Discord Bot Integration

for Replit Teams for Education

DESIGN DOCUMENT

15
Dr. Zambreno
Advisers
Patrick Demers, Kyle Rooney, Cole Mullenbach, Sophie Waterman Hines, and Kristen Nathan
sddec23-15@iastate.edu
http://sddec23-15.sd.ece.iastate.edu/

Revised: April 20th 2023

Executive Summary

Development Standards & Practices Used

- ISO/IEC/IEEE International Standard - Software and systems engineering --Software testing --Part 1: General concepts
- ISO/IEC/IEEE International Standard - Systems and software engineering -- Design and development of information for users
- ISO/IEC/IEEE International Standard - Software and systems engineering -- Software testing -- Part 5: Keyword-Driven Testing

Summary of Requirements

- The Discord Bot will be configurable on a class and assignment basis.
- The Discord Bot will be able to download up-to-date student code from Replit.
- The teaching staff will have an easy-to-use interface for updating the bot.
- The Discord Bot will be able to converse with multiple students at a time.
- The Discord Bot will execute tests on the student code.
- The Discord Bot will provide helpful hints to students based on compiler error messages.
- The project must be completed by the end of Iowa State University's fall 2023 semester.
- The project deployment infrastructure must be capable of running on Iowa State University Virtual Machines.
- The messages sent by the Discord Bot will support button reactions for positive/negative feedback.
- The Discord Bot will respond to a student within five seconds.
- The Discord Bot will be capable of escalating the student's request to a professor if its first attempt receives negative feedback.
- All snippets of code sent by the bot as examples will be written with correct formatting and spacing to improve readability.
- All bot responses will be easy to understand and formatted in an appealing and organized manner.

Applicable Courses from Iowa State University Curriculum

- Com S 227
- Com s 228
- Com S 309
- DS 201

- SE 319
- Com S 363
- SE 317
- SE 339
- SE 421

New Skills/Knowledge acquired that was not taught in courses

- Python
- Docker
- Discord API
- Pycord
- Discord.py
- Webscraping

Table of Contents

List of figures/tables/symbols/definitions (This should be the similar to the project plan)

1 Team

1.1 Team Members

- Patrick Demers
- Kyle Rooney
- Cole Mullenbach
- Kristen Nathan
- Sophie Waterman Hines

1.2 Required Skill Sets for Your Project

- Software engineering: developing the Discord bot
- Cybersecurity engineering: developing the Discord bot with user privacy and protection in mind
- Communication: keeping aligned with client and advisor goals
- Software testing: ensuring proper functionality of the Discord bot
- Documentation: developing written documentation to ensure future users or developers know how to use the application.
- UX Design: creating the Discord bot interactions to flow naturally for students.

1.3 Skill Sets covered by the Team

- Software engineering: Kyle Rooney, Kristen Nathan, Sophie Waterman Hines, Patrick Demers, Cole Mullenbach
- Cybersecurity engineering: Sophie Waterman Hines
- Communication: Kyle Rooney, Kristen Nathan, Sophie Waterman Hines, Patrick Demers
- Software testing: Kyle Rooney, Kristen Nathan, Patrick Demers, Cole Mullenbach
- Documentation: Kyle Rooney, Kristen Nathan, Sophie Waterman Hines, Patrick Demers, Cole Mullenbach
- UX Design: Patrick Demers

1.4 Project Management Style Adopted by the team

Our team plans to use the Agile project management methodology. Since we meet with our advisor/client weekly, the team will work in one week sprints. During our weekly meetings, we will refine the backlog and make a plan for the week ahead.

1.5 Initial Project Management Roles

- Cole Mullenbach: Software Engineer/Tester
- Kristen Nathan: Software Engineer/Tester
- Kyle Rooney: Software Engineer/Tester
- Sophie Waterman Hines: Cybersecurity engineer/software engineer
- Patrick Demers: Communication Lead and Software Engineer

2 Introduction

2.1 Problem Statement

Students in beginner programming classes at Iowa State University frequently have questions about their programming assignments. Since these questions are often repetitive and asked at odd hours of the day, it can take professors many hours to respond. This delay disrupts learning and frustrates young students. From the professor's perspective, taking the time to help each student consumes time from an already filled schedule. The Discord Bot project attempts to answer student questions immediately, without the need for a professor's interaction.

2.2 Requirements & Constraints

Functional Requirements

- The Discord Bot shall respond to a student within five seconds.
- The Discord Bot shall be configurable on a class and assignment basis.
- The Discord Bot shall be able to download up-to-date student code from Replit.
- The Discord Bot shall be capable of the following:
    - Executing tests on the student code.
    - Providing helpful hints to students based on compiler error messages.
    - Forwarding a student interaction to teaching staff if the bot is unable to assist.
- The teaching staff shall have an easy to use interface for updating the bot.
- The Discord Bot will be able to converse with multiple students at a time.

Resource Requirements

- The project must be completed by the end of Iowa State University's fall 2023 semester.
- The project deployment infrastructure must be capable of running on Iowa State University Virtual Machines.

Qualitative Aesthetics Requirements

- All snippets of code sent by the bot as examples will be written with correct formatting and spacing to improve readability.
- All bot responses will be easy to understand and formatted in an appealing and organized manner.
- All bot reaction prompts will communicate their intent through two methods to prevent issues with visual impairments.

UI Requirements

- The messages sent by the Discord Bot will support using button reactions for positive/negative feedback.
- The Discord Bot will be capable of escalating the student's request to a professor if its first attempt receives negative feedback.

2.3 Engineering Standards

- ISO/IEC/IEEE International Standard - Software and systems engineering --Software testing --Part 1: General concepts

Testing is something that is inevitable in every software project. Therefore, general concepts of testing are going to be a standard that aligns very closely with our project. We are going to have lots of system testing, unit testing, integration testing and user-acceptance testing.

- ISO/IEC/IEEE International Standard - Systems and software engineering -- Design and development of information for users

This standard aligns well with our project because it establishes what information users need, how to determine the way in which that information should be presented, and how to prepare the information and make it available. The main idea for creating our bot is to convey information to the students in Dr. Zambreno's class when asked. We need to implement standards for presenting the information effectively and making it available to the student when asked.

