1 | | Translate design into code (up to 100%) | 1.00 |
| 5.1.2 | | Evaluate code | 1.00 |
| 5.1.3 | | Identify gaps in requirements and design | 1.00 |
| 5.1.4 | | Evaluate implementation of code | 1.00 |
| 5.1.5 | | Refine requirements traceability matrix | 1.00 |
| *5.2* | | *Phase 2 source code implementation complete* | 0.00 |
| 5.3 | | Complete testing | 5.20 |
| 5.3.1 | | Refine test plan (traceability matrix) | 0.40 |

| | | | |
|---|---|---|---|
| 5.3.2 | | Prepare test tracking tools | 0.40 |
| 5.3.3 | | Create test environment | 0.40 |
| 5.3.4 | | Complete system testing | 0.40 |
| 5.3.5 | | Implement build procedures | 0.40 |
| 5.3.6 | | Load data into test environment | 0.40 |
| 5.3.7 | | Develop test scripts | 0.40 |
| 5.3.8 | | Execute test scripts | 0.40 |
| 5.3.9 | | Correct defects (if any) | 0.40 |
| 5.3.10 | | Document test results | 0.40 |
| 5.3.11 | | Obtain approval of testing from Facilitator | 0.40 |
| 5.3.12 | | Update support documents (if required) | 0.40 |
| 5.3.13 | | Store test data | 0.40 |
| *5.4* | *Testing complete* | | 0.00 |
| 5.5 | Complete deployment | | 6.00 |
| 5.5.1 | | Refine deployment plan | 1.00 |
| 5.5.2 | | Prepare production environment | 1.00 |
| 5.5.3 | | Complete data conversion activities | 1.00 |
| 5.5.4 | | Complete operation and maintenance prep | 1.00 |
| 5.5.5 | | Validate code functionality | 1.00 |
| 5.5.6 | | Deploy code | 1.00 |
| *5.6* | *Deployment complete* | | 0.00 |
| 5.7 | Complete presentation | | 4.50 |
| 5.7.1 | | Select slide presentation template | 1.50 |
| 5.7.2 | | Develop presentation content | 1.50 |
| 5.7.3 | | Submit presentation to Facilitator | 1.50 |
| *5.8* | *Presentation complete* | | 0.00 |
| ***5.9*** | ***Assignment 3 complete*** | | 0.00 |
| ***5.10*** | ***Programming Language Research Project complete*** | | 0.00 |

# Status

## Introduction

This report gives a progress update on the accomplishments, goals, roadblocks and lessons learned over the span of Assignment 2.

## Accomplishments

### Selecting the Tools and Technology

The evaluation and selection of available tools and technologies, along with the selection criteria were drafted the weekend of June 18th. This aligns with the proposed schedule stated in the gantt chart. The draft consisted of brief bullet points to structure content that would be refined and elaborated on during the report structuring at the end of the phase.

### Implementing the Source Code – Phase 1

#### *Translating Design to Code*

Translating the design into Python code took a lot less time than originally anticipated. Transitioning from a boilerplate language like Java to one less verbose like Python ended up being a relatively pleasant experience that resulted in quick development during the initial phase. The discord.py API carried a lot of the weight regarding function, and offered several methods that reduced the amount of Python code and configuration needed by the developer.

The timeline for development was slated for June 27 to July 17. Around 30% of the code was completed before the 27th to show UI examples in screenshots for Assignment 1, and the remaining 70% of the 60+% required by the due date was completed July 13/14, as the last week of June and first week of July were consumed by other commitments and time constraints.

#### *Evaluating Code*

The evaluation of code was done in iterations, with the final pass before the submission deadline on July 15th.

The first evaluation pass was done after the first iteration of the “create” command was completed. Another pass was completed after the “pair” function was created and when the lines of code to opt into the member caching were added.

The code was formatted, commented (including removal of redundant comments used for debugging purposes), and refactored down to more stylistic shorthand notations, etc. after each pass. Python, by nature, is a language dependent on whitespace for function, so the amount of stylistic formatting is less than a language like Java, which does not read whitespace when compiling.

