CEBU INSTITUTE OF TECHNOLOGY UNIVERSITY

COLLEGE OF COMPUTER STUDIES

Software Project Management Plan

for

CyberKids

Change History

Version	Date	Amendment	Author
1.0	3/20/2025	Initial	Baguio Cultura Dedumo Vequiso
2.0	9/15/2025	The final SPMP has been amended to reflect the actual processes, tools, and team contributions during Capstone 2. Key changes include updated roles, realistic schedules, a switch from Azure to Railway, refined budget details, and alignment with our Agile iterative approach. This version accurately represents how the *CyberKids* project was planned, managed, and completed.	Cultura Dedumo

TABLE OF CONTENTS

1. Overview

1.1 Project Summary

1.1.1. Purpose, scope and objectives

Purpose

The purpose of this Software Project Management Plan (SPMP) is to define the strategies, processes, responsibilities, and deliverables necessary to guide the development, deployment, and evaluation of the CyberKids system. This plan serves as a blueprint for managing the entire project lifecycle—from requirement gathering to system maintenance—ensuring that the software is delivered on time, within scope, and according to quality standards.

CyberKids is designed to address the increasing need for engaging, age-appropriate cybersecurity education tools for elementary students, particularly those in Grades 5 to 6. As children are now exposed to digital environments at an earlier age, they become more vulnerable to threats such as phishing, weak password habits, oversharing of personal information, and online scams. The project's primary purpose is to use game-based learning to instill strong cybersecurity practices and awareness in young learners, leveraging the Roblox platform for its accessibility and popularity among the target demographic.

This SPMP aligns all stakeholders—students, educators, developers, project managers, and evaluators—toward a shared understanding of the project's goals, structure, deliverables, risks, and management processes. It ensures the development team adheres to a well-defined methodology (Agile), manages tasks effectively, mitigates risks, and maintains consistent communication throughout the project.

Scope

The scope of **CyberKids** encompasses the design, development, deployment, and evaluation of a gamified cybersecurity education platform comprising four core components:

1. **Interactive Game Levels in Roblox Studio**
    - Three immersive and educational mini-games targeting key cybersecurity concepts:
        - **Online Privacy** (e.g., what personal data is safe to share)
        - **Password Security** (e.g., creating strong, memorable passwords)
        - **Phishing Awareness** (e.g., identifying scams and suspicious links)
    - Features: timers, scoring systems, and dynamic leaderboards to encourage student engagement and competition.
2. **Real-Name Registration System**
    - A secure student registration process that links each user's real name to their Roblox user ID.
    - This allows teachers to monitor performance and assign learning tasks to identifiable students.
    - Includes encryption and data protection mechanisms to safeguard student information.
3. **Teacher Web Dashboard**
    - A React-based web application enabling teachers to:
        - Monitor student progress in real-time.
        - View analytics and reports.
        - Create and manage classes.
        - Assign or reassign levels.
        - Authenticate via Microsoft Teams for secure access.

4. **Real-Time Communication System**
    - Push notification system that informs teachers of key events, such as:
        - When a student registers.
        - When a student completes a level.
        - Real-time online/offline student status updates.
        - Alerts for classroom monitoring or student intervention.

Out-of-Scope Features (Not included in this project version):

- Support for non-Roblox-based platforms or mobile apps.
- Full curriculum integration or alignment with national K-12 learning standards (beyond cybersecurity).
- Voice-based chat moderation or AI-driven content moderation in-game.
- Multi-language support (the project is currently in English only).

System Components

Backend Server (Spring Boot):

- Acts as the core logic and data processing unit for the system.
- Handles API endpoints for user registration, score submission, leaderboard management, and teacher dashboard data.
- Ensures secure data storage, encryption of sensitive student information, and real-time notifications.

Frontend Web Dashboard (ReactJS):

- Provides teachers with a modern, responsive interface to monitor student progress, generate reports, manage classes, and receive real-time alerts.
- Supports Microsoft Teams authentication for secure login and access control.

Game Engine (Roblox Studio):

- Utilized to build the three interactive cybersecurity learning game levels focused on Online Privacy, Password Security, and Phishing Awareness.
- Roblox Studio enables creation of age-appropriate, engaging gameplay mechanics familiar to children within the target age group.

Platform as a Service (PaaS):

- **Vercel:** Hosts the ReactJS frontend dashboard, offering seamless continuous deployment and scalability.

- **Render:** Manages deployment of the Spring Boot backend services, ensuring reliability, automated scaling, and easy maintenance.

- **Railway:** Supports additional backend infrastructure needs such as database hosting, webhooks, and messaging queues to facilitate real-time communication and data synchronization.

