COMP3850 Project Deliverable Certificate

Name of Deliverable	*D5- Final Group Reflective Report*
Date Submitted	*30/ 05 / 2024*
Project Group Number	*Team 26*
Rubric stream being followed for this deliverable	*SOFTWARE Rubric*

We, the undersigned members of the above Project Group, collectively and individually certify that the above Project Deliverable, as submitted, **is entirely our own work**, other than where explicitly indicated in the deliverable documentation.

INITIALS	SURNAME	GIVEN NAME	STUDENT NUMBER	SIGNATURE (IN-PERSON OR DIGITAL)
Rx	*Ruike*	*Xu*	*46271481*	RuikeX
KP	*Kritchanon*	*Prasobjaturaporn*	*46122206*	KritchanonP
LT	*Long*	*Trinh*	*46281657*	LongT
OB	*Bush*	*Oliver*	*46534032*	OliverB
KH	*Karl*	*Holzmann*	*47092475*	KarlH

List of tasks completed for the deliverable and activities since the last deliverable certificate with totals for each individual team member and the whole team

Performed by *(Student Names)*	Duration *(hrs)*	Complexity *(L, M, H)*	Name of task	Checked by *(Initials)*
Long Trinh	3	M	Project Planning	KH
Long Trinh	2	M	Reflection	OB
Long Trinh	2	M	Presentation	KP
Oliver Bush	4	H	Design	LT
Oliver Bush	3	M	Presentation	KH
Karl Holzmann	5	H	Bot Development	LT
Karl Holzmann	1	L	Implementation	OB
Karl Holzmann	2	M	Presentation	KP
Kritchanon Prasojaturaporn	3	M	Conclusion	KH
Kritchanon Prasobjaturaporn	2	M	Introduction	OB
Kritchanon Prasobjaturaporn	2	M	Presentation	LT
Ruike Xu	4	H	Requirements and Analysis	LT
Ruike Xu	3	M	Presentation	KP
Long Trinh	7			
Oliver Bush	7			
Karl Holzmann	8			
Kritchanon Prasojaturaporn	7			
Ruike Xu	7			
Team Total	36			

1. Introduction

In today's fast-paced digital communication landscape, the ability to quickly digest and comprehend vast amounts of textual information is crucial. Our project aims to address this need by developing a Discord bot that can efficiently summarise chat logs and paragraphs of text. This innovative solution is powered by the large language models provided by our sponsor, RapidAnalysis, through seamless API integration.

Our bot is designed to assist users in various scenarios, from casual conversations in social channels to important discussions in professional settings. By providing concise and accurate summaries, the bot enhances user productivity and ensures that key information is easily accessible without the need to sift through extensive dialogue.

In this report, we will detail the journey of our project, following the system development life cycle. We will cover each phase of development, from initial project planning and requirements analysis to design, implementation, and the valuable lessons learned along the way. Our goal is to provide a comprehensive overview of our process, challenges, and achievements, culminating in a reflection on the overall project and its future potential.

2. Project Planning

2.1. Risk Management:

Risks are categorised into technical, people, organisational, requirements, and estimation risks. The risk analysis matrix assesses the probability and impact of each risk, with strategies in place for mitigation. Regular stakeholder meetings, training, and conflict resolution are key strategies to manage these risks.

Our risk planning table:

Risk Number	Strategies
i.	Have regular meetings with stakeholders to discuss requirements and documentation. Try to differentiate ourselves from competitors in the market. Ensure all files containing sensitive credentials are excluded from version control, accounts all have multi-factor authentication and utilise GitHub's token leak scanner.
ii.	Ensure each team member gets the same training/documentation. Check-in/have meetings regularly to update the team on progress. Reschedule meetings or otherwise assist team members.

	Follow conflict resolution documentation in the team manual.
iii.	Try to differentiate ourselves from competitors in the market. Create and follow conflict resolution documentation in the team manual. Ensure we follow a code of conduct and follow conflict resolution if issues occur. Reduce code complexity to minimise costs.
iv.	Have regular meetings with stakeholders to discuss requirements and documentation.
v.	Check-in/have meetings regularly to update the team on progress. Communicate with stakeholders regularly to ensure accurate estimations are made by subject matter experts. Track costs. Regularly reevaluate estimations.