#### *Identifying Gaps in Requirements/Design/ Evaluating Implementation*

Requirements have been checked off as they have been completed. All requirements intended for the first phase of development have been completed, along with a few requirements scheduled to be completed next phase.

# Goals

## Time Management

Managing my time efficiently and staying on schedule was an important goal for this phase of the project that was only partially met.

While the development of the source code took less overall time than anticipated, the project fell behind due to a combination of personal time constraints and unforeseen events.

I requested an extension ahead of time knowing the circumstances would affect my ability to turn this project in on time. This allowed me time to address the other projects that needed attention and catch up to schedule.

Overall, this goal was not completely met due to my lack of foresight and ability to plan "cushion" room for disasters. Besides the implementation of source code, this assignment had very little wiggle room as I struggled to manage my time between multiple classes and other means. However, I did manage to complete most tasks within the end of their respective time frames and get back on track by finishing the report and presentation with an extension.

## Development – Phase I

The phase I development goals are as follows:

- Create the bot within the Discord development portal
- Generate the bot token, create a .env file to store token data for security
- Write logic to trigger a "create" post, using the prefix "!"
    - The "!create" command must accept two arguments, each contained in quotation marks and separated by a space. The first argument determines the title of the embed, the second argument determines the content/description.
    - The "!create" command must create a .txt file to save the message ID of the message so the "!pair" command knows which message to fetch the user list from.
- Write logic to trigger a "pair" post, using the prefix "!"
    - The "!pair" command must fetch a list users who reacted to the post, mention them in the embed post and present the users in groups of two
    - The "!pair" command must also randomize the list every time it is triggered
    - The "!pair" command must delete reactions from users no longer in the Discord server/guild

All goals listed above have been completed this phase and within the allotted time frame for development. Below are the goals set to be submitted with the remainder of the source code for Assignment 3:

- Write logic to ensure only an Admin ranked member can trigger the bot
- Write logic to ensure only one specific type of reaction is read from the list
- Make the text in the pair post variables instead of hard coded strings for user flexibility
- Move the hard coded channel/guild IDs into the .env or .txt files
- Ensure the logic complies with the Discord terms of service and other legal obligations/compliances.

# Roadblocks

## Time Management

### *Roger's Outage*

As mentioned previously, an unforeseen outage by Canada's largest telecom company, Roger's caused havoc on many systems across the country, including Algonquin College's learning platform and email services.

The outage revoked access to reports and resources hosted via Microsoft's SharePoint documents connected through the school's email and left students and staff scrambling as no one was able to communicate via email and access essential documentation stored online.

This outage caused a "domino effect" for my project timeline, delaying a presentation and report deadline my team for another project had scheduled that day, and limited the ability for the team to work for the remainder of the weekend.

Thankfully, all services provided via Roger's has been restored and this should not be a roadblock in the future for this project.

### *Personal Constraints*

Personal constraints have consumed more time than I would like and managing my time has been very difficult. Personal constraints will remain as a roadblock for the remainder of the semester to some varying degree.

## Development

Below are the roadblocks related to development.

### *Message ID Storage*

Initially, no data was to be saved for the scope of this project. After beginning development, I realized that the message ID of the reaction host post will need to be documented to ensure that the "pair" command fetches reactions from the correct post. I implemented logic to create/overwrite a .txt file called "root_message.txt" that stores the ID of the message generated via the "!create" command.

This variable is stored via an external file instead of a variable because powering off/stopping the bot will remove the variable data. Hard-coding the message ID would require the user to edit the code every time the "!create" command is executed with an updated message ID.

### *Discord API Update*

The Discord API recently pushed an update requiring bots to "opt in" to caching member data. This means regardless of the language used to create a Discord bot, the bot must have lines of code stating the Intent and setting the value to "true". In Python, this is done using the code below:

```
intents = discord.Intents.default()
intents.members = True

bot = commands.Bot(command_prefix="!", intents=intents)
```

This caused a bit of a headache as many troubleshooting websites with out of date answers like StackOverflow omitted this detail, and I was unable to fetch the list of members without this vital information. More can be read about the Discord API Privileged Intents here.