Objectives

Main Objectives:

1. **Develop Gamified Cybersecurity Game Levels** that teach essential digital safety topics—such as online privacy, password security, and phishing awareness—through interactive, challenge-based activities designed to increase engagement and knowledge retention.

2. **Implement a Student Real-Name Registration System** that links each student's real identity to their in-game activity, allowing for personalized progress tracking while ensuring data privacy and security through encrypted storage and restricted access.

3. **Create a Teacher Web Dashboard** that allows educators to monitor student performance, access progress reports, and manage classroom-level controls. The dashboard serves as a central tool for teachers to oversee learning outcomes and adapt instruction based on student needs.

4. **Enable Real-Time Communication and Notifications** between the game environment and the teacher dashboard. This system ensures that teachers receive immediate updates on key student actions, such as level completions and registrations, supporting timely feedback and improved classroom interactivity.

Specific Objectives:

1. **Deliver a Fully-Functional Cybersecurity Education Game on Roblox Studio**
    - Develop and launch three interactive game levels, each emphasizing a critical cybersecurity theme.
    - Integrate dynamic gameplay features such as timers, scoring, and leaderboards to motivate engagement.

2. **Student Real-Name Registration**
    - Implement a registration process where students input their real names upon first login.
    - Link each real name securely to a unique Roblox account to ensure accurate student identification.
    - Enable tracking of individual student progress across game levels using their registered names.
    - Encrypt and securely store student names in the backend, accessible only by authorized teachers via the dashboard.

3. **Build a Responsive ReactJS Teacher Dashboard Hosted on Vercel**
    - Provide real-time visualization of student data and notifications.
    - Enable class management, student monitoring, and report generation.
    - Integrate Microsoft Teams OAuth for teacher authentication.

4. **Ensure Real-Time Communication and Notification**
    - Provide teachers with real-time notifications regarding key student activities (e.g., level completions, registrations).

1.1.2. Assumptions and constraints

The planning and execution of the **CyberKids** project are based on a set of critical assumptions that underpin the project's feasibility and success. At the same time, the project must operate within defined constraints—technical, operational, legal, and environmental—that limit its scope, design choices, and implementation strategies. Recognizing these assumptions and constraints helps the team manage risks effectively and set clear expectations for stakeholders.

Assumptions:

The following assumptions are considered essential and are expected to remain valid throughout the project lifecycle:

1. **User Hardware Access:**
   It is assumed that students and teachers have access to desktop or laptop computers capable of running Roblox Studio (for students) and modern web browsers (for teachers). The hardware should meet minimum performance requirements, such as a multi-core CPU and at least 4 GB of RAM.

2. **Basic Digital Literacy of Users:**
   Students in Grades 5 to 6 are presumed to have basic digital skills, including operating a keyboard and mouse, navigating graphical user interfaces, and understanding simple instructions provided during gameplay and dashboard use.

3. **Stable Internet Connectivity:**
   The system requires a reliable internet connection during use to support real-time features such as leaderboard updates, score synchronization, and dashboard notifications. It is assumed that both students and teachers have access to stable broadband or school network connections during activity sessions.

4. **Teacher Engagement and Support:**
   Teachers are expected to actively incorporate **CyberKids** into their classroom curriculum and utilize the teacher dashboard regularly to monitor student progress, manage access, and communicate effectively.

5. **Backend Service Availability:**
   The backend services hosted on Render and Railway platforms are assumed to have high uptime and availability, especially during typical school hours (e.g., 8 AM to 5 PM), ensuring seamless interaction between the game, dashboard, and server.

6. **Compliance with Security and Privacy Standards:**
   The project assumes that all data handling will comply with applicable local laws and regulations (e.g., the Philippines Data Privacy Act of 2012), including encrypted transmission and secure storage of sensitive student data.

7. **Roblox Platform Stability:**
   It is assumed that the Roblox platform will continue to support necessary APIs and integrations required for game development and communication with external backend services throughout the project duration.

Constraints:

The project must operate within the following limitations that define its boundaries and affect design and deployment decisions:

1. **Platform Support Constraint:**
   The game is developed exclusively for the **Roblox desktop client** (Windows and macOS). There is no official support or optimization for mobile devices, tablets, or web-based Roblox play modes, which may limit accessibility for users with only mobile devices.

2. **Hardware Performance Constraints:**
   The game and dashboard are designed to run efficiently on low- to mid-range computing hardware commonly available in schools and homes, but complex 3D graphics or resource-heavy features are avoided to ensure smooth performance.

3. **Data Privacy and User Anonymity:**
   To comply with data protection regulations, the system encrypts all personally identifiable information (PII), such as real names, and restricts access to authorized educators only. No third-party data sharing or public disclosure of personal information will occur.

4. **Limited External System Integration:**
   The system is self-contained and does not integrate with external social media platforms, third-party educational systems, or external authentication providers beyond Microsoft Teams for teacher login. This simplifies security management but restricts interoperability.