Most of our activities in the project timeline have followed these strategies strictly. The project planning phase was comprehensive, including a detailed risk management plan and resource allocation strategy. These initial steps ensured clarity and direction for the team.

2.2. Resources Management:

Human resources include software developers, a project manager, and a lead developer. Technological resources encompass development environments, version control systems, and collaboration platforms. Digital resources involve secure storage and management of code repositories.

Jira is used for task allocation and tracking.
Google Doc is used for documentation.
Discord is used for communication and testing.
Github is used for version control.

2.4. Change Management:

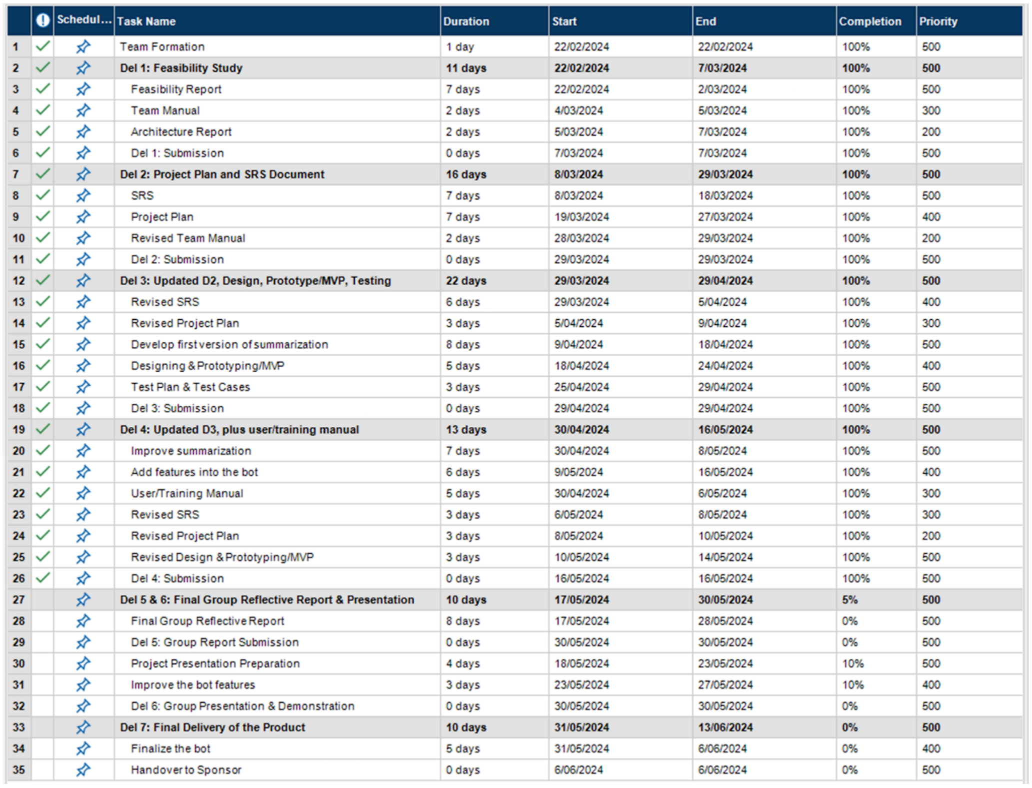
Changes to requirements and scope are managed through a formal change request process. Approved changes are documented and reflected in all project documents. All of these changes between each sprint or deliverable were recorded in the Revisions History Table. Version control is maintained using Git and GitHub, with tagging for release points and version history tracking.