### *Learning Curve*

Learning a new programming language always comes with a learning curve to a certain degree. While my transition from Java to Python has been easier than expected, it still came with its faults. Taking the time to translate what I know in Java to Python syntax takes time.

Also, while both languages can produce a nearly identical bot, Python and Java have different features. For example, the 'do-while' loop commonly used in Java programs does not exist in Python. Until Python 3.10, the 'switch-case' statement did not exist either. Other methods, like using flags to trigger/stop while loops and using 'if-elif-else' statements had to be used instead of these concepts.

In Python, declaring variable types isn't necessary. Colons are used more frequently than semi colons, and whitespace *does* matter. When tallying up the little changes between what I knew previously and what I am learning now, the time invested in research adds up and will be considered a minor roadblock (or better yet, a "traffic jam") for the remainder of this project.

## Lessons Learned

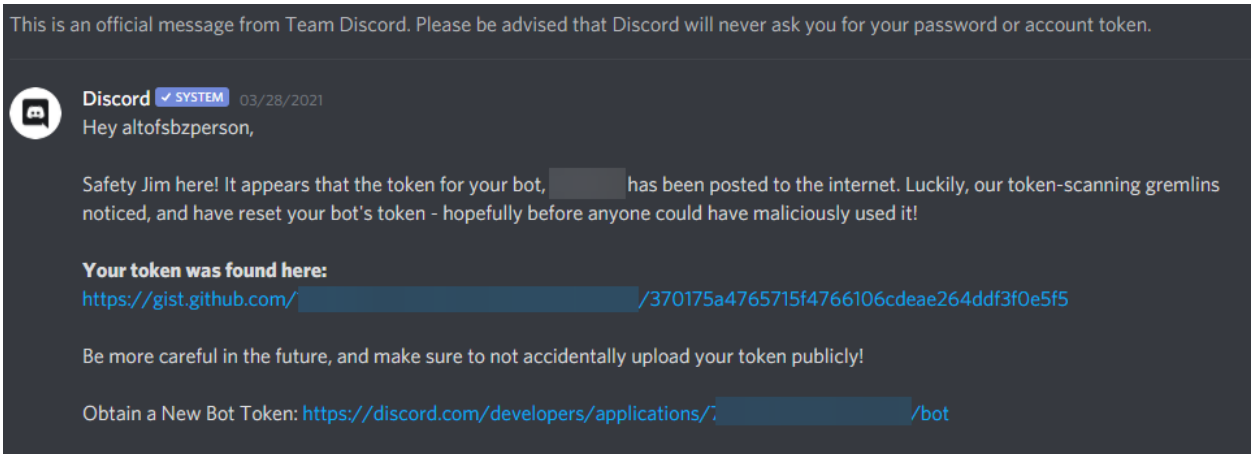
### Time Management

Time management has always been a struggle of mine, but over time, I have become better at managing my time and staying on schedule. A lesson I have learned is to make room for unforeseen circumstances and disasters, even if it feels redundant or unnecessary.

### Token Data Breech

When creating the Git repository and posting it to GitHub, I made a critical mistake of pushing the .env file, which contains the token value of the bot, to the public repository.

Discord automatically flagged this, notified me via private message and invalidated the token. Had Discord not caught this error, the bot could have been compromised.



It is important to ensure the .env file is not accidently pushed for the world to see, as it contains sensitive information.

# Tools and Technology

## Introduction

In today's tech-driven society, many options are available to choose from when deciding the tools and technology to utilize for an application. Many programming languages with different target purposes exist to meet the needs of developers of all programming backgrounds.

While many tools are available, certain ones are more favorable than others for the scope of this project. Unlike mobile development versus web development, or iOS versus Android, there is no "one" popular language Discord bots are written in. Discord bots exist and are commonly written in languages from Java, using the *JDA* or *Discord4J* APIs, to Python with the *discord.py* API and JavaScript and its respective Discord API called *discord.js.*

Due to a technical requirement eliminating Java from the pool of potential languages,

Cost, user efficiency, availability, and user experience are the main selection criteria chosen for selecting the right toolset for this project.