- ISO/IEC/IEEE International Standard - Software and systems engineering -- Software testing -- Part 5: Keyword-Driven Testing

Our project will involve extensive testing of keywords and phrases which will be the majority of our input from real life users. This standard will help guide us by providing a reference approach to implement keyword-driven testing and defining requirements on frameworks for keyword-driven testing. Ultimately, this standard will help us create keyword-driven test specifications, create corresponding frameworks, and build test automation based on keywords within our discord bot testing.

2.4 Intended Users and Uses

CPR E 161 students:

- Will be able to access quick advice/solutions for coding problems.
- Will spend less time waiting for an instructor to be available.
- Will be provided with helpful coding resources via bot commands.
- Will be able to access prior questions and answers.

CPR E 161 instructors:

- Will spend less time answering repetitive questions.
- Will be able to configure the Discord bot for their own questions and answers.

3 Project Plan

3.1 Project Management/Tracking Procedures

The agile project management methodology will be used to manage project progress. Agile will allow the team to make realtime changes to goals based on client feedback. Since weekly reports and advisor meetings are on a weekly basis, one week sprints will be used. This will allow the team to quickly address issues or misunderstandings as they arise.

A kanban board on Trello will be used to track team progress within a sprint. Before a sprint begins, the team will discuss goals for the next iteration and update the Trello board accordingly. As team members complete tasks, they will move the task to "Awaiting Feedback." Tasks in this column will undergo code review and functionality testing by another member of the team. When a task is moved to this column, a message is to be sent in the team Discord server to request feedback. Conversations coming from this feedback process should be constructive and thorough. Once the task is completed, it is to be moved to the completed status in Trello.

3.2 Task Decomposition

Task 1 – Research Discord Bots: Understanding how basic bots work.

- Buttons with Discord bots
- Threading with Discord bots
- Ticketing systems with Discord bots
- Reading reactions between Dr. Zambreno and his students
- Understanding how to parse error messages
- Understanding how replit works.

Task 2 – Bot Initialization: Setting up the CPRE 161 bot from scratch.

- Creating a bot from Discord's website
- Enabling all of the wanted admin features
- Use the given tokens to get the bot up and running
- Set up simple code that gets bot up and running
- Assign the bot to the CPR E 161 class Discord

Task 3 – Bot Implementation: Implementing all of the features we want the bot to be able to do.

- Button interaction with the bot
- Creating threads with buttons
- Creating a ticketing system
- Student Interactions
- Beautifully parsed error messages
- Access and download files from Replit
- Run test cases from Replit

Task 4 - Deployment Setup: Automatically deploying the code to the virtual machine.

- Setup Docker on the virtual machine
- Create a docker compose file for the bot.

- Setup a CI/CD pipeline in GitLab.
- Setup Grafana for log accessibility.
- Send logs to Grafana using Loki or Prometheus.

Task 5 – Internal Bot Testing: Making sure all implemented features are working correctly.

- Complete manual testing for student user
- Test administrator side of bot

Task 6 – Reiteration: Make fixes and add any more wanted features.

- Refine and clean up code while adding anything extra

Task 7 – Deployment to server: Deploy the bot to the current CPR E 161 class to get more feedback.

- Make changes based on feedback
- Supervise bot use with students

Task 8 - Automatic Question Answering

- Automatically attempt to help students before pinging a professor.

Task 9 - User Experience/Workflow Optimization

- Improve bot experience.

Task 10 - Professor Controls

- Allow professors to configure assignments and class sessions.

Task 11 - Code Rework and Optimization

- Fix sloppy or unclear code.
- Ensure proper usage of environment variables.
- Dependencies updated.

Task 12 - User Acceptance Testing

- Test with teaching staff to identify workflow problems.
- Test with students to gauge engagement and experience.

Task 13 - Revisions from User Acceptance Testing

- Fix any issues identified during User Acceptance Testing.

Task 14 - Handoff and Documentation

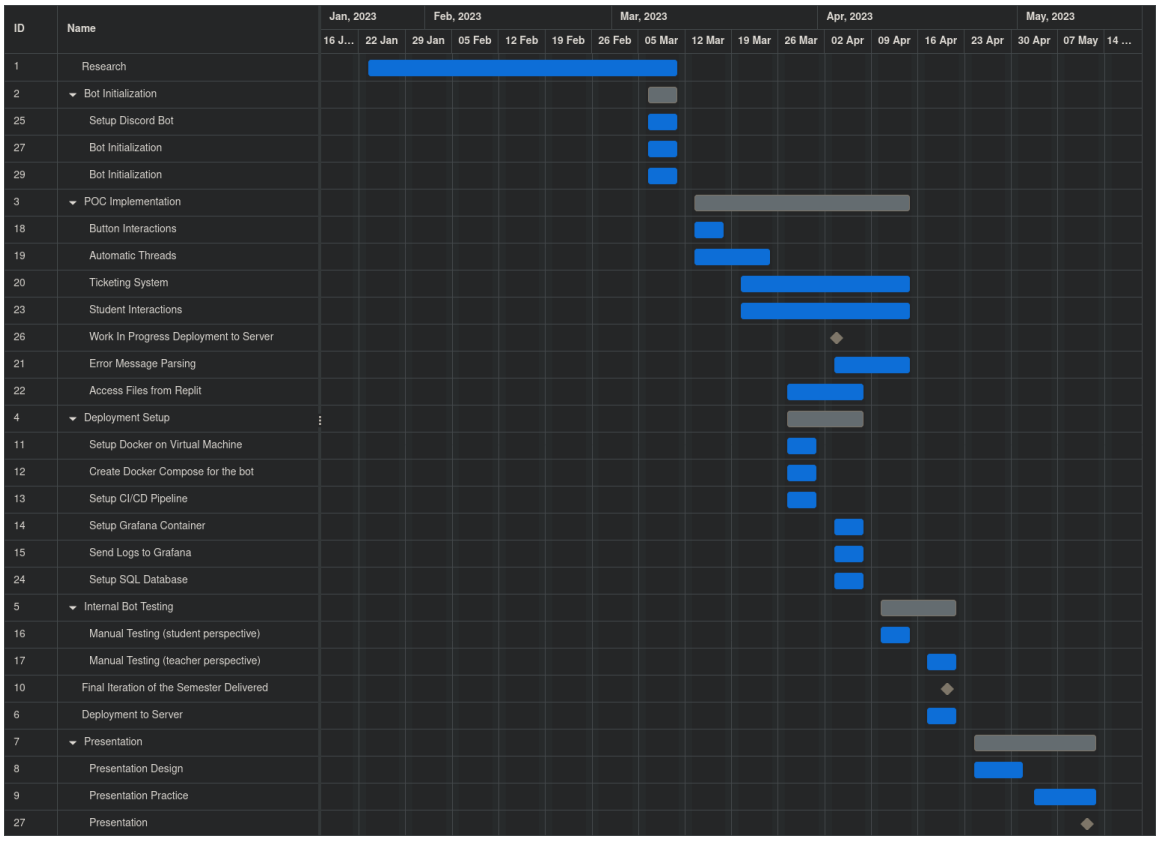
- Document running code, where the codebase is, environment setup, etc.