5. **Development Timeframe and Resource Constraints:**
   The project timeline limits the development scope to three core game levels and a functional teacher dashboard. Additional game features, multi-language support, or advanced AI components are out of scope for this phase.

6. **Network Dependency and Latency:**
   Real-time features such as leaderboards and notifications depend on network quality. High latency or intermittent connectivity may degrade user experience during gameplay or dashboard interactions.

7. **Compliance with Roblox Platform Policies:**
   All game content, data collection, and communications must comply with Roblox's Terms of Service and Community Standards, which restrict certain types of content and interactions.

8. **Teacher Dashboard Authentication Constraint:**
   The teacher dashboard uses Microsoft Teams OAuth for authentication, limiting access to educators with valid Microsoft Teams accounts within participating schools or districts.

1.1.3. Project deliverables

The CyberKids development team will provide the following project deliverables in accordance with the defined scope and goals of the capstone research project. These outputs represent all core components of the system—including the game itself, supporting software, documentation, and evaluation tools—required for demonstration, classroom implementation, and future enhancement.

Software Project Management Plan (SPMP)

This document serves as the formal project management guide, detailing the project scope, objectives, Agile-based methodology, sprint planning, team responsibilities, timelines, and resource allocation. It also outlines risk identification and mitigation strategies, communication protocols, and quality assurance measures to ensure consistent progress and accountability throughout the development lifecycle.

Gantt Charts and Schedules

Project timelines, sprint schedules, and task assignments are maintained and visualized using **ClickUp**, a project management platform that supports Agile workflows. ClickUp is used to:

- Define and assign development tasks
- Track sprint milestones and deadlines
- Manage task dependencies and team responsibilities
- Monitor progress in real time across all phases of the project

Milestone Tracking Systems

All major project milestones and adviser feedback checkpoints are documented and tracked using two platforms:

- **QueueIt** – Used for logging submission milestones and aligning deliverables with academic deadlines
- **SPEAR** – Serves as a submission and validation system where incremental progress reports are formally uploaded and reviewed by advisers and panelists

These tools help ensure that academic requirements and timelines are closely followed.

Increment-Based Weekly Progress Reports

Instead of traditional weekly reports, **increment-based reporting** is implemented. Weekly updates are submitted to the adviser during scheduled consultations. Each report contains:

- Updates on completed development tasks
- Demo or walkthrough of newly implemented features
- Ongoing blockers or technical issues
- Planned next steps for the upcoming increment
  This process enables continuous feedback and allows adjustments based on adviser input or evolving requirements.

Code Repository and Version Control

All source code for the system—including the Roblox game levels, backend (Spring Boot), and teacher dashboard (React.js)—is managed through a centralized **GitHub repository**. The repository supports:

- Collaborative coding via branches and pull requests
- Version control and rollback capability
- Issue tracking and documentation via README and wiki
- CI/CD integration support for future enhancements

GitHub ensures development transparency, version integrity, and accountability among team members.

Final Documentation and Handoff

Final Capstone Project Report

- A comprehensive narrative of the entire project journey, including background research, development process, testing outcomes, challenges faced, and reflections on project impact.

Project Handoff Package

- A bundled resource containing:
    - Setup and deployment instructions for local and cloud environments
    - Maintenance and update notes
    - Recommendations for scalability, new features, or integration into school IT infrastructure

All deliverables will be organized in a shared Google Drive folder and mirrored (when applicable) on a private GitHub repository. Final documents will be submitted in PDF format, with editable source files included. Sensitive student data (e.g., real names) will be anonymized or encrypted to comply with institutional privacy guidelines.

1.1.4. Schedule and budget summary

Schedule Summary

The project schedule identifies key milestones and their respective start and completion dates. This timeline ensures that all deliverables are completed on schedule to meet stakeholder expectations.

Milestone	Start Date	Completion Date
Project Planning	July 1, 2025	February 7, 2025
Requirements Gathering	July 11, 2025	February 13, 2025
System Design	July 18, 2025	June 25, 2025
Game Development (Roblox)	April 1, 2025	September 27,2025
Backend Development	March 20, 2025	September 20,2025
Dashboard Development	April 1, 2025	September 7,2025
Integration & Testing	September 18, 2025	*Ongoing*
Initial Deployment	September 28, 2025	*Ongoing*
Feedback & Iteration	*Upcoming*	*Upcoming*
Final Presentation	*Upcoming*	*Upcoming*

Budget Summary

The budget summary outlines the estimated and actual costs incurred during the project, highlighting resource allocation and cost control measures. Significant cost savings were realized by transitioning from Azure hosting in Capstone 1 to Railway hosting in Capstone 2.