## Evaluation

### Cost

The cost is determined by the amount of money a tool would require. Since the project has a relatively narrow scope and no payment/compensation is being gathered for the finished product, no money will be spent acquiring tools for this project. All tools utilized will be either free to use or were purchased for other means before this project began.

### User Efficiency

For this project, efficiency refers to how efficient the developer is when using the tool (regarding time and learning curve), and less about the level of efficiency impact the tool has on the application itself.

Should the scope be broadened in the future for the application to be deployed on a larger scale, the efficiency level of the application itself and how a tool could affect that must also be considered when deciding which toolset to use for development.

### Dependability

The dependability is measured by the application's ability to run without the suggested tool. If a tool is not critical for development and/or operation, the higher the likelihood of it being omitted for the small scope of this project.

## Selection

### Programming Language

The programming language selected for this project is Python. I selected Python to not only meet the needs of the assignment criteria, but to add another programming language to my portfolio. I have

created many Java projects and have a project using React JS I am completing in parallel to this one, so this project was a perfect opportunity to branch out and learn a language I have never used before.

## Hardware

The project is being written via and hosted on one machine, an HP Omen 15 inch laptop/16GB RAM/Core i7 CPU.

## Software

### *PyCharm IDE*

The IDE selected for this project is PyCharm IDE by JetBrains. Typically, JetBrains products are not free to use beyond the basic community edition, but I have acquired a student license for JetBrains products and will be able to utilize this software without any additional cost.

### *IDE Plugins*

Along with PyCharm IDE, I am utilizing a few plugins to help with development. They are as follows:

- EnvFile
- Highlight Bracket Pair
- Discord Integration
- Git
- GitHub
- GitHub Desktop

GitHub Desktop is a desktop application used to manage GitHub repositories.

### *Microsoft Word*

While this may seem redundant to list, MS Word is essential for writing documentation to correspond with the project.

## API

The API used to serve as the bridge from the Python code to Discord is a library called discord.py.

## Documentation

Documentation is an important tool for development, especially if developing with a specific language or library for the first time. Below is a list of documentation and other resources referenced for this project:

- Python documentation
- Discord API documentation
- StackOverflow and other online resources
- Algonquin College module content

# Conclusion

This project has been a wonderful opportunity to research and learn by practice about tools I have never used for software development. I look forward to advancing to the next phase of the project with the toolkit I have assembled to set this project up for success.

## Requirements Traceability Matrix

| ID | NAME | DESCRIPTION | TEST CASE ID | SCENARIO | EXPECTED RESULT |
|---|---|---|---|---|---|
| B-01 | Create reaction collection post | Create the post to collect reactions using “!create! | | | |
| B-02 | Fetch reaction list from collection post | Gather users who selected the reactions when “!post” is triggered | | | |
| B-03 | Remove redundant reactions | Remove reactions of deleted users upon “!post” trigger | | | |
| B-04 | Randomize fetched list | Randomize the list of users upon “!post” trigger | | | |
| B-05 | Pair users | List the users in groups of two | | | |
| B-06 | Post pair results | Send an embed message back to the client with the pair results | | | |
| T-01 | Physical host | Running machine to host bot | | | |
| T-02 | User Interface | Create a UI that allows for user interaction | | | |
| T-03 | Input/Output – Creating Reaction Post | Ensure bot is triggered by the correct input and generates the correct output | | | |
| T-04 | Input/Output – Creating Pairing Post | Ensure bot is triggered by the correct input and generates the correct output | | | |
| T-05 | Name Format - Mention | Users in the pairing posts must be tagged properly using the @ notation | | | |
| C-01 | Discord Community Guidelines | Bot must comply with and not promote activity against Discord’s Community Guidelines | | | |
| C-02 | Discord Developer Policy | Bot must comply with the API constraints and policy of Discord’s Developer Policy | | | |
| C-03 | Discord Privacy Policy | Bot must not obtain sensitive user data without user consent as per the Discord Privacy Policy | | | |
| C-04 | Discord Terms of Service | Bot must not violate the general Discord Terms of Services | | | |
| C-05 | Discord Store Distribution Agreement for Developers | Bot must comply with the Discord Store Distribution Agreement for Developers | | | |
| S-01 | Hide Token Data | Ensure critical token data is stored in .env files away from the code | | | |