2.5. Quality Management:

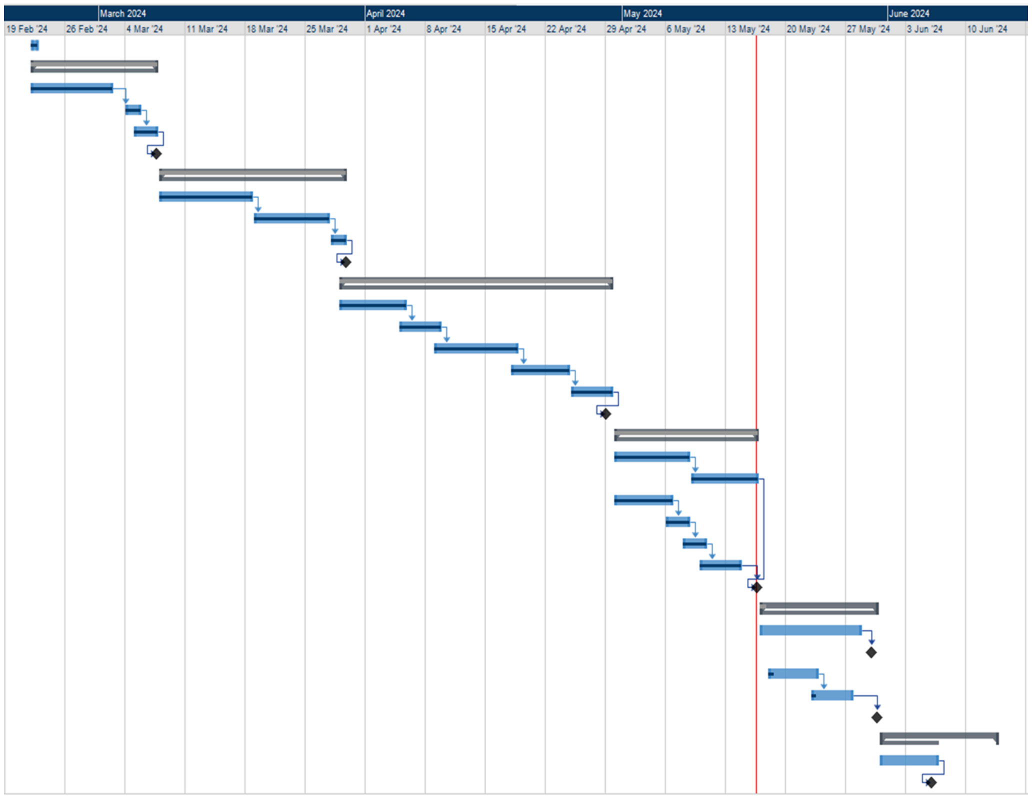
Quality planning involves setting clear, measurable goals and adhering to industry standards. Quality assurance includes regular code reviews and audits, while quality control encompasses comprehensive testing strategies. Continuous improvement is driven by user feedback and root cause analysis.

2.6. Project schedule:

The project schedule details tasks, deliverables, and timelines. Key tasks include team formation, training, research, development, testing, and final deployment. The timeline spans 14 weeks, with specific milestones for each phase of the project.



