MACQUARIE
University
SYDNEY·AUSTRALIA

COMP3850 Project Deliverable Certificate

Name of Deliverable	*Deliverable 2*
Date Submitted	*28 / 03 / 2024*
Project Group Number	*Team 25*
Rubric stream being followed for this deliverable (highlight one) *Note: the feasibility study has the same rubric for all streams.*	[✓ *SOFTWARE Rubric*] *GAMES Rubric* *CYBERSECURITY Rubric* *DATA SCIENCE 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)
YL	*Lee*	*Yasmin*	*46366377*	YLee
FG	*Gelwyn*	*Faith*	*46421114*	FGelwyn
DL	*Lee*	*Donghyun*	*45773602*	Lee
BB	*Bostami*	*MD N S Baizid*	*46128980*	baizid
IW	*Weston*	*Isabel*	*46570616*	IWeston
SR	*Rasmussen*	*Sebastian*	*47223421*	Sebastian R

NB: Please write all details clearly (if handwritten).

Performed by *(Student Names)*	Duration *(hrs)*	Complexity *(L, M, H)*	Name of task	Checked by *(Initials)*
All Team Members	1.75	L	Team Meetings	IW,FG,DL, SR,BB,YL
Yasmin Lee	6	M	SRS 2.2, 2.3	IW,FG,DL, SR,BB,YL
Faith Gelwyn	0.5	M	Edited Team Manual (Communication Plan and Meeting Schedule)	IW,FG,DL, SR,BB,YL
Faith Gelwyn	9	M	SRS 1.4, 2.1, 2.6, 4.2 and editing	IW,FG,DL, SR,BB,YL
Baizid Bostami	7	M	Project Plan- Risk Management, Resource Management and allocation, Change and quality management, Tasks SRS- User Interface, Software Interface	IW,FG,DL, SR,BB,YL
Donghyun Lee	1.5	M	Architecture and user requirements	IW,FG,DL, SR,BB,YL
Donghyun Lee	4	M	Software Requirements Specification -System features 1&2 -Security Requirements	IW,FG,DL, SR,BB,YL
Sebastian Rasmussen	4	M	SRS 2.4,2.5,4.4	IW,FG,DL, SR,BB,YL
Sebastian Rasmussen	1.5	L	Formatting SRS	IW,FG,DL, SR,BB,YL
Isabel Weston	5	M	Manual: change management and conflict resolution Plan: purpose, change, schedule SRS: introduction sections except for scope	IW,FG,DL, SR,BB,YL
Isabel Weston	2	L	Editing and formatting all documents	IW,FG,DL, SR,BB,YL
Yasmin Lee Total	7.75			
Faith Gelwyn Total	11.25			
MD N S Baizid Bostami Total	8.75			
Donghyun Lee (Ben) Total	7.25			
Sebastian Rasmussen Total	7.25			
Isabel Weston Total	8.75			
Team Total	51			

Infobyte

Team 25

Project Plan

Slack app with RapidAnalysis API

Version 1.0 28.03.2024

Contents

1 Statement of Purpose

The purpose of this document is to present a plan for the development and deployment of an AI chatbot in Slack using the RapidAnalysis API.

The team, InfoByte, will develop a Slack app for the client, RapidAnalysis. The chatbot will address common issues in team communication over Slack by summarising long messages and giving users intelligent, context-aware responses to their prompts. The final product will be a user-friendly solution that is monetisable by RapidAnalysis and extensible by other developers in the future.

This project plan will outline the management of risks, resources, changes, and quality of the project, as well as the schedule for the project, including a timeline of deliverables, overview of tasks with specific examples taken from Jira, resources allocated and assumptions for the duration of the project. It will assume a familiarity with basic acronyms involved in the project, used in other documents like the Team Manual.

2 Risk Management

Risk management is essential for discovering, assessing, and managing possible threats to a project's objectives and success and informing decision-making. A three-step risk management strategy has been established for this project:

1. Risk identification
2. A matrix that categorises risks according to type, asset, severity, likelihood, and impact
3. Mitigation strategies

2.1 Identification

Risks to the project have been identified in the following areas.

2.1.1 Technology

1. Dependency on third-party tools, namely APIs and libraries.
2. Data breaches, unauthorised access, and compromised user information
3. Delays in development, poor quality code.
4. Integrating RapidAnalysis API with Slack could lead to complexity.
5. User acceptance testing (UAT) issues can lead to severe functional and non-functional problems.
6. A lack of scalability can slow down the system
7. After development, the API may face legal and compliance risks as there are already several other chatbots in the market, e.g., ChatGPT, Microsoft Copilot, and Google Gemini.
8. As the team members are University students, the team might have skill gaps.
9. Ample data is not provided to the language model, so the LLM can't provide enough response.

10. Chatbot infrastructure and maintenance can cause bot downtime.
11. Users can input spam or abusive material to the chatbot.

2.1.2 Human

1. Loss of contact.
2. Inability to attend the meeting.
3. Withdrawal from the project.
4. Conflict within the team.
5. Incomplete tasks due to lack of ability.
6. Incomplete tasks due to lack of motivation.
7. Incomplete tasks due to illness.

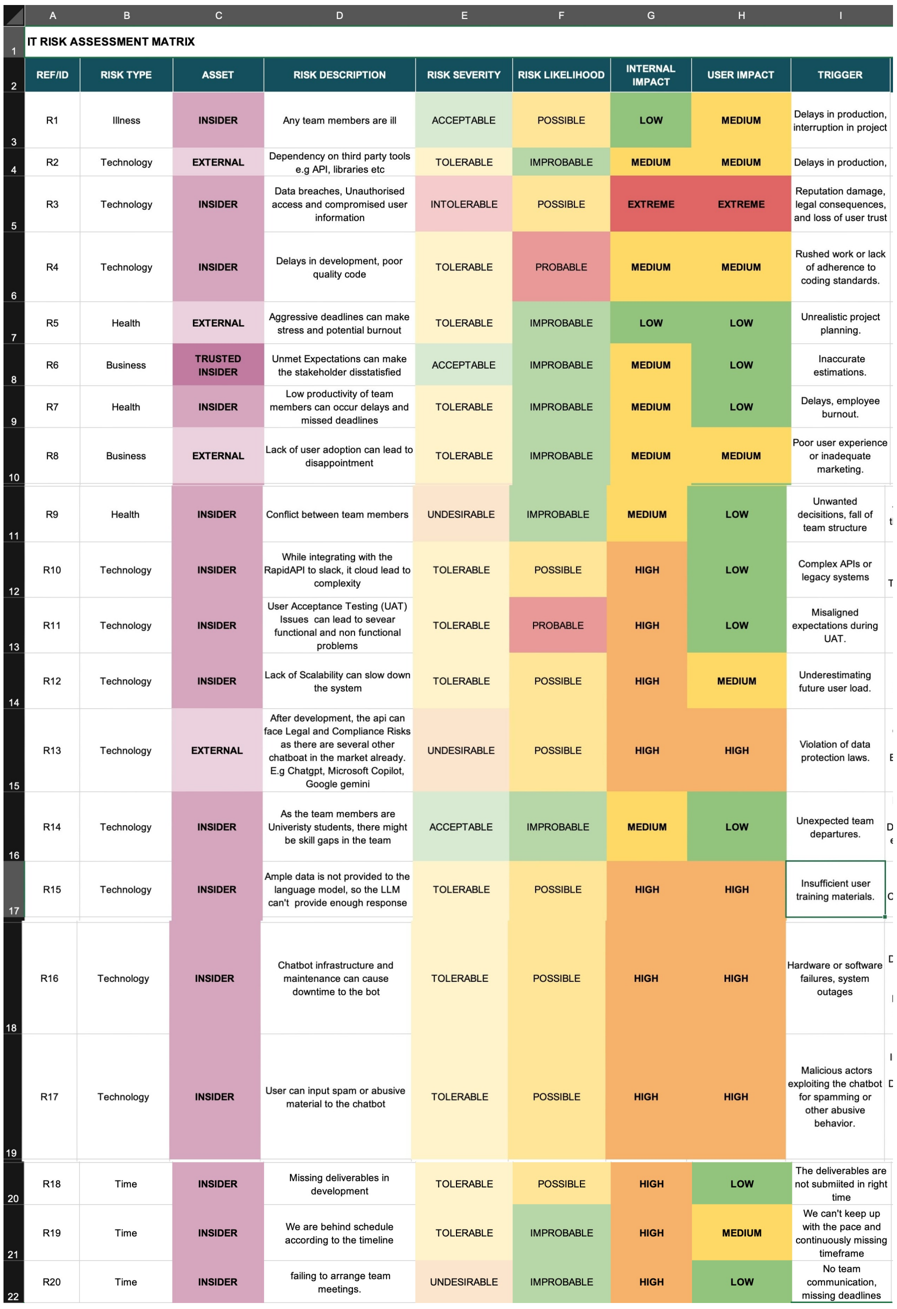
2.1.3 Organisation

1. Unmet expectations can make the stakeholders dissatisfied.
2. Insufficient resources can hinder project success.
3. Resistance to change may hamper the progress of the project.