| | | | | | |
|---|---|---|---|---|---|
| S-02 | Granted Admin Role | Bot must be granted "Administration" privileges on Discord in order to function | | | |
| UC-01 | Create reaction host post | Bot posts embed message as a response to the command that accepts user reactions | | | |
| UC-02 | Pair users/post pair message | Bot posts embed message as a response to the command that pairs users who reacted in groups of two | | | |
| UC-03 | Delete user reactions no longer on server | Delete users who are unable to be paired (due to no longer being a member of the server) before pairing/posting. | | | |

# Source Code Implementation

## Introduction

The language selected for this project is Python. Python is a high-level programming language that is interpreted instead of compiled and has a dynamic range of syntax usages. Python is a very versatile language and is user friendly relative to many other verbose programming languages on the market. \

## Coding Checklist

| Activities | Completed |
|---|---|
| Translate the design into code that will satisfy requirements. | Yes/No |
| Identify inadequacies in the design and requirements. | Yes/No |
| Document further decisions and rationale. | Yes/No |
| Complete? Is everything that is in the requirements and design is implemented. | Yes/No |
| Consistent? Ensure there are no mismatched interfaces and consistent with the design. | Yes/No |
| Stylistic? Exhibits good programming style. | Yes/No |
| Understandable? The code should be constructed in a way that is easy to read, not necessarily easy to write. | Yes/No |
| Modifiable? If any changes need to be applied, Modify the code as per the changes required. | Yes/No |

| | |
|---|---|
| Is the code Confirmable, Verifiable and testable? You can tell when you've met the design and requirements. | Yes/No |
| Is the complete code submitted to facilitator? | Yes/No |
| Feedback taken from facilitator. | Yes/No |
| Update and submit the final project code and update project related document and presentation after completion the module. | Yes/No |

Submit Code

See the .txt file(s) attached in the ZIP file to access the source code.

# References

Algonquin College. (2020a). *CST8333 Budget Template*. Algonquin College.

Algonquin College. (2020b). *CST8333 Timeline Template*. Algonquin College.

Algonquin College. (2020c). *CST8333 Assignment 1 Outline*. Algonquin College.

Chi, E. (2020, August 10). *How to Make Discord Bot Commands in Python*. Better Programming; Medium. https://betterprogramming.pub/how-to-make-discord-bot-commands-in-python-2cae39cbfd55

Discord. (n.d.-a). *Discord Developer Portal — API Docs for Bots and Developers*. Discord Developer Portal. Retrieved June 8, 2022, from https://discord.com/developers/docs/intro

Discord. (n.d.-b). *Discord Store Distribution Agreement for Developers (Self-Service)*. Discord Developer Portal; Discord. Retrieved May 31, 2022, from https://discord.com/developers/docs/policies-and-agreements/store-distribution-agreement

Discord. (n.d.-c). *OAuth2*. Discord Developer Portal; Discord. Retrieved June 8, 2022, from https://discord.com/developers/docs/topics/oauth2

Discord. (n.d.-d). *Permissions*. Discord Developer Portal; Discord. Retrieved June 10, 2022, from https://discord.com/developers/docs/topics/permissions

Discord. (2020, July 1). *Discord Developer Policy*. Discord Developer Portal; Discord. https://discord.com/developers/docs/policies-and-agreements/developer-policy

Discord. (2022a, February 25). *Community Guidelines*. Discord. https://discord.com/guidelines

Discord. (2022b, February 25). *Privacy Policy*. Discord. https://discord.com/privacy