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

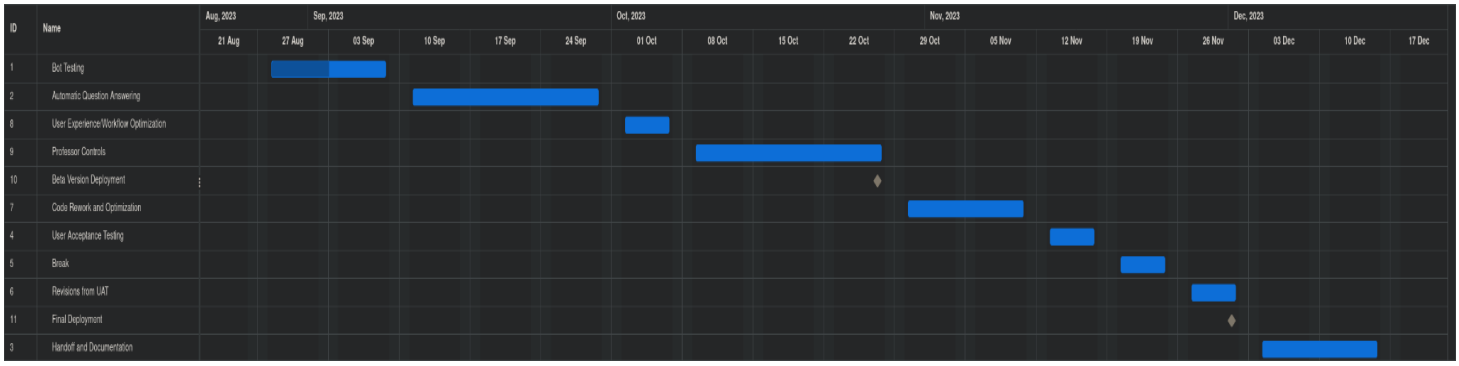
In an agile development process, these milestones can be refined with successive iterations/sprints (perhaps a subset of your requirements applicable to those sprint).

- Opening and closing tickets in our ticketing system
- Being able to use threads
- Adding buttons for user interaction
- Storing ticket information for easy accessibility
- Being able to download and compile Replit files.
- User deployment for testing

3.4 Project Timeline/Schedule



In the first semester of the project, the team's focus is on a proof of concept bot. The work will include creating the ticketing system, basic student assistance, and integrating with Replit. Additional software infrastructure will be developed to aid in testing, deployment, and observability. By the end of the first semester, a first version of the bot will be available in the class Discord server.



The second semester will be aimed at perfecting student interactions, automating answers, and refining code. The team will also take time to optimize the codebase for extensibility and readability. Prior to Thanksgiving, we will perform User Acceptance Testing which will include having Dr. Zambreno, TAs, and students test the bot. Then, a week will be spent iterating on the feedback received. Lastly, documentation and instructions will be handed off to Dr. Zambreno to ensure the project can continue to thrive in the future.

3.5 Risks And Risk Management/Mitigation

An Agile project can associate risks and risk mitigation with each sprint.

Trying to integrate Replit without an API will have certain risks. This will affect the project's long term viability as the Replit interface may change with time and affect the effectiveness of the bot. This has a risk factor of 0.4. Other than that, our project is primarily software based and does not have much risk involved.

3.6 Personnel Effort Requirements

Task:	Task Time:
Task 1 – Discord Bot Research	This task requires the most time for completion, as deciding which topics to research and how much time to devote to each topic may change as the project evolves. Because of this, an estimated time for this task is about 60 hours.
Task 2 – Bot Initialization	The bot initialization step entails creating the Discord bot from scratch, including using the Discord bot API to create a bot API token and assigning bot permissions. This task will not require much time, so the time estimate given is 2 hours.

Task 3 – Bot Implementation	Bot implementation will be another time consuming task project, as it encompasses the entirety of creating the ticketing system and thread management system, as well as reading reactions and storing thread information. The time estimate for this task is 60 hours.
Task 4 - Deployment Setup	Bot deployment setup will require tying many separate functionalities together. Creating a completely automated deployment pipeline with necessary support infrastructure will take 25-30 hours.
Task 5 – Internal Bot Testing	Internal bot testing consists of making sure the bot features work correctly and are resistant to incorrect/malicious input. This task completion time will depend on the quality of our work in task 3 and testing methods used, however our estimated completion time for this task is 15 hours.
Task 6 – Reiteration	Reiteration involves refining bot features and fixing any issues that appeared during bot testing. Our estimated time for this task is 10 hours.
Task 7 – Deployment to server	Deployment to the server involves deploying the bot to the class discord server. While this process is short in theory, there is the possibility that something goes wrong while trying to deploy the bot (permission issues, import issues, etc.) Therefore, the estimated time for this task is 10 hours.
Task 8 - Automatic Question Answering	Automatically answering student questions without professor interaction is difficult since students ask questions in unique ways. Research into methods of matching questions, natural language processing, and

	computer algorithms may be necessary. This portion of the project is estimated to take 80 hours.
Task 9 - User Experience/Workflow Optimization	While optimizing the user experience, the team will take a step back to ensure all aspects of the bot are friendly, easy to use, and non-obtrusive. This may involve gathering feedback from peers or professors. This will take 30 hours.
Task 10 - Professor Controls	Professor controls will allow the professor to manage classes and assignments through Discord. This is done in an effort to keep interactions managed through Discord. This is estimated to take 50 hours.
Task 11 - Code Rework and Optimization	As the codebase grows, the team needs to take time to optimize it - particularly with the impending handoff to Dr. Zambreno in mind. This will take approximately 30 hours.
Task 12 - User Acceptance Testing	At the conclusion of development efforts, the team will spend a week on User Acceptance Testing. This will allow the team to gather any last feedback of revisions that need to be made. This is estimated to take 20 hours.
Task 13 - Revisions from User Acceptance Testing	The team will spend 30 hours making revisions from the feedback gained during user acceptance testing in the previous week. These changes should be relatively minor but may require swift rework.
Task 14 - Handoff and Documentation	At the conclusion of the project, the team will hand the project over to Dr. Zambreno with concise, clear documentation.The team will spend 40 hours performing documentation and handoff related tasks.