2.1.4 Estimation

1. We miss deliverables in development.
2. We are behind schedule according to the timeline.
3. We continually fail to arrange meetings.

2.2 Matrix



IT RISK ASSESSMENT MATRIX

REF/ID	RISK TYPE	ASSET	RISK DESCRIPTION	RISK SEVERITY	RISK LIKELIHOOD	INTERNAL IMPACT	USER IMPACT	TRIGGER
R1	Illness	INSIDER	Any team members are ill	ACCEPTABLE	POSSIBLE	LOW	MEDIUM	Delays in production, interruption in project
R2	Technology	EXTERNAL	Dependency on third party tools e.g API, libraries etc	TOLERABLE	IMPROBABLE	MEDIUM	MEDIUM	Delays in production,
R3	Technology	INSIDER	Data breaches, Unauthorised access and compromised user information	INTOLERABLE	POSSIBLE	EXTREME	EXTREME	Reputation damage, legal consequences, and loss of user trust
R4	Technology	INSIDER	Delays in development, poor quality code	TOLERABLE	PROBABLE	MEDIUM	MEDIUM	Rushed work or lack of adherence to coding standards.
R5	Health	EXTERNAL	Aggressive deadlines can make stress and potential burnout	TOLERABLE	IMPROBABLE	LOW	LOW	Unrealistic project planning.
R6	Business	TRUSTED INSIDER	Unmet Expectations can make the stakeholder disstatisfied	ACCEPTABLE	IMPROBABLE	MEDIUM	LOW	Inaccurate estimations.
R7	Health	INSIDER	Low productivity of team members can occur delays and missed deadlines	TOLERABLE	IMPROBABLE	MEDIUM	LOW	Delays, employee burnout.
R8	Business	EXTERNAL	Lack of user adoption can lead to disappointment	TOLERABLE	IMPROBABLE	MEDIUM	MEDIUM	Poor user experience or inadequate marketing.
R9	Health	INSIDER	Conflict between team members	UNDESIRABLE	IMPROBABLE	MEDIUM	LOW	Unwanted decisitions, fall of team structure
R10	Technology	INSIDER	While integrating with the RapidAPI to slack, it cloud lead to complexity	TOLERABLE	POSSIBLE	HIGH	LOW	Complex APIs or legacy systems
R11	Technology	INSIDER	User Acceptance Testing (UAT) Issues can lead to sevear functional and non functional problems	TOLERABLE	PROBABLE	HIGH	LOW	Misaligned expectations during UAT.
R12	Technology	INSIDER	Lack of Scalability can slow down the system	TOLERABLE	POSSIBLE	HIGH	MEDIUM	Underestimating future user load.
R13	Technology	EXTERNAL	After development, the api can face Legal and Compliance Risks as there are several other chatboat in the market already. E.g Chatgpt, Microsoft Copilot, Google gemini	UNDESIRABLE	POSSIBLE	HIGH	HIGH	Violation of data protection laws.
R14	Technology	INSIDER	As the team members are Univeristy students, there might be skill gaps in the team	ACCEPTABLE	IMPROBABLE	MEDIUM	LOW	Unexpected team departures.
R15	Technology	INSIDER	Ample data is not provided to the language model, so the LLM can't provide enough response	TOLERABLE	POSSIBLE	HIGH	HIGH	Insufficient user training materials.
R16	Technology	INSIDER	Chatbot infrastructure and maintenance can cause downtime to the bot	TOLERABLE	POSSIBLE	HIGH	HIGH	Hardware or software failures, system outages
R17	Technology	INSIDER	User can input spam or abusive material to the chatbot	TOLERABLE	POSSIBLE	HIGH	HIGH	Malicious actors exploiting the chatbot for spamming or other abusive behavior.
R18	Time	INSIDER	Missing deliverables in development	TOLERABLE	POSSIBLE	HIGH	LOW	The deliverables are not submiited in right time
R19	Time	INSIDER	We are behind schedule according to the timeline	TOLERABLE	IMPROBABLE	HIGH	MEDIUM	We can't keep up with the pace and continuously missing timeframe
R20	Time	INSIDER	failing to arrange team meetings.	UNDESIRABLE	IMPROBABLE	HIGH	LOW	No team communication, missing deadlines

Figure 1: Risk assessment matrix with scores for severity, likelihood and impact assigned to each risk item, as well as identified potential triggers.

2.3 Mitigation

The following measures will mitigate the risks to the project in each identified area.

2.3.1 Technology

1. Dependency on third-party tools: Discuss the best solution with the team and client before proceeding, like using alternative libraries.
2. Data breaches, unauthorised access, and compromised user information:
   (a) Implement robust security practices.
   (b) Conduct regular vulnerability assessments and follow best practices.
   (c) Do not keep user information in the cloud or any other storage.
3. Delayed development with poor quality code:
   (a) Frequent code testing.
   (b) Resolve bugs and logical errors promptly.
   (c) Establish coding standards and best practices.
4. Complexity of integrating RapidAnalysis API with Slack:
   (a) Thoroughly understand integration requirements.
   (b) Test integrations early and often.
5. UAT issues:
   (a) Involve stakeholders in UAT planning.
   (b) Clearly define acceptance criteria.
6. Lack of scalability:
   (a) Design for scalability from the outset.
   (b) Regularly assess performance metrics.
7. Legal and compliance risks with other chatbots:
   (a) Consult legal experts team and stakeholders.
   (b) Ensure compliance with relevant regulations.
8. Skill gaps:
   (a) Ensure proper training for team members.
   (b) Document critical knowledge and everyone's strengths and skills to align with the project.
9. Unsatisfactory chatbot response:
   (a) Develop user-friendly documentation.
   (b) Conduct training sessions for end-users.

10. Chatbot downtime:
    (a) Implement a high-availability infrastructure for the chatbot.
    (b) Develop a robust monitoring and alerting system for potential issues.
    (c) Have a backup plan in place to restore functionality in case of failures.
11. Spam or abusive input:
    (a) Implement spam filters and user authentication mechanisms.
    (b) Define clear rules and guidelines for chatbot interaction.
    (c) Monitor chatbot activity for suspicious behavior and take corrective actions as needed.

2.3.2 Human

1. Illness, loss of contact, or inattendance: Another team member will get the handover and continue the task.
2. Withdrawal: The project plan changes and other team members will be allocated based on their strengths to fill the position.
3. Conflict: All team members will discuss the point of conflict, and the project manager will lead the discussion to find the solution.
4. Incomplete tasks due to lack of ability or motivation:
    (a) We are avoiding any excessive workloads in our project. We also have well-paced project planning.
    (b) Per our team structure and guidelines, if any team member cannot complete their assigned task, they must inform the team. Subsequently, the team collaborates with the project manager to find a solution and ensure the completion of any remaining tasks.

2.3.3 Organisation

1. Unmet stakeholder expectations: We have established clear communication channels and set realistic goals to address this issue. We also follow weekly updates and transparent reporting to manage expectations effectively.
2. Insufficient resources: Our approach involves comprehensively evaluating resources and prioritising tasks according to their importance for the project's success. In our feasibility Report, we have also discussed and recorded various alternative strategies to address the issue.
3. Resistance to changes: We have involved our stakeholders in the change process, providing them with the rationale behind changes and how they will benefit the project. We will also ensure training and support within our team to facilitate smoother transitions.

2.3.4 Estimation

1. Missed deliverables:
    (a) Conduct regular reviews to track progress and identify any deviations from the plan.
    (b) Ensure well-defined requirements to minimise misunderstandings and scope creep.
    (c) Have backup plans in place to address unexpected delays.
2. Behind schedule:
    (a) Set realistic timelines based on data and team capacity.
    (b) Allocate extra time for unforeseen issues or unexpected delays.
    (c) Keep stakeholders informed about any schedule changes promptly.
3. Failure to arrange meetings:
    (a) Set reminders to remember to arrange and attend Discord meetings.
    (b) Clearly define the purpose and agenda for each meeting.
    (c) Ensure all participants commit to attend the scheduled weekly meeting.