	!	Schedul...	Task Name	Duration	Start	End	Completion	Priority
1	✓	📌	Team Formation	1 day	22/02/2024	22/02/2024	100%	500
2	✓	📌	**Del 1: Feasibility Study**	**11 days**	**22/02/2024**	**7/03/2024**	**100%**	**500**
3	✓	📌	Feasibility Report	7 days	22/02/2024	2/03/2024	100%	500
4	✓	📌	Team Manual	2 days	4/03/2024	5/03/2024	100%	300
5	✓	📌	Architecture Report	2 days	5/03/2024	7/03/2024	100%	200
6	✓	📌	Del 1: Submission	0 days	7/03/2024	7/03/2024	100%	500
7	✓	📌	**Del 2: Project Plan and SRS Document**	**16 days**	**8/03/2024**	**29/03/2024**	**100%**	**500**
8	✓	📌	SRS	7 days	8/03/2024	18/03/2024	100%	500
9	✓	📌	Project Plan	7 days	19/03/2024	27/03/2024	100%	400
10	✓	📌	Revised Team Manual	2 days	28/03/2024	29/03/2024	100%	200
11	✓	📌	Del 2: Submission	0 days	29/03/2024	29/03/2024	100%	500
12	✓	📌	**Del 3: Updated D2, Design, Prototype/MVP, Testing**	**22 days**	**29/03/2024**	**29/04/2024**	**100%**	**500**
13	✓	📌	Revised SRS	6 days	29/03/2024	5/04/2024	100%	400
14	✓	📌	Revised Project Plan	3 days	5/04/2024	9/04/2024	100%	300
15	✓	📌	Develop first version of summarization	8 days	9/04/2024	18/04/2024	100%	500
16	✓	📌	Designing & Prototyping/MVP	5 days	18/04/2024	24/04/2024	100%	400
17	✓	📌	Test Plan & Test Cases	3 days	25/04/2024	29/04/2024	100%	500
18	✓	📌	Del 3: Submission	0 days	29/04/2024	29/04/2024	100%	500
19	✓	📌	**Del 4: Updated D3, plus user/training manual**	**13 days**	**30/04/2024**	**16/05/2024**	**100%**	**500**
20	✓	📌	Improve summarization	7 days	30/04/2024	8/05/2024	100%	500
21	✓	📌	Add features into the bot	6 days	9/05/2024	16/05/2024	100%	400
22	✓	📌	User/Training Manual	5 days	30/04/2024	6/05/2024	100%	300
23	✓	📌	Revised SRS	3 days	6/05/2024	8/05/2024	100%	300
24	✓	📌	Revised Project Plan	3 days	8/05/2024	10/05/2024	100%	200
25	✓	📌	Revised Design & Prototyping/MVP	3 days	10/05/2024	14/05/2024	100%	500
26	✓	📌	Del 4: Submission	0 days	16/05/2024	16/05/2024	100%	500
27		📌	**Del 5 & 6: Final Group Reflective Report & Presentation**	**10 days**	**17/05/2024**	**30/05/2024**	**5%**	**500**
28		📌	Final Group Reflective Report	8 days	17/05/2024	28/05/2024	0%	500
29		📌	Del 5: Group Report Submission	0 days	30/05/2024	30/05/2024	0%	500
30		📌	Project Presentation Preparation	4 days	18/05/2024	23/05/2024	10%	500
31		📌	Improve the bot features	3 days	23/05/2024	27/05/2024	10%	400
32		📌	Del 6: Group Presentation & Demonstration	0 days	30/05/2024	30/05/2024	0%	500
33		📌	**Del 7: Final Delivery of the Product**	**10 days**	**31/05/2024**	**13/06/2024**	**0%**	**500**
34		📌	Finalize the bot	5 days	31/05/2024	6/06/2024	0%	400
35		📌	Handover to Sponsor	0 days	6/06/2024	6/06/2024	0%	500



3. Requirements and Analysis

3.1. Introduction to Requirements and Analysis

In order to save people's time on social media (such as Discord), our group decided to try to build a Discord bot to quickly analyse large language models by using APIs to aggregate DIscord chat logs. Therefore, analysing the needs of our stakeholders is also one of the steps that we are currently considering and doing.

3.2. Requirements Gathering

- Stakeholder Identification:
  - Discord server administrators:
    Administrators use the Discord server to distinguish users who have registered for the Summary Bot and promote the health of the community.
  - End-users:
    Those users who need a chat summary bot to get a discord weekly summary of information and a summary of chat logs

- development team members:
  Team members involved in the project planinternal

- Supervisor:
  Advise on projects and review milestones
- School:
  Review the final project outcomes and summary report

- Methods of Gathering Requirements:
  - Internal:
    The supervisor will set the initial requirements for the project and continue to make progress.
  - External:
    Users can summarise the bot's feedback window through the chat to give feedback on the problems and suggestions encountered
- Types of Requirements:
  - Functional Requirements:
    - The bot should summarise chat conversations weekly or on-demand.
    - The bot should be able to handle multiple channels and provide channel-specific summaries
    - It should integrate with Discord's API and also provide users with commands to interact with the bot and the ability for users to change commands to personal preference.
    - The bot should support different summary formats

  - Non-Functional Requirements:
    - The bot should have high availability and reliability.
    - It should be user-friendly and easy to set up, and with a detailed user manual for the user to set up
    - Performance requirements:
      The response time and efficiency of the chat summary bot need to meet the needs of users and not be too long
    - Security requirements to ensure data privacy and prevent unauthorised access.

3.3. Documenting Requirements

- Requirements Specification:
  - With a clear project plan (e.g. Gantt chart), you can see the percentage of completion and the time spent at each stage of the project
  - Each stage should have a corresponding report that provides a detailed description of each stage.
  - Fixed meeting times for the team and supervisors, and weekly reports summarising issues to understand follow-up needs.
  - A PDF for reporting all the results and follow-up plans to present to all possible stakeholders.

- Requirement prioritisation:
    - First, online (face-to-face) meetings, Gantt charts and periodic reports are requirements that need to be carried out on a regular basis, which involve the real-time progress of the project and other new or existing requirements and issues that need to be addressed over time, so that stakeholders can see more clearly that the requirements are being addressed over time.
    - Second, the final report summary and PDF, the ultimate goal of which is to provide stakeholders with a clear and brief summary of whether the final outcome of the project has met expectations during this time.