Discord. (2022c, February 25). *Terms of Service*. Discord. https://discord.com/terms

Duke, R. (2022). *What is a Work Breakdown Structure*. Workbreakdownstructure.com. https://www.workbreakdownstructure.com/

MiiaHash. (2021, May 3). Microservices architecture (discord bot). *MCMARKET*. https://www.mc-market.org/threads/662952/#post-4826533

Pngtree. (2021). meb style yellow cute pear clipart [Online]. In 阿Go (Ed.), *Pngtree*. https://pngtree.com/freepng/meb-style-yellow-cute-pear-clipart_5867760.html

Rapptz, D. (n.d.). *Cogs*. Discord.py. Retrieved June 9, 2022, from https://discordpy.readthedocs.io/en/stable/ext/commands/cogs.html

Rapptz, D. (2019, September 30). *Welcome to discord.py*. Discord.py. https://discordpy.readthedocs.io/en/stable/

Rapptz, D. (2022). *A primer to gateway intents*. A Primer to Gateway Intents. Retrieved July 19, 2022, from https://discordpy.readthedocs.io/en/latest/intents.html#privileged-intents

Seewald, D. (2022, April 11). *Send an Embed with a Discord Bot in Python*. Python in Plain English; Medium. https://python.plainenglish.io/send-an-embed-with-a-discord-bot-in-python-61d34c711046

Taneja, N. (2021, August 27). *Architecting discord bot the right way*. DEV Community. https://dev.to/itsnikhil/architecting-discord-bot-the-right-way-383e

The Python Software Foundation. (2022). *What is python? executive summary*. Python.org. Retrieved July 14, 2022, from https://www.python.org/doc/essays/blurb/

Warren, T. (2021, November 17). *Discord is quietly building an app empire of bots*. The Verge. https://www.theverge.com/2021/11/17/22787018/discord-bots-app-discovery-platform

## Appendix A: Module 6 Checklist

| Activity | Completed |
|---|---|
| What are the evolution trends of tools/technologies according to project scope and type? | Yes/No |
| What are the different technologies according to your project complexity? | Yes/No |
| Analysis of project development speed in order to select the right technology. | Yes/No |
| Preparation of available toolsets for your project. | Yes/No |
| Think about using chosen toolsets for out of the box solution for your project. | Yes/No<br><br>Out of the box solution in my example wouldn't involve much development. The API used carries most of the weight as-is. |
| Clear idea about how to load your requirements by using chosen technology. | Yes/No |
| The security aspects of using any tools in your project. | Yes/No |
| Criteria for tool maintenance and support. | Yes/No |
| Advantages and Disadvantages of choosing a particular technology. | Yes/No |
| Submit all the information along with the technology platform chosen for the project to the facilitator. | Yes/No |
| Feedback taken from facilitator about selected tools. | Yes/No |
| Prepare documentation regarding all the information of selected tools and technology and submitted to the facilitator within the given time. | Yes/No |
| Facilitator approved the submission. | Yes/No |

## Appendix B: Module 7 Checklist

| Activity | Completed |
|---|---|
| Write a project scope statement. | Yes/No |
| Create a work breakdown structure. | Yes/No |
| Break your project into smaller tasks. | Yes/No |
| Determine project dependencies. | Yes/No |
| Determine total time needed for each task. | Yes/No |
| Identify resource availability. | Yes/No |
| Identify important milestones. | Yes/No |
| Build your project management timeline. | Yes/No |
| Submit your project timeline to the facilitator for verification. | Yes/No |
| Update your timeline as per the changes suggested by the facilitator and submit the final project timeline. | Yes/No |
| Add Timeline information in your project report and brief information in presentation. Submit all the documents to the facilitator. | Yes/No |