3 Resource Management

Resource management encompasses the efficient allocation, scheduling, and continuous monitoring of various resources.

3.1 Human

The team consists of six students under the consultancy name "InfoByte" working collaboratively on the project. Roles and responsibilities are allocated to team members effectively based on their strengths, skills, and availability. Team members are given one of three formal roles defined in the team manual: project manager, lead developer, and developer. Role rotation ensures that everyone experiences all three sets of responsibilities. Workloads are balanced by evenly assigning Jira tasks to team members.

3.2 Technology

There are no specific hardware resources other than standard development machines (laptops or desktops) belonging to the team members which have sufficient processing power, memory, and storage for the project. The system requirements for using Slack are MacOS 11 or above, Windows 10 version 21H2 or above, Ubuntu LTS releases 20.04 or above, or Red Hat Enterprise Linux 8.0 or above.

The software stack includes the following tools, which have been thoughtfully selected to ensure effective resource management across the team:

- IDE: Microsoft Visual Studio Code (VSCode), a versatile code editor with features like syntax highlighting, debugging, and extensions for enhanced productivity.
- Version control: Git for code, with GitHub to host the repository, and Google Docs and Overleaf to save and iterate on notes, drafts, and completed documents.

- Communication channels: Discord for messaging and meetings, and Slack to test the chatbot features.
- APIs: RapidAnalysis API for LLM capabilities and Slack API for Slack app development.
- Project management: Jira project for assigning tasks and viewing progress, using Kanban board and timeline views.

3.3 Information

The team uses formal documentation and the Discord server to save relevant information relating to the project. This includes the RapidAnalysis website, which demonstrates how the client expects their AI smart utilities and APIs to appear to users and developers, and important pages on Slack app development and AWS.

3.4 Time

The PM schedules weekly client meetings to discuss project progress and address client-specific requirements.

Weekly team meetings are a priority to understand upcoming deliverables and allocate tasks. These are held once or twice a week, considering the availability of team members.

Jira provides a timeline view of deliverables and subtasks. The PM prioritises tasks to ensure timely completion.

3.5 Finances

The RapidAnalysis API and Slack are currently free for the developers and users.

4 Change Management

Change management for this project involves

1. managing requirements and scope change, and
2. version control for code and documents.

4.1 Requirements and scope change

The team understands that change in the project is inevitable and must be communicated because of potential impacts.

For these reasons, it is crucial for the team to document and clearly define project requirements. The PM should regularly review and validate requirements with the client. Finally, the team expects that any change will be communicated during the weekly team meeting or in a Discord message, viewable to the rest of the team. Private, closed-door decisions contradict the team manual.

4.2 Version control

Developers are expected to document code changes accurately using Git, assisted by the LD who reviews pull requests and merges them into the GitHub project once accepted.

All deliverable documents are completed in collaborative environments: Google Docs and Overleaf. This ensures that team members iterate on the latest version and are able to view changes or restore an older version in case of accidental change.

5 Quality Management

The quality of this project depends on the communication within the team, dictated by the team manual with clearly defined roles and responsibilities, and with the client, whose feedback and expectations are received in weekly emails and/or meetings with the PM. Specific quality monitoring and assurance strategies will include the following.

1. Error checking mechanisms to ensure the chatbot operates smoothly and without errors, which could involve validating user inputs, checking for null and corner values, and handling unexpected responses from APIs or databases.
2. Performance optimisation by monitoring stress under different loads to ensure the chatbot can handle many simultaneous users or requests.
3. Comprehensive, up-to-date documentation to facilitate collaboration among developers, understand the system, and help in troubleshooting and maintenance.
4. Regular testing to ensure the chatbot works as expected, including unit testing individual components, integration testing the whole system, and user acceptance testing to validate the chatbot from the perspective of the end user.

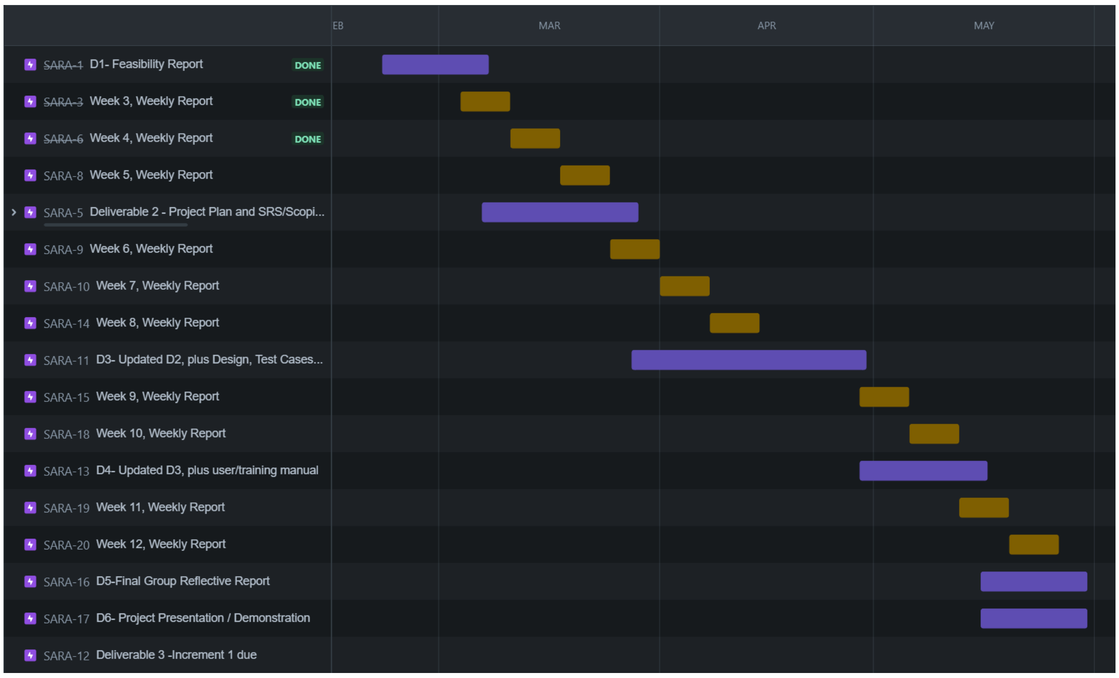
Tester fatigue should be taken into account when the developers assign Jira tasks to themselves and communicated with the LD, as per the communication policy for all technical tasks.

6 Project Schedule

The project and sprints are structured around the deliverables, which are summarised in Table 1. This is reflected in the timeline view of the Jira project in Figure 2.

Deliverable	**Due**
D1: Feasibility study and team manual	07/03/2024
D2: Project plan, requirements document, updated team manual	28/03/2024
D3: Updated documents and minimum viable product (MVP)	29/04/2024
D4: Updated deliverables and user manual	16/04/2024
D5: Final report	30/05/2024
D6: Presentation	30/05/2024
D7: Final product	30/05/2024
D8: Exam	TBC

Table 1: Scheduled deliverables for the project.



~~SARA-1~~ D1- Feasibility Report DONE
~~SARA-3~~ Week 3, Weekly Report DONE
~~SARA-6~~ Week 4, Weekly Report DONE
SARA-8 Week 5, Weekly Report
SARA-5 Deliverable 2 - Project Plan and SRS/Scopi...
SARA-9 Week 6, Weekly Report
SARA-10 Week 7, Weekly Report
SARA-14 Week 8, Weekly Report
SARA-11 D3- Updated D2, plus Design, Test Cases...
SARA-15 Week 9, Weekly Report
SARA-18 Week 10, Weekly Report
SARA-13 D4- Updated D3, plus user/training manual
SARA-19 Week 11, Weekly Report
SARA-20 Week 12, Weekly Report
SARA-16 D5-Final Group Reflective Report
SARA-17 D6- Project Presentation / Demonstration
SARA-12 Deliverable 3 -Increment 1 due

Figure 2: Timeline view of the Jira project, Slack App for RapidAnalysis (abbreviated to SARA). The orange bars indicate progress reports which are submitted once every week, excluding the recess. The purple bars represent sprints; clicking on these expands the details of that deliverable.

6.1 Tasks

As an agile team of 6 members with rotating roles in each sprint, a flexible approach to task allocation is crucial. The PM sets up specific issues under the deliverables on Jira; see the examples in Figures 3 and 4. There are also recurring core tasks for each sprint which the team must complete to ensure the development of the project, which are divided into the following areas.