- Requirement Traceability:
    - Internal:All internal requirements (e.g. supervisors, schools) can be seen in the periodic report and the Chat Summary Bot update log
    - External:In the bot's feedback window, the changelog allows you to trace the progress of the requirements throughout the project lifecycle

- Requirement analysis:
    - Feasibility Study
        - Technical Feasibility:
            - Discord API Capabilities:Try the Discord API to make sure it can support the features required by the chat digest bot, and see if the API allows access to chat history, real-time messaging, and command processing.
        - Technology Stack:
            - Assess the suitability of the chosen technology(javascript) stack of implementing the bot and consider the libraries and frameworks that facilitate natural language processing (NLP) and data summarization.
        - Integration with Discord:
            - Ensure that the chat summary bot can be integrated seamlessly with discord servers. Easier to deploy for users and simple to maintain.
        - Operational feasibility
            - User Adoption:
              The bot commands and functionalities should be intuitive for server administrators and end-users. To make sure of the ease of use.
            - Maintenance:
              A plan for regular updates and maintenance. Identify resources required for monitoring and improving the chat summary bot based on user feedback.
    - Impact Analysis
        - Scope Impact:

                - Requirement Changes or Additions: New requirements or changes in requirements will increase the time cost of project completion and development costs.
                - Project Boundaries: Because the purpose of the project is to design a chat summary bot based on Discord, a social software, its use also needs to meet the requirements of Discord, and it can only analyse the information that the user wants to summarise.
        - Time Impact:
            - Development Timeline: Significant changes in requirements (e.g. using other social media or adding features that are not related to chat summarization) may cause the project to be restarted, and small changes in requirements (increasing the bot's summary word limit, or adding new features related to chat summarization) can be changed in the future regular maintenance and updates of the program, and will not affect the main direction of the project.
            - Milestones:Identify key milestones and deadlines(weekly meeting and periodic report). Ensure that the project stays on track and any delays are anticipated and managed.
        - Cost Impact:
            - Resource Allocation: Determine the resources needed for each requirement and consider both human resources (developers, testers) and technical resources (servers, software licences).
        - Quality Impact:
            - Performance Metrics:The responsiveness of the program as well as the accuracy of the summary
            - Testing and Validation: Rigorous testing to ensure that the bot can meet all current requirements
    - Risk Assessment
        - Monitoring and Control:
            - Regular Reviews: Adjust plans as needed based on new information and project progress from the supervisor.

4. Design

4.1. Introduction to System Design

The design phase was a pivotal part of our project, shaping the overall architecture, user experience, and operational efficiency of our Discord bot. The design phase outlines all key design elements, including system architecture, data persistence, concurrency management, user interface strategy and design decision. Furthermore the iterative nature of the design process is clear and detailed, showing changes and adjustments made to the final implementation of the system.