3.7 Other Resource Requirements

A virtual machine provided by the Electronic Technology Group.

4 Design

4.1 Design Context

4.1.1 Broader Context

Area	Description	Examples
Public health, safety, and welfare	This project aims to make the feedback loop simpler and faster for both students and professors. If executed properly, the bot will increase the mental health of both parties.	Using the bot will help reduce student anxiety associated with asking questions. The faster response times will increase student free time allowing for more time on healthy activities.
Global, cultural, and social	Iowa State works to foster an environment for learning and inclusion. The bot will carry out these goals by enabling all students to ask questions and gain insightful feedback.	As students ask questions at odd hours of the day, the bot will provide answers. Additionally, students uncomfortable speaking English will have the opportunity to communicate through a text-based medium.
Environmental	The Discord bot will have minimal environmental impact beyond the electricity needed to support network transmissions and the virtual machine's resources.	The impact on the environment will primarily depend on the energy sources being utilized to power the computer equipment. The team will work to create an efficient bot which executes as lean as possible to minimize electricity usage.
Economic	This project has minimal economic impact since the only cost is the virtual machine provided by the Electronic Technology Group.	The team plans to constrain resources to the virtual machine and not utilize any further physical resources in an attempt to create an economically viable bot deployment. Economic impact could be minimized by spinning the virtual machine down between semesters.

4.1.2 User Needs

Students in Programming Classes

Students in programming classes need an efficient method to answer questions on homework assignments because a quick feedback loop improves student learning.

Teaching Staff

Teaching staff need to reduce repetitive student questions because answering the same question over and over is not an efficient use of time.

4.1.3 Prior Work/Solutions

As Discord is a relatively new medium for student learning, there are fewer relevant projects than one may expect. Relevant projects include StudyBot and Pymon.

StudyBot aims to enhance the learning experience of students on Discord by providing materials that may be useful while studying. This bot can provide formula sheets for math, a dictionary for English, a unit converter for science, and other useful utilities. The interaction style and value provided by this bot can be used when designing the Discord bot which will help Iowa State students.

Pymon is a Discord bot which can answer student questions based on a predefined set of responses. The bot builds its knowledge database from a JSON file. Then, when a student asks a question, it matches it to the closest related question. This technique could be used by this senior design project to improve the bot's question/answer abilities.

Both of the previously discussed Discord bots provide great functionality on their own, but do not provide for the needs of beginner programming students at Iowa State. Other resources the team has reviewed includes the source code of open source Discord bots since these codebases provide rich details about bot implementation and design.

4.1.4 Technical Complexity

Components/Subsystems:

- Ability to have automated replies to student questions
- Ability to organize and micromanage conversations
- Ability to create and edit threads in Discord
- Ability to simplify complicated error messages for easy understanding
- Ability to run code and test cases within Replit

The problem scope is specifically designed to aid the students and staff of the CPR E 161 class. There are other Discord bots in the public that are used for educational purposes, but none of them are designed to help with the particular content of CPR E 161. The group has yet to discover a Discord bot that is directly integrated with Replit. For our bot to be useful, it will need to have the ability to run code and test cases within Replit just from a link provided by a student. This has not been done before based on current research, and will be a challenging task considering there is no available API for Replit.

4.2 Design Exploration

4.2.1 Design Decisions

1. Ticketing System: The team's design idea to keep track of unresolved problems involves implementing a ticketing system. Initially a student will ask a question to the bot. The bot will then attempt to answer the students question automatically. After the bot gives a solution, the student will then react to the bot by telling it, “yes my question has been answered”, or “no I need help from the professor”. If the student reacts with “yes my question has been answered”, then the ticket will be closed. If the student reacts with “no I need help from the professor” then a ticket will be opened. Next, the professor and TAs will be tagged and will be able to easily go into Discord and see open tickets which symbolize unanswered questions. Once the staff helps the student, the ticket will be either closed by a TA or the professor.
2. Use of threads: The team has decided to handle the extra messages within the Discord channel by using Discord threads. Discord threads are a way of organizing conversations and micro managing conversations automatically. Our bot will automatically be creating, deleting, and talking in threads to help manage the class.
3. Docker: The Discord bot will be deployed using Docker. Docker is a platform that allows developers to package their applications and dependencies into self-contained containers that can run on any machine. These containers provide a consistent and reliable environment for the application to run in, regardless of the underlying infrastructure. This will make the project more sustainable into the future since the bot can easily be run on different virtual machines.
4. Programming Language: The Discord bot will be written in Python. Python is a high-level language known for its simple syntax and variety of third party modules. By using Python, the project will be easily maintainable by the client.
5. Py-Cord: When building a Discord bot, it is tedious to implement all user interface and interactions from scratch. The project will use Py-Cord as the primary means to interact with Discord since it enables quick development and clean code.

4.2.2 Ideation

1. Discord.py: A Python wrapper for the Discord API that allows you to create bots with features such as message sending, role management, and voice connections.
2. Py-Cord: Another Python wrapper for the Discord API that provides similar functionality to Discord.py, but with a focus on performance and scalability.
3. Direct API interaction: You can interact directly with the Discord API using HTTP requests and websockets, without using a wrapper library.
4. Discord Bot Maker: A drag-and-drop interface for creating Discord bots without any coding knowledge. While not a Python-based solution, it can be used to create bots that interact with Python scripts or programs.
5. Bot templates: Some developers have shared templates or starter code for creating Discord bots in Python, which can be used as a foundation for building your own bot.