6.1.1 Planning and Management

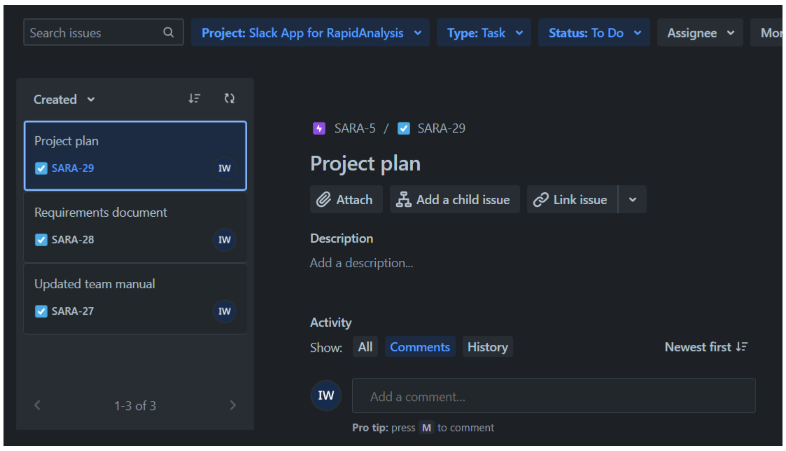
At the beginning of each sprint, the designated PM initiates communications with the client over email and arranges a meeting if necessary, and schedules a team meeting based on shared availability.

All team members attend the weekly meeting to collaboratively define sprint goals, share ideas and notes on the upcoming deliverables, and accept Jira tasks set by the PM.

The PM updates and maintains the Jira tasks, and checks that the team is on track to complete tasks for deliverables with input from the LD.

6.1.2 Development and Implementation

Each developer will focus on their assigned tasks, which could include building specific chatbot functionalities and integrating the chatbot into Slack.



Search issues | Project: Slack App for RapidAnalysis | Type: Task | Status: To Do | Assignee | Mor

Created

Project plan
SARA-29 IW

Requirements document
SARA-28 IW

Updated team manual
SARA-27 IW

1-3 of 3

SARA-5 / SARA-29

Project plan

Attach | Add a child issue | Link issue

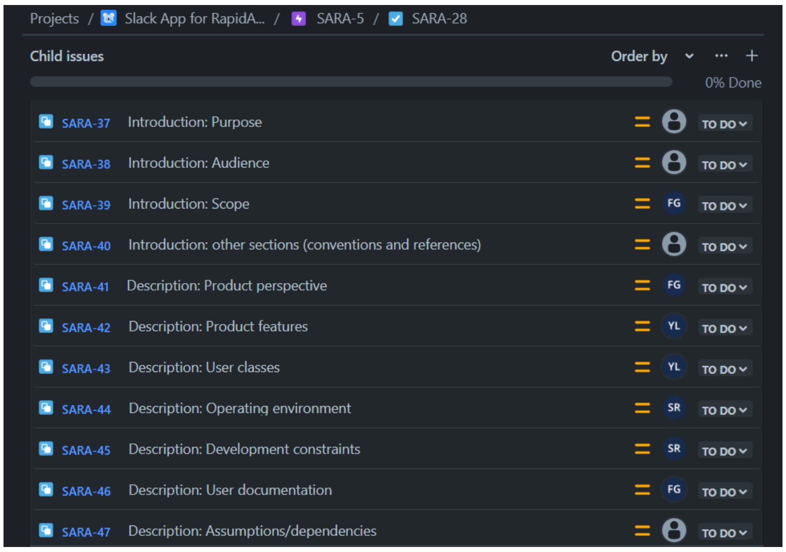
Description
Add a description...

Activity
Show: All | Comments | History — Newest first

IW Add a comment...

Pro tip: press M to comment

Figure 3: Example of a Jira task. The sidebar to the left shows all tasks listed under D2 (uniquely identified as SARA-5) which have all been assigned to the PM for that sprint (see initials).



Projects / Slack App for RapidA... / SARA-5 / SARA-28

Child issues — Order by — 0% Done

SARA-37	Introduction: Purpose		TO DO
SARA-38	Introduction: Audience		TO DO
SARA-39	Introduction: Scope	FG	TO DO
SARA-40	Introduction: other sections (conventions and references)		TO DO
SARA-41	Description: Product perspective	FG	TO DO
SARA-42	Description: Product features	YL	TO DO
SARA-43	Description: User classes	YL	TO DO
SARA-44	Description: Operating environment	SR	TO DO
SARA-45	Description: Development constraints	SR	TO DO
SARA-46	Description: User documentation	FG	TO DO
SARA-47	Description: Assumptions/dependencies		TO DO

Figure 4: Example of child issues listed under a task. Some team members have already assigned the task to themselves. The completion status can also be updated from "TO DO" to "IN PROGRESS" or "DONE".

The LD provides guidance to developers through paired programming, conducting code reviews, approving changes, and overseeing the overall technical direction of the project in accordance with the demands of the PM and, ultimately, the client.

6.1.3 Testing and Deployment

All team members can participate in beta testing, unit testing and integration testing for quality assurance. Most importantly, feedback for improvement should be given in a Discord message or Jira issue. The team will collaborate on deployment strategies, including choosing a Slack channel for user testing and establishing a feedback mechanism.

6.1.4 Documentation and Support

Throughout the project, everyone will contribute to documenting the development process. This includes deliverable documents like requirements and specifications and the user manual, as well as weekly progress reports. Using text-based communication channels like Discord messaging and Jira issue comments are also a form of documentation and are crucial for team members to support the overall project progress.

6.2 Resources

For the duration of the project, Team 25 is allocated the following key resources.

- 6 team members, assuming no unexpected circumstances like withdrawal from the project, with their own development devices
- Discord server and Jira project accessible to all team members
- GitHub repository provided by the client for developers to work on
- Amazon Web Services provided by the client

6.3 Assumptions

In completing the client's desired Slack application with RapidAnalysis AI smart utilities integration, it is assumed that our team will have access to various resources with sufficient reliability and performance. Access to the RapidAnalysis API and Slack API is assumed alongside the required documentation to make requests as desired. In addition, access to hosting services such as Amazon Web Services (AWS) will be provided by the client where required, as well as a Jira project for project management and a GitHub repository for a version control code base. The RapidAnalysis API will allow for information to be exchanged with sufficient data sizes such that long chat histories can be analysed by the AI utilities provided.

7 Appendices

1. Risk Management Template by Dan Martin of ClearRisk
2. Document Standard Header Template by InfoByte
3. SRS Template

Infobyte

Team 25

Software Requirements Specification

Slack app with RapidAnalysis API

Version 1.0 28.03.2024

Contents

1 Introduction

1.1 Overview

1.1.1 Document Conventions

For improved readability, this document uses numbered, bold headings and subheadings, lists, and captioned figures. Other specific conventions include the following.

- Code is written in monospace format like `code`.
- Proposed commands for users to access the chatbot in Slack are written as `/command`.
- The Definitions, Acronyms, and Abbreviations emphasises key terms in bold like **SRS**.

1.1.2 Intended Audience and Reading Suggestions

This document is intended for current and future project stakeholders, namely the client, RapidAnalysis, but also other teams or individuals who will continue to expand the scope of the Slack app, develop and test new functionalities, and add documentation. This document will help readers to understand the purpose, features, and functional requirements of the project. The authors have the following suggestions to readers.

- For a high-view understanding of the project, refer to the Introduction and Overall Description, which includes diagrams and hyperlinks to further resources.
- For deeper understanding of the product's functionality, see 3.1 Interact with RapidAnalysis API. For deeper understanding of the project requirements, see 3.3 Usability Requirements and 3.4 Other Nonfunctional Requirements.
- Read Client Feedback to understand how the expectations of the client informed this document.
- Readers should continuously refer to the Definitions, Acronyms and Abbreviations, which contains definitions of specialised terms and acronyms, to understand this document.

1.2 Purpose

The client, RapidAnalysis [1], is developing their AI smart utilities with Machine Learning (ML) endpoints for a wide range of use cases. Currently, users can create free developer accounts and interact with their AI smart utilities with a monthly limit of 5,000 API calls, but the AI smart utilities will eventually be monetised.

This Software Requirements Specification (SRS) version 1.0 describes the specific project of integrating RapidAnalysis capabilities in a Slack app. It communicates the intention of the project's development team, InfoByte, to other stakeholders, namely the client. It conforms to the ISO/IEC/IEEE 29148:2011 standard [2].

1.3 Project Scope

RapidAnalysis is focused on delivering customisable business solutions using their machine learning APIs. For this project InfoByte will be creating an AI summarising chatbot that can be downloaded to Slack user's workspace. Slack, a messaging application for organisation, has a dedicated Marketplace where users can download third-party applications [3]. After installing the application, a user will be able to interact with the AI bot to summarise the channels in their workspace, using RapidAnalysis's machine learning API to do so.