4.2. System Architecture

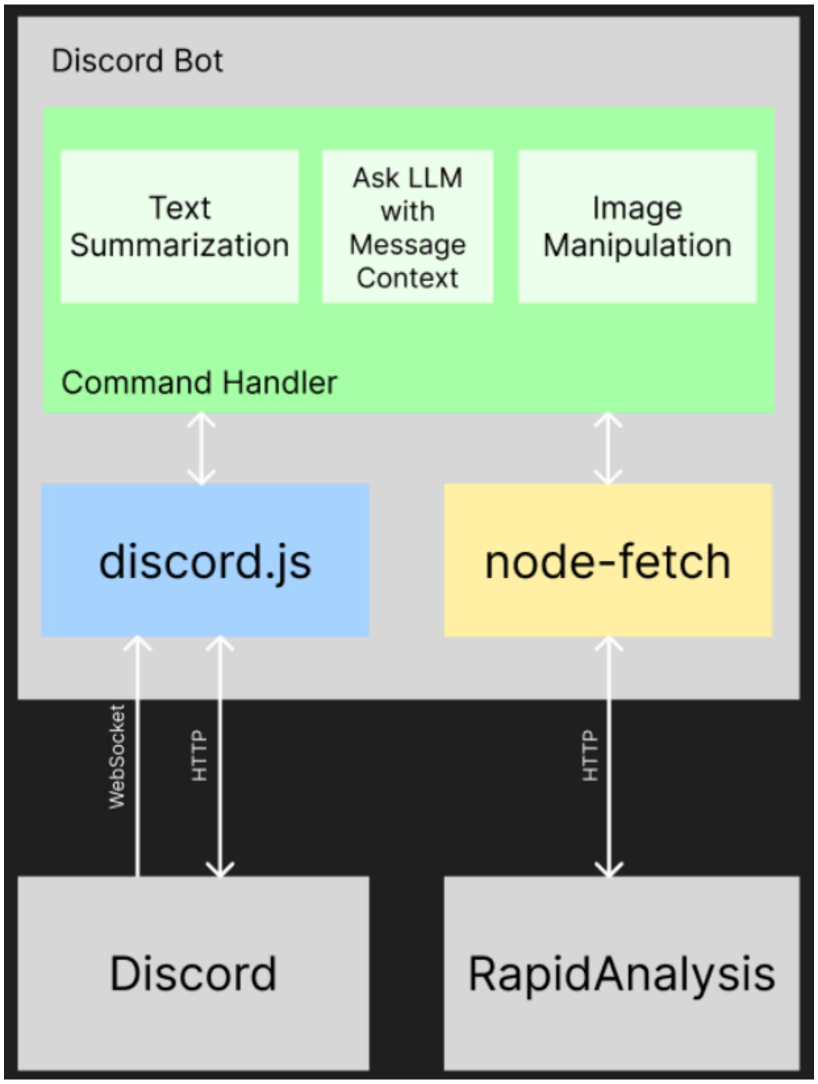


Fig 1: System architecture diagram

Our system architecture uses Discord's WebSocket and REST API, while utilising discord.js to handle Message Create and Interaction Create events, this is for parsing commands and managing operations such as sending messages, images, and mass fetching messages. User and guild data such as API keys and settings is persisted using the database engine MySQL. Although JavaScript is a single threaded language we have employed the event loop within JavaScript to handle long running tasks efficiently.

4.3. User Interface Strategy

Our user interface strategy is covered by three main concepts. Seamless integration with Discord, Accessibility, and a Feedback Mechanism.

- The **integration with Discord** is achieved by utilising the text-based commands and interactions within the Discord chat.
- **Accessibility** for users is ensured by making all text outputs accessible via screen readers and making commands easy to learn and use.
- **Feedback mechanism** is implemented with simple responses that allow users to provide feedback quickly and directly.

4.4. Design Decisions

The decision to use Discord as the platform for the bot involves several design choices and trade-offs:

- **Advantages:** Discord offers a robust infrastructure and a large user base, providing a platform for the bot to operate within. Its API allows for seamless integration and interaction with the messaging feature.
- **Challenges:** Using the Discord API limits the bot's functionality due to message length restrictions and rate limits, which may impact the bot's performance and scalability. Additionally, Discord's policies could be largely limiting.

4.5. Design Reflection

Throughout the project there were many design changes made to the final implementation of the Discord bot. These changes were made as we gained greater insight to the project and responded to our sponsor feedback. Outlining our effective utilisation of the Agile methodology.

Key changes and Decisions

- **Command Parsing:** Initially, we opted for command parsing using "." for faster testing but later transitioned to Discord's native slash commands ("/") to align with standard practices, enhancing user experience.
- **Command Simplification:** Commands were simplified, such as changing "/summarise" to "/sum" and adjusting parameters like "number of latest messages" to "limit," streamlining user interactions.
- **Removed Commands:** We removed the "/cancel" command in favour of a more intuitive backspace or click-out method, and simplified feedback options to "good" and "bad" buttons, which improved usability but reduced detailed user feedback.
- **Removed Functionalities:** Some functionalities, such as summarising messages between two sets of time and searching by subject, were removed due to their

complexity and limited project scope. Server-level default settings were also discarded over privacy and complexity concerns.

- **New Command:** The introduction of the "/reg" command for API key registration was a crucial enhancement suggested by our sponsor, showcasing the value of continuous stakeholder engagement in refining our design.