4.2.3 Decision-Making and Trade-Off

We identified the pros and cons of each ideated option by researching and discussing which option will be the most useful for our project. Ultimately we did not have to choose just one method of implementation, but needed to have a plan that made the implementation possible given the guidelines needed to be followed, along with being as simple and effective as possible. We decided to use Py-Cord because it is a much better user experience for both the programmer and user. Py-Cord is also highly scalable with good performance. There are many features within Py-Cord such as slash commands and views that will make our CPR E 161 Discord bot look more professional and improve ease of use compared to using direct API interactions or a Discord bot maker. Our project will also be using Discord.Py as it is a very fundamental wrapper API for Discord bots. Ultimately, Py-Cord is a newer and improved version of Discord.Py.

4.3 Proposed Design

So far the team has designed a way for the student to efficiently interact with the Discord bot. At first we tried using the basic Discord.py commands. We implemented the basic commands but they did not provide us with the ease of use that we were hoping for. We then decided to implement slash commands from the Pycord library, which allows students to easily understand what available commands the bot provides. Slash commands also make formatting/inputting commands extremely simple.

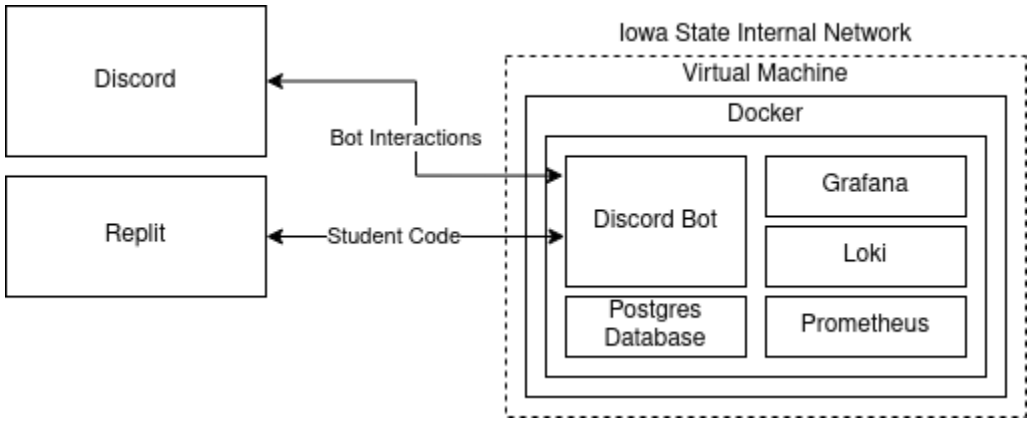
Originally, the group tried using default emoji reactions to react to the bot's messages. This worked, but seemed clunky and was not as clear as we wanted the communication to be. Instead we implemented a view of buttons using the Pycord library which allows us to have a very organized and professional look that is very portable across our code.

The team has also tried implementing the creation and management of Discord threads. Recently, the team was able to create threads based on reactions, but still have more implementation and testing to do in this area.

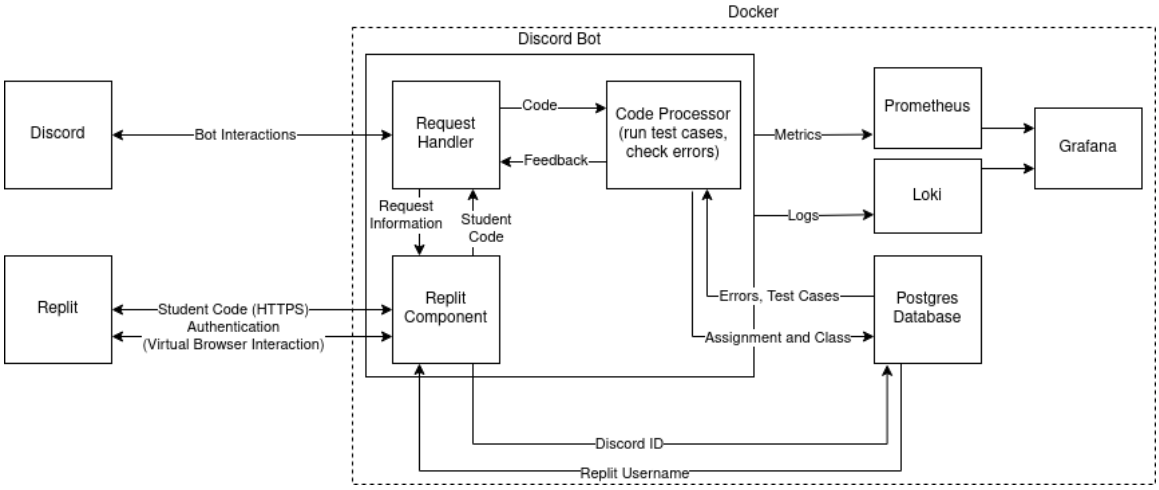
The group is now starting to implement a ticketing system to make sure that if the bot cannot help the student that an instructor will be able to get notified and see the issue. Once the ticket is opened, instructors will be able to communicate with the student until the issue is resolved.

The team is currently working on using a student's Replit url to download and compile files to give additional feedback on assignments. This feedback will help students resolve and understand errors to pass unit tests in order to complete the assignment.

4.3.1 Design Visual and Description



From a high level perspective, the project will consist of five Docker images. The Discord Bot and Postgres Database will work in tandem to run the Discord Bot. Grafana, Loki, and Prometheus will provide observability by providing project maintainers with logs and metrics about the application.



Looking into the finer details of the Discord Bot, there are a few components which will work together. The request handler will receive all Discord interactions and be responsible for sending responses. This request handler will interact with the other components to formulate the responses.

The Replit component is responsible for interacting with Replit which includes maintaining authentication and fetching student code. When a request is received which requires fetching code from Replit, the request handler will send the request to the Replit component. The Replit component will then determine the URL to fetch code from based on the Replit Username which can be looked up in the Postgres database. The Replit component will then send a request to Replit to fetch the student's code.

The code processing component is responsible for taking a piece of C code and checking it for errors and running test cases. The test cases to run will be stored in the database and can be looked up based on the assignment number and class.