Slack users will be able to save time and confusion by utilising the AI bot. Instead of reading through a backlog of messages in multiple channels, the AI bot will be able to read the channel and give a comprehensive synopsis of each channel when requested. This increases productivity in an organisation, allowing users to spend more time on the actionables rather than backtracking and trying to understand what has been spoken about when they may not have been available.

Having the application available on Slack opens the doors for exposure and potential monetisation of the API for RapidAnalysis. For this project it is intended to have a functioning application that is deployed on the Slack Marketplace, but also to leave it in a state that can be further improved by other developers in the future.

1.4 Definitions, Acronyms, and Abbreviations

Acronyms:

- **AI** Artificial Intelligence
- **API** Application Programming Interface
- **AWS** Amazon Web Services
- **DM** Direct Message
- **HTTP** HyperText Transfer Protocol
- **HTTPS** HTTP Secure
- **JSON** JavaScript Object Notation
- **LLM** Large Language Model
- **ML** Machine Learning
- **OS** Operating System
- **PEP** Python Enhancement Proposal
- **PM** Private Message
- **REST** Representational State Transfer
- **SDK** Software Development Kit
- **SRS** Software Requirements Specification

1.5 References

[1] RapidAnalysis. "Rapidanalysis." Accessed on March 14, 2024. (Feb. 2024), [Online]. Available: https://rapidanalysis.github.io/.

[2] "ISO/IEC/IEEE international standard - systems and software engineering – life cycle processes –requirements engineering," *ISO/IEC/IEEE 29148:2011(E)*, pp. 1–94, 2011, Accessed on March 14, 2024. DOI: 10.1109/IEEESTD.2011.6146379.

[3] Slack. “Add apps, get work done.” Accessed on March 14, 2024. (), [Online]. Available: https://slack.com/intl/en-au/apps.

[4] Slack. “Unlock your productivity potential with Slack Platform.” Accessed on March 14, 2024. (Mar. 2024), [Online]. Available: https://api.slack.com/.

2 Overall Description

2.1 Product Perspective

Previously, RapidAnalysis has created an add-on for Google Sheets using their machine learning API and are looking to expand the applications of their technology. InfoByte has been tasked with implementing the machine learning API with Slack. Slack has its own API for developers to use when creating new applications [4], allowing them to upload the application to their (Slack) own marketplace where users can download the applications and extensions to their workspace.

The goal for this product is to have a summarising AI bot that can be downloaded from the Slack Marketplace. The scope of the application has been reduced to ensure that it will be executed well, leaving the project extendable for other developers to build on.

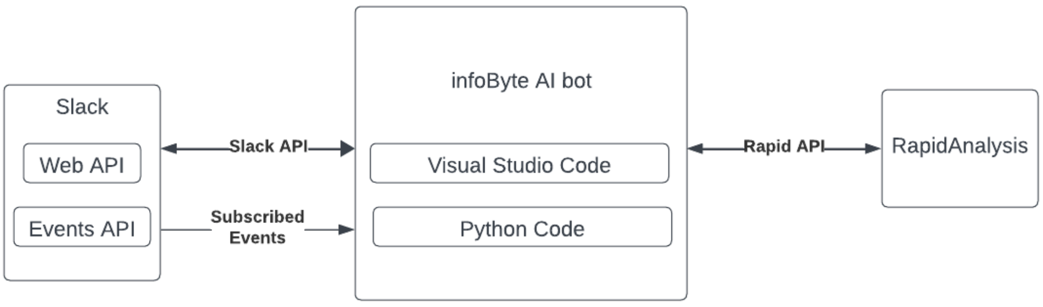


Figure 1: Architecture diagram showing how the chatbot, created with Python in VSCode, will integrate RapidAnalysis AI smart utilities in Slack.

2.2 Product Functions

Users can ask the chatbot questions about conversations in a Slack channel and it will provide a summary of information relevant to the user request based on the channel logs it has been given; see Figure 2. Developers can expand upon the functionality of the chatbot using the RapidAnalysis API.

2.3 User Classes and Characteristics

The target audience of Slack is any organisational employee, whether in small teams to multinational organisations, typically in “office” jobs or work-from-home roles. Freelancers, independent contractors, non-profit organisations, and student organisations may also use Slack.

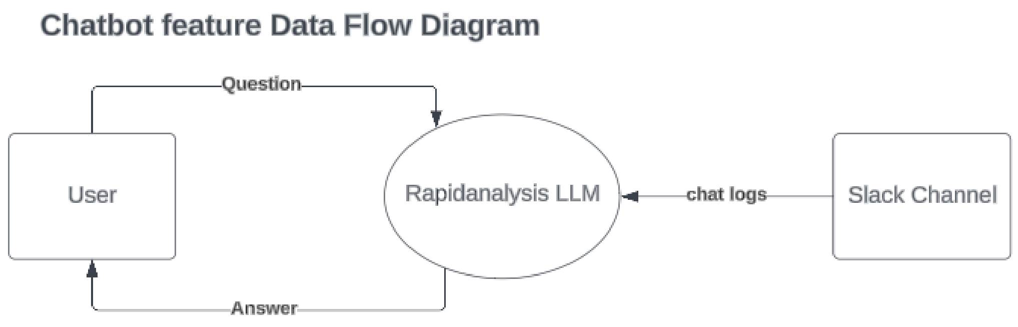


Figure 2: Data flow diagram showing how the user will interact with the chatbot, ultimately the RapidAnalysis LLM, which receives chat logs from a Slack channel.

Most of the users will be employees of organisations who already use Slack for communication within teams every day during working hours. They only need a basic understanding of Slack to download the app from the Slack App Directory and access the chatbot.

- As an employee, I would like to be able to automate message summarisation and search for information more easily in Slack channels so that I don't have to waste time finding it.
- As a team leader, I would like to ensure that my team is communicating clearly and efficiently, and maximise their understanding of my messages.
- As a business leader, I would like to use the summaries of conversations to make informed business decisions.

Some of the user base will be developers who would like to expand upon the app. This will be less common and less frequent. They will have a high level of technical expertise to be able to develop the app further.

2.4 Operating Environment

The software's only dependencies are Python version 3.12, and the support for HTTP to interact with the Slack and RapidAnalysis API. Thus the software requires the same hardware requirements as Python, where the OS can be Microsoft Windows, MacOS, or Linux.

The Python application will operate with 2 packages, and their dependencies, which are (in the form `'package_name' == 'version'`):

- `slack-bolt == 1.18.1`, for interacting with the Slack API.
- `requests == 2.31.0`, for creating HTTP requests, specifically for interacting with the RapidAnalysis API.

The software must run on the AWS Lambda Service, and thus must coexist with all its components.

2.5 User Documentation

The software will be delivered with a user manual and help guides. As the software is being launched on the Slack Marketplace, there is a dedicated space for a description and links under 'Learn more & support' to help users with the app. These links lead the user to the developer's website where there is documentation on how to use their software. RapidAnalysis has a dedicated section on their website for their Slack App information to be updated on the completion of the project and will be linked in the Slack App Directory description.

2.6 Assumptions and Dependencies

The team assumes the following for this project.

- Access to Slack API and corresponding documentation
- Access to RapidAnalysis API and corresponding documentation
- AWS Lambda Service for hosting the Python application, provided by the client
- Jira for project management
- GitHub repository

3 Requirements

3.1 Interact with RapidAnalysis API

3.1.1 Description and Priority

Priority: High.

Interacting with APIs is crucial for the application's core functionality. Without this feature, the application would lack the ability to integrate external services and functionalities, severely limiting its usefulness and value to users. Therefore, ensuring the successful implementation of API interactions is of utmost importance.

3.1.2 Stimulus/Responses Sequences

- User requests interaction with API through application interface.
- Application sends API request with relevant parameters.
- API processes request and returns response to application.
- Application receives response and presents it to the user.

3.1.3 Functional Requirements

The application must provide an interface for users to initiate interactions with external APIs.

Users should be able to specify parameters and data required for API requests. The application must handle authentication and authorization mechanisms required for securely accessing external APIs.

Error handling: The application should appropriately handle errors and exceptions returned by the APIs, providing meaningful feedback to the user.

Input validation: Ensure that user-provided data for API requests is validated to prevent invalid or malicious inputs.

Logging: The application should log API requests and responses for debugging and auditing purposes.

Asynchronous processing: If applicable, the application should support asynchronous API calls to prevent blocking user interactions during long-running operations.

REQ-1: The application should support various HTTP methods (GET, POST, PUT, DELETE) for interacting with APIs.