Reflection on Agile Methodology

- **Flexibility and Iteration:** Our use of Agile allowed us to iteratively develop and refine the bot, responding quickly to feedback and changing requirements.
- **Requirement Analysis:** However, earlier and more detailed requirement analysis could have prevented some rework and scope adjustments.
- **Feasibility Study:** A more thorough initial feasibility study might have highlighted the complexities of certain features, allowing us to manage expectations and resources more effectively from the outset.

Overall, while our design evolved to meet user needs and technical constraints, a more proactive approach to requirement gathering and feasibility analysis could have been more efficient during design and minimised mid-project changes.

5. Implementation

The Discord bot was implemented in JavaScript, running on the Node.js runtime using discord.js as the library to interact with the Discord API. Each command is written as a separate class to ensure modularity and ease future maintenance of the bot. RapidAnalysis interaction is handled with an API client class which provides a method to ease network requests to the RapidAnalysis API. All these classes have been well documented to allow for ease of development. MySQL/MariaDB was used as our database backend for storing user preferences and API keys.

We chose these technologies mostly due to our collective experience with the libraries/languages. This would allow us to be more productive as a team rather than necessitating members learning new technologies or languages. This approach worked well for us as we had more time to improve our end product with new features or other feedback given to us from our sponsor.

6. Learning Outcomes

7.1. Reflection:

Planning: The project planning phase was comprehensive, including a detailed risk management plan and resource allocation strategy. These initial steps ensured clarity and direction for the team.

Execution: The development process was aligned with Agile methodology, allowing for iterative improvements. Regular meetings and sprint reviews facilitated continuous progress and adaptation to new insights.
Collaboration: Effective communication within the team and with stakeholders was maintained through weekly meetings and progress reports. This ensured transparency and timely resolution of issues.

7.2. What worked well:

Agile Methodology: The iterative approach allowed for flexibility and continual refinement of the bot's features, ensuring alignment with stakeholder needs and feedback. Weekly team and sponsor meetings created a great and collaborative environment, it helped push the project progress further. Additionally, each sprint role changing also helped members to get all the experiences needed for the future.
Risk Management: Proactive identification and mitigation of risks prevented significant disruptions to the project timeline. The risk planning table has created a great guideline for the team moving forward. Our team actions have been based on the table and there has not been any significant issue so far.
Resource Utilisation: Efficient use of tools like Jira and GitHub streamlined task management and version control, contributing to smooth project execution. Moreover, Google Doc and Google Calendar also helped the team on efficient documenting and planning.
Communication: It is an advantage for us as Discord is both our communication channel and development ground. All team members have communicated openly and regularly given feedback and suggestions. Even with the replacement of Shuo earlier in the second sprint, the team is still standing strong with newly-added member - Ruike. Continuous and lively discussions weekly helped us adapt and move the project forward.

7.3. Areas for Improvement:

Requirements Clarification: Initially, some requirements were too ambiguous, leading to rework. Earlier and more frequent communication with stakeholders could have clarified these points sooner.
Bot Development: The development of the bot took more time than expected due to unexpected issues regarding APIs' potential. Earlier development and better understanding of the API could have pushed the development faster.
Testing Phases: While testing was comprehensive, integrating better testing earlier in the development cycle could have identified issues sooner, saving time in later stages.
Documentation: Although documentation was thorough, ensuring all team members consistently updated documents could have prevented discrepancies and ensured everyone had the latest information.

7. Conclusion

The completion of our Discord useful bot project demonstrated the effectiveness of solid planning, and agile execution. Our adherence to these principles resulted in the successful delivery of a high-quality final product. Key achievements included robust risk management that kept the project on track, efficient use of resources through tools like Jira, GitHub, and Discord, and adaptive development practices that allowed for rapid integration of feedback, enhancing functionality and user experience.

We learned valuable lessons throughout the project. Early and clear analysis of requirements would have streamlined the project timeline by preventing mid-project adjustments. Additionally, implementing more rigorous testing early in the project could have identified potential issues sooner, optimising development time.

Looking ahead, we plan to expand the bot's features based on user feedback, potentially incorporating more advanced processing capabilities. We will also ensure continuous improvement through regular updates to maintain the bot's performance and relevance.

The project not only met its initial goals but also provided valuable insights into effective project management and software development practices. The experience has prepared us well for future projects, ensuring that we carry forward both the successes and the lessons learned to continue delivering high-quality software solutions.