The entire Discord bot will send logs and metrics to Loki and Prometheus respectively. Loki and Prometheus are data providers for Grafana, which provides an intuitive user interface for project maintainers to access information about the application.

4.3.2 Functionality

The bot is designed to be used by a student and an admin, the professor or TAs. Students and admins will be part of a class Discord that will allow students to ask questions of staff. Students can use the bot as a first step in asking for help. If able the bot will give a solution to the student or escalate the issue and notify the professor. The bot is also intended to be used for when the professor is not available to help students. This could be during late hours of the night or when the professor is too busy to respond. At any time the admin can add or change bot responses for easier use in the future. The bot will also be able to access a student's Replit files in order to download and compile the assignments to be able to give helpful feedback.

The current design satisfies all functional requirements. The bot will be able to use a ticketing system to help assist students at all times with a variety of ways.

4.3.3 Areas of Concern and Development

One of the groups biggest concerns within the design is, how capable is the bot going to be when answering questions automatically? For example, when a student has a question as to why their code is not working, will the bot be able to provide the answer just by being given the link to the student's code?

The next concern is if the bot will be able to communicate with the student in a helpful manner. For example, will the bot be able to give the student enough information so that a ticket is not always needed to be opened? Will an instructor be needed for every question still?

4.5 Technology Considerations

Using Python to code our bot:

- Strengths
    - Python is easy to learn and read
    - Python has convenient libraries to use such as Discord.py and Pycord
    - Python has a large community which gives us more material to research
- Weaknesses
    - Some of the simplicity of syntax such as brackets can be confusing for developers
    - Python has dynamic typing which can lead to runtime errors and bugs
    - In Python, white space matters, which can be a learning curve to developers
    - Python's interpreter may be slower than compiled languages
- Trade-offs
    - Python is a great choice for ease of use developing, but performance may be an issue of concern if the code base gets large enough
    - Dynamic typing can be either an advantage or a disadvantage depending on the programmer

4.6 Design Analysis

The plan that was proposed has been successful. The team is on track to have met all of the milestones for the given semester. The bot has a well designed response and interaction system, and a ticketing system using threads which allows the bot to operate without a database. Instead the bot can get the whole thread and move it into a "Finished Help" text channel. This way other students can see if they had the same question. The team also implemented buttons and commands to create and move the threads. Lastly, the entire bot is in the process of being linked to Replit.

4.7 Design Plan

There will be a few modules needed to meet the requirements of the project. The Discord bot interaction, Relpit communication, code analysis, and question answering.

The Discord bot interaction has already begun development and meets the core requirement of being a Discord bot. This module will define the different commands and responses for the Discord bot.

The Replit communication module will communicate with Replit to fetch user code. This meets the requirement of building the bot to be integrated with Replit. When a user command requires information from Replit, the Replit module will be asked to provide the information.

The code analysis module will take a provided C file and run some analysis on it. This helps meet the requirement of answering student questions without student interaction. Examples of output from the code analysis module may include compiling errors or test case errors.

The questioning answer module will take a student question and respond with an answer attempt. This meets the requirement of being able to help the student without professor interaction.

5 Testing

5.1 Unit Testing

- Input Commands
    - Dpy Test and Testcord will be used to aid in unit testing. The use of these tools will allow the team to simulate Discord interactions.
- Student Code Tester and Error Checker
- Replit Fetcher

5.2 Interface Testing

- Buttons - The team created a small bot to see if we could get buttons to pop up below the bots response and get the buttons to produce the output or a similar output to what is needed.
- Command List - Used the same bot to get the whole list of commands that are available to pop up by just typing the "/" key which is how the bot knows you want to communicate with it.
- Responses - Asked the bot to give small responses so the team could tweak the way the response was displayed in discord.
- Threads- Test the ability for the bot to create a thread that adds the proper users to the new discussion thread.
- Will be using Testcord and dpytest to test these features

5.3 Integration Testing

- Database Integration (testing that the bot interacts with the database)
- Replit Integration (testing we can grab what we need from Replit, authentication, code, etc)
- Conduct load testing to ensure expected traffic

5.4 System Testing

- Connectivity testing - after deployment, the bot needs to be online and connected to Discord.
- Interaction testing - The bot should be able to interact with students to provide answers to inquiries. This is where the bot's requirements can be evaluated.
    - Ability for the bot to resolve simple queries without professor interaction.
    - Ability for the bot to create a new thread for the student and professor if the bot is unable to help.

5.5 Regression Testing

Whenever new code is introduced to the repository, automated tests will be conducted by the CI/CD pipeline to ensure previous functionality is not broken. If any features are broken, the changes will be rejected until the issue is resolved.

In the event bot functionality is found to be broken, the following process will be followed:

- A new failing test case will be written.
- The bug will be resolved in the codebase.
- The originally written test case will be validated to be a passing test case.

Acceptance Testing

We will do manual testing ourselves to determine if it works functionally and meets all non-functional requirements. Then once we determine everything works fine through all methods of testing listed above then we will deploy our first version to a CprE 161 class for them to use. This way real users can give us feedback on how helpful the bot is and what functionalities might need to be added or fixed. The client will be able to see the students using our first version of the bot in order to make any adjustments or notes. We will also approve the bot's functionality with the client before deployment.

5.6 Security Testing (if applicable)

- User account linking will be handled through the user sending a randomly generated verification code from the discord bot to an assignment created by a Replit bot user. This process must be secure to prevent students from linking their Discord account to someone else's Replit. To test this, we will ensure that the method made to create the Replit assignment assigns users correctly and will not permit users to add/remove anyone from the assignment.

- Adding the correct users to bot generated threads is another feature to be security tested due to wanting to keep these threads confidential until completion. To test this, we will make sure that our thread generation method cannot be tampered with through user

input (argument checking and input sanitization) and check the thread generation method to make sure it correctly assigns users to threads.

5.7 Results

Results will be determined at a later time, once the bot's development is more finalized.

6 Implementation

Our implementation plan for the next semester is to finish the bots basic interaction commands early on in the semester. Our second task is to have the bot completely linked to Replit. After we have linked to Replit we will move forward in automating solutions to students' questions. This will be the most challenging part of the project to implement. Once we get the automated answering implemented, we will be deploying the bot to Dr. Zambreno's CPR E 161 class for testing and live feedback.