REQ-2: API responses should be parsed and formatted appropriately before presenting to the user.

REQ-3: Implement rate limiting mechanisms to prevent abuse and ensure fair usage of APIs.

REQ-4: Ensure compatibility with different API versions and specifications to accommodate changes and updates by API providers.

3.2 Summarising Messages and Asking Questions

3.2.1 Description and Priority

Priority: Medium.
This feature enables the system to summarise messages and ask questions based on the conversation context. It plays a pivotal role in enhancing communication efficiency and comprehension within the application.

3.2.2 Justification

The ability to summarise messages and ask relevant questions enhances user engagement and comprehension within the application. By providing concise summaries and prompting further discussion through questions, this feature contributes to a more interactive and productive user experience.

3.2.3 Stimulus/Response Sequences

Summarising Messages:

- The system monitors ongoing conversations.
- When prompted or at designated intervals, the system analyses the conversation content.
- Based on the analysis, the system generates a concise summary of the conversation's key points.
- The summary is presented to the user either within the conversation thread or as a separate notification.

Asking Questions:

- Upon detecting potential areas for clarification or further discussion, the system formulates relevant questions.
- These questions are designed to encourage users to elaborate on specific topics or provide additional context.
- Questions are seamlessly integrated into the conversation flow, prompting users to respond and engage further.

Message Summarization:

- The system must be able to analyze conversation content in real-time.
- It should identify and extract important information and key points from the conversation.
- Summaries should be generated accurately and efficiently, considering the context and relevance of the messages.

Question Generation:

- The system should have predefined templates or algorithms for generating questions based on conversation context.
- Questions should be formulated in a natural and conversational tone to encourage user engagement.
- The system must ensure that questions are contextually relevant and align with the ongoing conversation.

User Interaction:

- Users should have the option to enable or disable message summarization and question prompts.
- They should be able to provide feedback on the accuracy and usefulness of summaries and questions.
- The system should adapt its summarization and questioning strategies based on user preferences and feedback.

Integration and Compatibility:

- This feature should seamlessly integrate with other system functionalities, such as messaging interfaces and user profiles.
- It should be compatible with various messaging platforms and communication channels supported by the application.

Performance and Optimization:

- The system should be optimized for efficiency and scalability to handle large volumes of messages and conversations.
- Continuous monitoring and improvement mechanisms should be in place to enhance the quality and relevance of message summarization and question generation over time.

3.3 Design and Implementation Constraints

Hardware Limitations

- The application must be able to respond to a user's request in a reasonable amount of time to be perceived as real time.
- There exists no memory requirements, although a lightweight application is preferred to reduce server load and cost.

Interfaces to other Applications

- The application is required to access and interact with the REST APIs provided by RapidAnalysis and Slack.

Specific Technologies, Tools, and Databases to be used

- GitHub, for a version control code repository.

Parallel Operations

- The application must be able to handle multiple user requests simultaneously while interacting with both used APIs.

Language Requirements

- Python, for the programming language to be used in constructing the application.

Communications Protocols

- HTTP, when interacting with APIs using both HTTP POST and HTTP GET requests when needed.

Design Conventions or Programming Standards

- PEP8, a Style Guide for Python Code to conform to industry standards.
- PEP257, Docstring Conventions for Python to conform to industry standards.

3.4 Usability Requirements

3.4.1 User Interfaces

1. **Pop-up Chat (Help Bot):**
   - Users can interact with the chatbot using a trigger phrase like "`/infobyte`,".
   - When a user types "`/infobyte`," the chatbot replies with relevant information or assistance.
   - This approach allows users to quickly access help or information without leaving the current channel. So it acts like a shortcut.
   - Consider adding a consistent message prefix (e.g., "This message was generated by [Chatbot Name]") to distinguish chatbot responses from human messages.
2. **Private Message (PM) Interaction:**
   - Users can open a direct message (DM) with the chatbot.

- In the DM, users can send commands or queries to the chatbot.
- The chatbot responds directly in the DM, providing personalized assistance.
- Advantages: Privacy, focused interaction, and no interference with other channel conversations.
- We are considering implementing a command prefix (e.g., "`/help`" or "`/info`") to initiate chatbot interactions.

3. **Hidden Command Approach (e.g., `/infobyte`):**
   - Users can send hidden commands to the chatbot within regular conversations.
   - For example, a user might type, "I need information about project deadlines. `/infobyte`"
   - The chatbot detects the hidden command and responds accordingly.
   - This approach integrates chatbot functionality seamlessly into ongoing discussions.
   - Ensure that the hidden command is unique and unlikely to be used accidentally.
4. **Accessing Channel Message History:**
   - To provide context, the chatbot can ask users to refer to specific channels or threads.
   - For example, a user might ask, "What was discussed in #project-planning?"
   - The chatbot can retrieve relevant information from the specified channel's message history.
   - We are considering using Slack's API to access channel history programmatically.
5. **GUI Standards:**
   - **Sample Screen Images:**
     - We will be using Figma to create mockups or wireframes of our chatbot's interface. These visual representations help convey the layout, components, and interactions.
     - We will Define a consistent visual style for our chatbot consider Slack's existing design patterns and adhere to them.
     - We will Avoid cluttered layouts to prioritise essential content.
   - **Standard Buttons and Functions:**
     - The interface will Include standard buttons or interactive elements for common actions:
       1. **Help**: A button or command to access chatbot documentation or FAQs.
       2. **Cancel**: For interrupting ongoing processes.
       3. **Submit**: To confirm or submit user input.
       4. **Back/Next**: For multi-step interactions.
     - Use consistent labels and icons for these buttons.
   - **Keyboard Shortcuts:**
     - Slack supports keyboard shortcuts. So we will implement shortcuts for common chatbot actions:
       1. E.g., `/help` for accessing help, `/info` for information recovery.

3.4.2 Hardware Interfaces

Since the application being developed is intended to run within Slack, the hardware interfaces required will match that of the Slack requirements. Slack is a cross-platform application and is able to run on any hardware that supports a browser, or uses Linux, Windows, MacOS, Android, or iOS.

3.4.3 Software Interfaces

API Protocols

- **Slack API**: Used to integrate the chatbot with Slack, enabling message handling, events, and interactive features.
- **RapidAnalysis**: Used to make analytical responses for message handling and interactive features.

Database Connection: The message history will not be saved on any external database for security concerns.

Programming Languages: We will use the latest Python (3.10) version for our project. Python has a rich set of Natural Language Processing (NLP), and Machine Learning libraries. Moreover, the scalability to integrate with other technologies and frameworks made it suitable for our project.

Operating System

The application will be hosted on Amazon AWS for serverless web applications. We chose AWS according to our client's recommendation as it runs and manages compute resources automatically. It is also safe and secure to maintain and monitor the APIs.

Services and Communication

- **Authentication Service**: Ensuring secure communication is paramount for authentication. This service will be employed to confirm the chatbot's identity when accessing APIs.
- **Analysis Service**: This service aids the chatbot in submitting data analysis requests to the RapidAnalysis API and receiving the processed results during communication exchanges.
- **Result Generation Service**:Based on designated keywords, parameters, or data inputs, the chatbot will produce detailed analytical reports.

Data Sharing Mechanism

It involves transferring various data and messages between the chatbot and RapidAnalysis API.

- Input Data:
  - The user requests for data analysis or report generation using specific parameters such as text, commands, etc. The authentication credentials are kept secure and transmitted via API access.
- Output Data:
  - It is processed data that generates formatted output by analysis findings.
  - If there is a failed request, it will show error messages during processing.

Implementation Constraints

- **Global Data area**: Used to share common data such as user information, histories, configuration and machine learning models across different parts of the chatbot. It will be implemented in AWS cloud storage for concurrent access and data persistence. We will also implement security measures to protect sensitive information.
- **Secure transmission**: Data Transfer between the chatbot and Rapidanalysis API must attach to secure transmission protocols such as HTTPS to provide data confidentiality and integrity.
- **Data validation**: When the user inputs any data, it must be validated to prevent spam requests to ensure accurate output.
- **Error Handling**: The system will use a strong error-handling procedure to manage exceptions and unpredictable responses from the API.