## Appendix C: Module 8 Checklist

| Activities | Completed |
|---|---|
| Translate the design into code that will satisfy requirements. | Yes/No |
| Identify inadequacies in the design and requirements. | Yes/No |
| Document further decisions and rationale. | Yes/No |
| Complete? Is everything that is in the requirements and design is implemented. | Yes/No |
| Consistent? Ensure there are no mismatched interfaces and consistent with the design. | Yes/No |
| Stylistic? Exhibits good programming style. | Yes/No |
| Understandable? The code should be constructed in a way that is easy to read, not necessarily easy to write. | Yes/No |
| Modifiable? If any changes need to be applied, Modify the code as per the changes required. | Yes/No |
| Is the code Confirmable, Verifiable and testable? You can tell when you've met the design and requirements. | Yes/No |
| Is the complete code submitted to facilitator? | Yes/No |
| Feedback taken from facilitator. | Yes/No |
| Update and submit the final project code and update project related document and presentation after completion the module. | Yes/No |

## Appendix D: Module 9 Checklist

| Activities | Completed |
|---|---|
| Have you added the Project Title, approved by facilitator, in the Report? | Yes/No |
| Have you added brief Description about project in your Report? | Yes/No |
| Have you mentioned about tools and technology you are going to use for this project in Progress Report? | Yes/No |
| Have you added Project Milestones [Project development phases] in Progress Report? | Yes/No |
| Have you added the Project Status in Progress Report? | Yes/No |
| Have you added the brief explanation about your working project? | Yes/No |
| Have you added conclusion and future work in your project progress report? | Yes/No |
| Have you submitted your Project Progress Report to your facilitator? | Yes/No |
| Facilitator approved your project progress report and allowed you to start working on next phase. | Yes/No |

The facilitator approved the use of Python as a programming language and the use of the discord.py API. However, I did not seek permission to use a specific IDE and other tools as I felt as if this was unnecessary. The API itself, next to the IDE, is the largest tool in the toolset for the type of project I am creating.

# Appendix E: Assignment Grading Rubric

| Criteria | Excellent<br>80-100% | Good<br>50-79% | Requires Improvement<br><50% | Points |
|---|---|---|---|---|
| **Assignment Quality** | All information offered is accurate<br><br>All views are clearly expressed and well explained<br><br>Contains original ideas, connections, or applications | Most information offered is accurate<br><br>Most views are clearly expressed and explained<br><br>Contains mainly original ideas, connections, or applications | Some or no accurate information offered<br><br>Views are rarely or never clear and require further explanation<br><br>Many non- original ideas, or unclear connections or applications | **/15** |
| **Assignment Knowledge and Skills Demonstration** | Clear, concise synthesis of course content to demonstrate understanding of topic<br><br>All ideas are clearly developed, organized logically, and connected with effective transitions<br><br>Explores ideas, supports points fully using a balance of evidence, and uses effective reasoning to make useful distinctions<br><br>All relevant course and topic links are made | Evidence of some synthesis of course content to demonstrate understanding of topic<br><br>Some unified and coherent ideas are developed with effective transitions<br><br>Supports most ideas with effective examples, and/or references, and details, makes key distinctions<br><br>Most relevant course and topic links are made | Lack of evidence or weakness in the synthesis of course content to demonstrate understanding of topic<br><br>Develops and organizes ideas that are not necessarily connected. Some ideas seem illogical and/or unrelated<br><br>Presents ideas in general terms; most ideas are inconsistent/unsupported, and reasoning is flawed or unclear<br><br>Some or no relevant course and topic links are made | **/15** |
| **Assignment Structure** | Formatted as per assignment details<br><br>Structure and format enhance delivery of the information | Formatted as per assignment details in most components<br><br>Structure and format fits well with the delivery of the information | Formatting has not been followed<br><br>Structure and format are unclear and impedes delivery of the information | **/5** |

| Criteria | Excellent<br>80-100% | Good<br>50-79% | Requires Improvement<br><50% | Points |
|---|---|---|---|---|
| | Clear language is used which leads to easy readability<br><br>Correct grammar and spelling are consistently used | Mostly clear language is used with minor readability issues<br><br>Few or no spelling and/or grammatical errors | Language used is often unclear which impedes readability<br><br>Many spelling and grammatical errors | |
| **Total Points** | | | | **/35** |