7 Professionalism

This discussion is with respect to the paper titled "Contextualizing Professionalism in Capstone Projects Using the IDEALS Professional Responsibility Assessment", *International Journal of Engineering Education* Vol. 28, No. 2, pp. 416–424, 2012

7.1 Areas of Responsibility

Area of responsibility	Definition	NSPE Canon	IEEE Canon	Analysis
Work Competence	Perform work of high quality, integrity, timeliness, and professional competence	Perform services only in areas of their competence; Avoid deceptive acts	To maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations.	The NSPE and IEEE canons are similar in the fact that work is to be performed within areas of their own competence. The biggest difference is that the IEEE canon also brings up the idea of having "full disclosure of pertinent limitations"
Financial Responsibility	Deliver products and services of realizable value	Act for each employer or client as faithful	To reject bribery in all its forms.	Both NSPE and the IEEE code of ethics rejects all

	and at reasonable costs.	agents or trustees.		forms of bribery and ask employers to be honest and trustworthy with financial responsibility.
Communication Honesty	Report work truthfully, without deception, and understandable to stakeholders.	Issue public statements only in an objective and truthful manner; Avoid deceptive acts	To be honest and realistic in stating claims or estimates based on available data	The NSPE and IEEE Canons both require engineers to be honest and non-deceptive when reporting information. Neither canon explicitly states that communications must be understandable to stakeholders.
Health, Safety, Well-Being	Minimizes risks to safety, health, and well-being of stakeholders.	Hold paramount the safety, health, and welfare of the public.	To accept responsibility in making decisions consistent with the safety, health, and welfare of the public, and to disclose promptly factors that might endanger the public or the environment;	The IEEE canon combines the NSPE canon rules with the rules for sustainability. In addition, it requires the prompt disclosure of any information that could impact the public or environment; however, the IEEE canon does not mention anything regarding the well-being of stakeholders.
Property Ownership	Respect property, ideas, and information of clients and others.	Act for each employer or client as faithful agents or trustees.	To seek, accept, and offer honest criticism of technical work, to acknowledge	The property ownership canon for all three code of ethics involve the respect of

			and correct errors, and to credit properly the contributions of others;	other intellectual property, however the IEEE also includes being honest about work criticism and providing credit for the work of others.
Sustainability	Protect environment and natural resources locally and globally.	DNE	To accept responsibility in making decisions consistent with the safety, health, and welfare of the public, and to disclose promptly factors that might endanger the public or the environment;	The IEEE canon describes that we must take into consideration the sustainability of mankind. Not allowing us to create something that would turn into an existential disaster.
Social responsibility	Produce products and services that benefit society and communities	Conduct themselves honorably, ethically, and lawfully so as to enhance the honor, reputation, and usefulness of the profession.	To treat fairly all persons and to not engage in acts of discrimination based on race, religion, gender, disability, age, national origin, sexual orientation, gender identity, or gender expression;	NSPE requires all individuals to act honorably and lawfully. In the IEEE code of ethics they write out more directly that individuals should treat each other fairly and not discriminate based on any identity. Both require each other to be honorable and respectful to all people.

7.2 Project Specific Professional Responsibility Areas

Financial Responsibility applies to this project as minimizing deployment costs helps sustain the Iowa State economy. By releasing the bot at a low cost, the university will improve student experience and retention, which helps improve financials. This canon is met by the team at a medium level as financial responsibility is kept in consideration, but not the top concern.

Social responsibility applies to the Discord Bot as the primary goal is to improve the lives of students and instructors for CPR E 161. Our team meets social responsibility standards to a high degree since the primary goal throughout design and implementation has been the betterment of the lives of the bot's users.

Work Competence applies to our project because we all have to be capable of creating such a program. We have to be confident that we can create a program that is able to do what is asked without being put in a situation that would result in us creating something that could be twisted to use for harm. Our team is performing at a high level in this area.

Sustainability does not apply to our project since we are creating a discord bot that will not be a physical item. This means we won't be interacting with the environment or any natural resources. We can choose not to use a high volume of memory or storage to lower environmental impact but that does not affect sustainability as much as physical items. This is a low level performance item for our team since our development and design of the discord bot will not affect sustainability or natural resources.

Health, Safety, Well-Being does not largely apply to the discord bot because it will not have the power to harm or endanger anyone. As a software project it does not have the capacity to affect the general public or users in a harmful way. This is a low level performance for our team since it is not very applicable to our design.

Property Ownership applies to our project in more than one way. Our team is performing at a high level for this area of professional responsibility. Relating to our client, we need to respect his wants/needs. Dr. Zambreno was also kind enough to share access to the CPR E 161 Repl.it which gives us the power to negatively impact his work. We also have access to the CPR E 161 Fall semester's discord which includes messages between other Iowa State Students and Dr. Zambreno. We also need to credit others with work that we use from them throughout this project.

Communication Honesty applies to our project due to our responsibility to properly inform our advisor, Dr. Zambreno, on the status of our project. Since advisor communication allows us to determine project choices and outcomes, we perform this at a high level. Communicating what we believe is possible with our discord bot, time frames in which we believe feature completion is possible, and using objective analysis to make decisions on future endeavors falls into the canons for NSPE and IEEE.

7.3 Most Applicable Professional Responsibility Area

Work competence is an area of professional responsibility that is important to our project and our team has demonstrated a high level of proficiency in our project. To our project, work competence means that we will all work together and get everything completed to the best of our ability. With nearly everything in our project being software based, it is very easy for us to stay in our area of expertise.

8 Closing Material

8.1 Discussion

The result of our project is to give CprE 161 students another resource to use for help on their programming projects. The Discord Bot will act as an item for students to use and ask questions of when they need help. The bot will be able to help solve compiler issues, give resources for more information, and escalate issues to the TAs or professor when necessary. At this moment in our project we have been able to implement threading, user reactions, escalating issues, and creating a collection of commands that can be used on the bot. The Discord Bot is not entirely finished at this time. We are hoping to implement a Replit interface to be able to download and compile files next semester. As for our current state, the project has been successful and will be able to meet all requirements going forward.