3.4.4 Communication Interfaces

The project utilises two REST-compliant API's, namely the RapidAnalysis API, and Slack API, each with specifications as below. All HTTP methods sent to each REST API are served over HTTPS. RapidAnalysis API:

- The RapidAnalysis API methods use HTTP GET, POST, PUT, and DELETE, with a JSON payload of data pairs expected by the server. A response from the server will be structured by the following, where the output value is the expected output from the method.

```
{
     ...,
     "output": ...
 }
```

Slack API [4]:

- Slack's Web API uses HTTP POST, and GET methods for the application to access, though majority of the barebones HTTP requests are hidden behind the 'slack-bolt' Python package used for this project.
- The Event API allows for the implementation of a static web server to receive HTTP POST requests consisting of the event's JSON-based payloads from Slack on the application's event triggers. The AWS Lambda Service allows for an API Gateway to forward the event's payload in a consistent format to the application.

3.5 Other Nonfunctional Requirements

3.5.1 Performance Requirements

The chatbot should respond to user prompts within 5 seconds so that users perceive the chatbot as interactive, responsive and available. Longer than this may negatively affect the user experience.

If applicable, the application should support asynchronous API calls to prevent blocking user interactions during long-running operations. In case of other factors that are slowing the chatbot, it should notify the user after the 5-second mark.

Regular performance testing is necessary for this project. The team might also implement monitoring tools to track response times.

3.5.2 Security Requirements

Risk Identification and Management:

- The product needs to implement a process for identifying and managing risk associated with product usage.
- Develop contingency plans for potential threats both internally and externally to the system to address security issues.

Privacy Protection:

When collecting, storing, processing, and transmitting user's personal information. The product must take appropriate security measures to protect it.

User Identification and Authentication:

Introduce mechanisms to verify and authenticate the identity of users. This enhances account security and prevents unauthorized access.

Privacy and Terms of User

Personal Information Collection and Use:

- Clearly specify the purposes of use before collecting the user's personal information.
- Personal information should only be collected and used when legally required or with the user's consent

Information Security**:**

- Implement appropriate technical and organizational security measures to securely protect user's personal information.
- Implement security features such as data encryption, access control, and prevention of unauthorized access.

Terms of Use:

- Provide users with clear and easy-to-understand terms of use.
- The terms of use should include service terms, privacy policies, responsibilities, and liability limitations

The product should not introduce the risk of security issues for users.

- No user information or messages are stored by the app.
- Only the user can view their messages to and from the chatbot.
- The connection to the RapidAnalysis API is authenticated with an API key. Each user has their own API key which no one else should be able to access, like a password.

- The app does not store API keys directly in main code, but allows users to securely input it into environmental variables. The user manual should explain in more detail, and the app should guide users through this one-time setup process.

The terms of use include the following.

- The user is responsible for maintaining the confidentiality of their Slack account credentials and preventing unauthorised access to their devices logged into the accounts.
- The user is responsible for maintaining the confidentiality of their RapidAnalysis API key and preventing other agents from interacting with the LLM on their behalf.
- The user will follow the other Slack terms of use.

3.5.3 Software Quality Attributes

The quality attributes are derived from the target audience of the product. As explained in the User Classes and Characteristics section, the product is primarily for professionals who use Slack to communicate with others in their organisation, and developers may extend on this project in the future. Therefore the product must be easy to use (for Slack users) and extensible (for future developers who want to implement more functionalities).

1. Ease of use: The target audience (professionals using Slack for team communications) has a positive user experience. Users find that using the chatbot is intuitive. Within any Slack channel, users simply send the correct command from the messaging box. Ease of use also encompasses other aspects of usability, e.g. continuous availability is a prerequisite for use.
2. Extensibility: Developers can extend the project to integrate more capabilities into Slack. The product information encourages development and includes hyperlinks to highly readable and comprehensive developer guides. The codebase is well-documented and modular for easy understanding.

4 Other Requirements

Client feedback

Date and time: 10:00, 22/03/2024

Feedback received

1. Include monetisation opportunity for RapidAnalysis in Project Scope.
2. Include a short user story.
3. State the Python version and package requirements.
4. There are no corporate or regulatory policies for this project.
5. Recommendation: Host the Python code on a AWS Lambda Service which is a serverless endpoint. Alternative recommendation: Run an Amazon Elastic Compute Cloud virtual server instance which allows more flexibility but requires management of the API wrapper and firewall access.
6. Outline where and how users will store their RapidAnalysis API key (or maybe OpenAI Key in the future) using simple language for non-technical users.

Response

1. Project Scope notes the "potential monetisation of the API for RapidAnalysis".
2. User Classes and Characteristics presents brief user stories in bullet points.
3. Operating Environment states the required Python version, along with package requirements and their versions.
4. Omitted corporate and regulatory policies from Design and Implementation Constraints.
5. The team has chosen to use the AWS Lambda Service, and updated accordingly.
6. Security Requirements notes that users store their API key privately.

Infobyte

Team 25

Team Manual

Slack app with RapidAnalysis API

Version 2.0 28.03.2024

Contents

Revision history

Name	Date	Reasons for changes	Version
IW	16/03/2024	Updated Change Management and Conflict Resolution based on feedback	1.1
FG	19/03/2024	Updated Communication Plan and Meeting Schedule	1.2
IW	28/03/2024	Polished document for Deliverable 2	2.0

1 Team Structure and Organisation

The client has suggested a rotating leadership system with the following three team roles. Note that the listed responsibilities apply only to that sprint, enabling the flexibility of agile development.

1. The project manager (PM):
   - ensures smooth communication and collaboration within the team and with the client
   - meets with the client and lead developer early in the sprint
   - defines and schedules tasks using Jira
   - makes final revisions to the deliverable and submits it in a professional format, e.g. LaTeX
   - tracks the abilities and accomplishments of team members, providing necessary development or support to foster the overall success of the team
2. The lead developer (LD):
   - maintains and improves code quality through code reviews and the creation of coding guidelines
   - defines functional requirements, consulting the client, other team members, and research where necessary
   - reports to PM to create a realistic sprint schedule
   - clarifies tasks to developers
   - works closely with developers, e.g. through paired programming
   - approves Git changes
3. The developers:
   - report to the LD for all technical matters
   - flag any issues and points of confusion to the LD (if technical) and/or PM (if non-technical)
   - add themselves to specific tasks on Jira, according to their abilities
   - enhance their skills through continuous learning and new technologies
   - conduct code testing and debugging to minimise bugs and ensure product stability
   - share knowledge with each other and collaborate to achieve project goals

The team has agreed on the schedule in Table 1.

Deliverable	**Week**	**LD**	**PM**
D1: feasibility	1 2 3	Isabel	Yasmin
D2: plan, scope	4 5 6	Faith	Isabel
	7 8	Baz	Faith
D3: updates, prototype	– –	Ben	Baz
D4: updates, manual	9 10 11	Sebastian	Ben
D5-6: report, presentation	12 13	Yasmin	Sebastian

Table 1: The leadership schedule is designed around the deliverables. The LD for any given sprint will be the PM for the next sprint. This system supports a smooth transition into the PM role, since the LD works closely with the PM.

2 Team Values and ACS Code of Professional Conduct

Through the project, our team values and our methods and approaches emphasise how each individual and the team as a whole will follow the Australian Computing Society's Code of Professional Conduct regarding its aspects of Honesty, Trustworthiness, and Respect for both the profession and others.

Focusing on our team's dedication to honesty to allow for healthy interactions between each acting member of the project. Our team's interactions with the client, in particular the project manager who will act as a single point of contact, will be honest, open, and truthful regarding all parts of the project. Some examples of these include being transparent about the progress by allowing the client to oversee the project through Jira, the project management software being used, alongside the codebase, and having weekly meetings to confirm the client's requests are being met. Furthermore, each team member will be committed to ensuring every one of their responsibilities throughout the project's rotating roles, and notifying either the project manager or the PACE Unit Convenor if required.

Upholding human dignity, and ensuring the public welfare are critical principles for our team, dedicating ourselves to developing systems, and representing ourselves in a trustworthy manner. Each team member will hold themselves accountable, as well as the team as a whole, in all submissions, taking responsibility for their resulting successes and failures. This includes completing all assigned tasks for both the deliverables and project tasks as assigned by the project manager and lead developer, being accountable for the written documents and code submissions, and welcoming constructive criticism. Our team's determination in the proactive development of our skills and abilities is another team value we will uphold throughout the completion of this project, improving our programming practices, professionalism, and abilities to be trustworthy in performing our assigned roles and responsibilities. Each team member has listed their abilities along with aspects in which they have potential and desire to learn. In doing so, they can undertake work assigned to them where they have the necessary skills and are competent in order to strive for a quality product.

Aligning with the principle of respecting others outlined by the ACS required to be upheld by all its members as an ICT professional, our team will be supportive and respectful in all aspects of interactions. During any stage of the project requiring a problem resolution, each member will have an equal opportunity to provide suggestions and are encouraged to present their ideas. All suggestions and ideas will be treated equally, with the team requiring each team member to be included and in agreement on the outcome. Our team will strive to build a healthy workspace with encouragement towards the development of each other's skills. To achieve this, tasks submitted by team members will receive respectful and supportive feedback, including that of written components of the deliverables and during code reviews.

In addition to respect for others, our team values align with that of the ACS's code regarding respect for the profession, in supporting the industry and its public view. Our team seeks to develop additional tools for communication through this project, enhancing the quality of communications for teams using Slack. We aim to allow for faster and more effective understanding of conversations and events conducted within this text-based environment, improving the quality of life regarding professional communication specifically that which occurs within Slack. Furthermore, the creation of a supportive learning environment will allow for our team members to grow collectively and individually in enhancing our professional decision-making, and how they will affect the public and those around them. This will ensure that each member supports the ongoing development of an open and inclusive industry, in which others are encouraged and supported in advancing their knowledge and competence.

3 Project Management Approach and Tools

The team will use an agile approach, allowing adaptability to changes in the project. Based on the Scrum framework, short sprints will enable the team to iterate on our work, complete deliverables effectively, and rotate team roles which also prevents burnout and promotes shared ownership.

Using Jira as a project management tool facilitates this team approach and offers transparency as the client to view the progress of the team. Other tools and software ensure effective communication with the team and client, documentation, following a timeline, and delivering the product.

- Jira to assign tasks and visualise project progress in different views, like Kanban and Gantt chart timeline
- GitHub for version control
- Google Docs to collaboratively revise notes during meetings into document drafts
- Overleaf to collaboratively create professional-looking final documents
- Discord to communicate urgent messages and meet in voice calls at least once a week
- VSCode to write and analyse Python code
- Slack to implement and test the AI chatbot

4 Communication Plan and Meeting Schedule

Using when2meet, an online application that allows the team to enter their availability, the PM will schedule a meeting (1–2 per week). At the time of writing, it has been identified that Saturday mornings are the most frequently available to all team members, and therefore at least one weekly meeting has been established to happen at this time.

The Discord server has a dedicated channel for team members to propose questions to the client. Once per week, the PM will organise a meeting with the client where they will address these questions. It should be noted that it has been established that the PM will be the direct communicator with the client at all times.

As well as the channel for client communication, the Discord server has been formatted to include other channels such as documentation links, what skills each team member has, meeting minutes, and a general discussion channel where most communication happens. Discord is also being used as the meeting application. The PM plans the Discord event to take place in the server's voice channel, and screen sharing is utilised. During the meeting, one team member (usually the PM or LD) is assigned the responsibility of taking the meeting minutes and posting them to the designated channel.

The main goal of the weekly meeting is to discuss the current deliverable and any feedback or confusion within the team. How the tasks will be divided between the team is decided here as well. After the discussion, tasks will be created in the Jira and it is the responsibility of team members to ensure they check what they are assigned and/or to assign themselves to the appropriate tasks.

The weekly progress report to be posted on iLearn will be written by any one of the team members (this is also decided in the weekly meeting) summarising tasks and plans for the week per meeting. This enables team members to understand the overall project progress and take necessary actions.

5 Change Management

The project entails a number of variables. Given the democratic team structure, the general approach is that any team member can suggest a change on the Discord server, during weekly meetings, or by adding an issue or comment on Jira. The impact of the change on the project will be considered by the LD (for technical matters) and/or the PM (for non-technical matters). If the change is significant to the team and/or project, the PM will also inform the client. The outcome will be communicated on the Discord server. The following sections outline how the team will manage changes to specific aspects of the project.

5.1 Changes to code

GitHub is used for version control. Developers submit pull requests with adequate descriptions documenting the proposed changes. The LD reviews all pull requests, discusses any modifications with the developer, and approves changes by merging them into the main branch. The LD also maintains the README.md. Using Git, previous versions can be restored.

5.2 Changes to documents

Collaborative Google documents allow any team member to suggest changes or restore previous versions. Changes to documents are tracked with version numbers in the header and a revision history. The team transfers their work into a shared Overleaf project to apply LaTeX formatting to all deliverable submissions.

5.3 Changes to team members

Team members are expected to use the when2meet schedule and Discord server to communicate changes in their personal circumstances that may affect their ability to contribute to the project. The PM must be aware of human resources and take this into account when assigning Jira tasks. Team members raise issues on the Discord server and/or in comments under Jira tasks.

5.4 Changes to requirements, deadlines, and processes

Changes to project requirements, deadlines, or team processes are relevant to all team members and therefore must be announced through the Discord server and/or weekly team meeting.

Requirements from the client may evolve over the course of the project. This is managed through the close working relationship between the PM and client. The PM gathers feedback and clarifies expectations during meetings and through emails. The team can then adapt to the client's requirements, if reasonable. If the team reaches a consensus that a demand is not viable, the PM will work to reach an alternative agreement with the client as soon as possible.

Deliverable dates are "hard deadlines" that cannot be missed. On the other hand, preferable but non-essential "soft deadlines" could include sending a draft document to the client for feedback by a specified date. The PM tracks progress by viewing the completion status of Jira tasks and communicates with the team. This ensures that deadlines are met or that soft deadlines are revised promptly.

6 Conflict Resolution

This section outlines ways to prevent, mitigate and resolve dysfunctions in team trust, conflict management, commitment, accountability, and project results.

6.1 Context

All team members face limitations of work and study commitments, but are motivated to follow the deliverable schedule and develop themselves professionally throughout this project. Each team member has an individual set of skills and circumstances, and may need more support in certain weeks than others. Moreover, the project is adaptable and its requirements may evolve, requiring a flexible, agile team approach.

6.2 Approach and expectations

Given this context, the team has agreed to a democratic structure and rotating leadership schedule, ensuring flexibility and a shared sense of ownership.

This approach requires all team members to actively use the Discord server and attend weekly meetings. For example, team members can communicate their competencies, capabilities and limitations during the meeting or using the Discord channel dedicated to "skills and abilities", especially if they find out they will be busy with work, study, or personal matters. Concerns and suggestions are raised on the Discord server or during weekly meetings. This approach to communication reduces the risk of conflict within the team.

Leadership roles are distributed between the PM and LD. A leadership role does not grant that team member the ability to make decisions without the input of other team members. Rather, the PM is required to coordinate the team and raise issues during meetings or on the Discord server that require agreement from all members. Decisions are made collaboratively with input from all team members. The current PM cannot unilaterally reverse previous team decisions but only confirms team decisions or makes a final decision in case of a split team. However, it is a technical issue, the PM should pass on the responsibility to the LD to make a final decision. For example, the LD might invite team members to cast a vote on an urgent technical issue by pinging the team on Discord and asking for a response, and a tie would require the LD to make the final decision and inform the team of what they have decided. Continued disagreement should be resolved as quickly as possible by the PM, with input from the client if necessary, especially if there is conflict between the PM and LD.

All team members are bound to the rotating leadership schedule, roles and responsibilities, and ACS code of conduct as described in this document. Team members collaborate to complete deliverables, remain open to learning, follow the processes defined in this manual, demand accountability, promptly communicate all concerns especially relating to conflict and ethical issues, and understand each other's strengths and limitations. Communication records and documentation with version control can be used to keep team members accountable. For example, if a team member previously stated that they would complete a task by a deadline and failed to do so, other team members are right to demand better communication from them.

6.3 Ethics issues

All team members must be aware and engaged in flagging any potential ethical concerns that may develop during the project, through the following.

- Awareness and vigilance:
  - Maintain a heightened awareness of ethical considerations across all project phases, encompassing data management, decision-making processes, and stakeholder engagement.
  - Proactively identify potential ethical dilemmas, including seemingly insignificant or ambiguous situations.
- Timely communication:
  - As soon as any member faces a situation that involves ethical dilemmas or worries, inform the team.
  - Share ethics issues with the PM or the assigned team leader, giving specific information and background to help them comprehend and solve the problem.
- Cooperative dialogue:
  - Participate in honest and courteous conversations within the team to examine the ethical aspects of the issue.
  - Seek different viewpoints and feedback from team members to achieve a thorough understanding of the ethical consequences involved.

- Solution and prevention:
  - Collaborate to devise suitable strategies for dealing with the ethics issue in a way that maintains professional norms and ethical values.
  - Apply prevention measures or changes to project plans as needed to ensure ethical soundness and adherence.
- Record and review:
  - Record the ethics issue, including the specifics of the situation, conversations, choices, and actions taken for resolution.
  - Review the resolution process to ensure that the ethics worry has been sufficiently resolved and prevented.