8.2 Conclusion

This semester we have completed research, deployment, creating an architecture, as well as connecting our bot to discord. We've completed a multitude of commands that will be used with the bot as well as organizing communication between the bot and students with threading and reactions. Our goals moving forward are to keep implementing commands and the Replit integration. We are going to keep using an agile work environment to make progress on our future goals. Weekly reports and meetings as a team will keep us centered on our goals. We were unable to achieve these goals yet since we spent a large amount of time researching and implementing higher priority items.

8.3 References

"Add StudyBot Discord Bot: The #1 Discord Bot List." *Add StudyBot Discord Bot | The #1 Discord Bot List*, https://top.gg/bot/840063974858162216.

Grifski, Jeremy. "Meet Pymon: A Discord Bot That Can Answer Any Question You Want." *The Renegade Coder*, The Renegade Coder, 7 Apr. 2022, https://therenegadecoder.com/teach/meet-pymon-a-discord-bot-that-can-answer-any-question-you-want/.

8.4 Appendices

8.4.1 Team Contract

Team Members:

1) Patrick Demers

2) Sophie Waterman Hines

3) Cole Mullenbach

4) Kyle Rooney

5) Kristen Nathan

Team Procedures

Day, time, and location (face-to-face or virtual) for regular team meetings: Thursdays at 3:00 PM virtually.

Preferred method of communication updates, reminders, issues, and scheduling: Discord

Decision-making policy: Majority vote

Procedures for record keeping: Kyle will write down meeting notes within a Discord channel named “meeting-notes.” The minutes will also be included in this channel for each meeting.

Participation Expectations

Expected individual attendance, punctuality, and participation at all team meetings:

We expect every member to attend every meeting. We have set up our meeting time so everyone is available to attend. If there are any complications about attending a meeting, it needs to be relayed to the rest of the team. Each member is also expected to discuss what they have been doing, what they plan to do, and converse in all discussion/decision making.

Expected level of responsibility for fulfilling team assignments, timelines, and deadlines: Each member will be responsible for fulfilling their team assignments before the due date.

Expected level of communication with other team members: We expect each member to actively talk in Discord weekly. Each member needs to join in on discussion and decision making in meetings.

Expected level of commitment to team decisions and tasks: Each member needs to have a vote in team decisions.

Leadership

Leadership roles for each team member (e.g., team organization, client interaction, individual component design, testing, etc.):

- Patrick - Technical Analyst, Team Lead
- Sophie - Documentation, Cyber Security, Testing
- Kyle - UX Design/Interactions, Communication
- Kristen - UX Design/Interactions, Testing
- Cole - Technical Analyst, Testing, Web Admin

Strategies for supporting and guiding the work of all team members: The team plans to use agile to keep our work organized. Agile will also allow us to always have a list of things to work on and keep things organized. If any member is having issues, they will be able to reach out to the team through our Discord channel.

Strategies for recognizing the contributions of all team members: Within the agile board, we will be able to see who has completed cards. We will also be able to see code contributed to the gitLab repository.

Collaboration and Inclusion

Describe the skills, expertise, and unique perspectives each team member brings to the team.

- Kristen Nathan: I have a background in computer engineering including software development and electrical engineering. I have previous experience with automation writing in python as well as writing scripts that use flags/tags to get information from the user.
- Sophie Waterman Hines: As a cybersecurity engineering student, I have a large background in securing applications and addressing potential security issues. I also have experience with python and some experience with discord bots.
- Patrick Demers: As a software engineering student, I have built various types of software through my internships, side business, and open source contributions. By leveraging this past experience, I hope to help the team focus on writing an extensible, well-designed Discord bot.
- Cole Mullenbach: I am a software engineering student and I have had a lot of full application programming experience however this is going to be a new experience as I have not made a discord bot and I have not worked with python at a high level like this. I have the basics of python from outside of the classroom projects. I am always in discord whether it be for school or for video games and the bots are great. I am excited to work on this project.
- Kyle Rooney: I have experience with software related projects and have many team management skills. I have a little experience in Python and have been using discord for 3-4 years now. I have worked with teams of all sorts through software projects at ISU, FFA, and sports. I try my best and value being a good leader and teammate.

Strategies for encouraging and support contributions and ideas from all team members:

Have an open safe discussion that allows everyone to give ideas without fear of being judged. Allow each other to have kind and helpful criticism to get the best solution to any problems and reach a compromise. Have the mindset of “all ideas are good ideas”.

Procedures for identifying and resolving collaboration or inclusion issues (e.g., how will a team member inform the team that the team environment is obstructing their opportunity or ability to contribute?)

If a team member disagrees with the way a task is being handled or is having issues related to a team decision, it should be brought up during a weekly team meaning. After the team understands where the problem is being had, we will work to reach a compromise or make changes to the established plan.

Goal-Setting, Planning, and Execution

Team goals for this semester:

This semester, the team desires to understand how the bot should interact with its users. By the end of the semester, the plan is to have a prototype of the bot completed, and begin testing the bot on sample student interactions. During the second half of this semester, the class the Discord bot aims to help will be in session. The team plans to monitor the student's progress throughout the course and document areas the bot may be able to assist.

Strategies for planning and assigning individual and team work:

Discuss as a team how individual work will be split up and let each other choose the portions of the work they desire to work on.

Strategies for keeping on task:

Communicate with each other weekly and set a timeline for individual parts to be completed.

Consequences for Not Adhering to Team Contract

On a team member's first two infractions of this contract, the team will communicate what expectations are not being fulfilled to the problematic team member. In coming weeks, the team will attempt to reach out to the team member to help ensure they are staying on track.

If problems continue after the initial warnings, the team will continue to attempt to support the team member in question. At this time, the team will alert Dr. Zambreno and Professor Shannon as continued lack of involvement may jeopardize the project.

****

a) I participated in formulating the standards, roles, and procedures as stated in this contract.

b) I understand that I am obligated to abide by these terms and conditions.

c) I understand that if I do not abide by these terms and conditions, I will suffer the consequences as stated in this contract.

1) _____Kyle Rooney_____ DATE __2/19/23_____

2) _____Kristen Nathan_____ DATE __2/19/23_____

3) _____Sophie Waterman Hines_____ DATE __2/19/23_____

4) _____Patrick Demers_____ DATE ___2/19/23_____

5) _____Cole Mullenbach_____ DATE __2/19/23_____