Item/Resource	Description	Estimated Cost (₱)
Roblox Studio	Game development platform for interactive levels	Free
ClickUp (Student Plan)	Project management and sprint tracking	Free
GitHub	Code repository and version control	Free
VS Code + Extensions	Code editor for frontend/backend development	Free
Spring Boot + Java SDK	Backend development tools	Free
React.js	Frontend framework for dashboard	Free
Railway Hosting (Capstone 2)	Cloud hosting for backend and database	₱500
Azure Hosting (Capstone 1)	Initial backend hosting and database (prior phase, not in used anymore)	₱2,000

1.2 Evolution of plan

This plan outlines the progressive versions of the Software Project Management Plan (SPMP) throughout the development of *CyberKids*. Each version reflects significant milestones, team collaboration, and feedback incorporation. It also specifies how the SPMP evolved from initial drafting to its final and post-deployment form.

Version	Primary Author(s)	Description of Version	Date Expected
Draft	Jesson Cultura	Initial draft created and submitted for internal team review	March 20, 2025
Preliminary	Jesson Cultura Emmanuel Dedumo Ashley Josh Vequiso John kenneth Baguio Jhudiel Artezuela	Second draft incorporating adviser's initial feedback	April 5, 2025
Prerelease	Jesson Cultura Emmanuel Dedumo Ashley Josh Vequiso John kenneth Baguio	Finalized for formal submission to adviser before Capstone 1 defense	June 25, 2025
Final	Jesson Cultura	Approved version placed under configuration/change control. Updated after development completion to reflect actual implementation	September. 25, 2025

2. References

[1] ReactJS, *"Glossary – React Documentation,"* ReactJS.org, [Online]. Available: https://legacy.reactjs.org/docs/glossary.html. [Accessed: Sep. 25, 2025].

[2] Roblox, *"API Reference – Roblox Creator Documentation,"* Roblox Developer Hub, [Online]. Available: https://create.roblox.com/docs/reference/engine. [Accessed: Sep. 25, 2025].

[3] National Institute of Standards and Technology (NIST), *NIST Cybersecurity Framework*, [Online]. Available: https://www.nist.gov/cyberframework. [Accessed: Sep. 25, 2025].

[4] *CyberKids Software Requirements Specification*, Version 1.2, 2025. [Online]. Available: https://drive.google.com/file/d/1SoJI7GnMo5U7xIbmJaK-ljxp1E_MQijm/view?usp=drive_link. [Accessed: Sep. 25, 2025].

[5] *CyberKids Software Design Description*, Version 1.1, 2025. [Online]. Available: https://drive.google.com/file/d/100GV1pGwGB6XcJXuXjJjjk6wDcOHR7P8/view?usp=drive_link. [Accessed: Sep. 25, 2025].

[6] Republic of the Philippines, *Data Privacy Act of 2012 (RA 10173)*, Official Gazette, 2012. [Online]. Available: https://www.privacy.gov.ph/data-privacy-act/. [Accessed: Sep. 25, 2025].

3. Definitions

This section of the Software Project Management Plan (SPMP) provides clear definitions of key terms, acronyms, and technologies referenced throughout the document. These terms are critical for understanding the scope, execution, and technical components of the CyberKids project.

Definitions and Acronyms

- **API (Application Programming Interface):**
  A set of protocols and tools for building and interacting with software applications, allowing different systems to communicate with each other.

- **DBMS (Database Management System):**
  A software system designed to manage databases, providing an interface for data storage, retrieval, and manipulation.

- **UI (User Interface):**
  The space where interactions between humans and machines occur. In this context, the user interface refers to both the game's in-game interface and the teacher's web-based dashboard.

- **Spring Boot:**
  A Java-based framework used to create stand-alone, production-grade web applications. It is used in the backend of CyberKids to develop RESTful APIs and integrate with the database.

- **React.js:**
  A JavaScript library used to build user interfaces, particularly for single-page applications. React.js powers the frontend of the teacher's web dashboard in CyberKids.

- **Lua:**
  A lightweight, high-level scripting language used for creating game logic and interactive components within the Roblox platform.

- **Roblox:**

  An online game platform and game creation system that allows users to design and play games. CyberKids is developed and deployed on this platform.

- **MySQL:**

  An open-source relational database management system that uses SQL (Structured Query Language) for managing and manipulating data. It is used in CyberKids as the primary database for storing game data and user information.

4. Project organization

This section defines the organizational structure of the *CyberKids: A Digital Safety and Cybersecurity Awareness Game using the Roblox Platform* project. It identifies the internal project team roles, outlines responsibilities, and describes interactions with key external stakeholders and systems.

4.1 External structure

The CyberKids project interacts with several external entities that contribute to development guidance, compliance, testing, and deployment:

• Academic Institution (Cebu Institute of Technology – University)

CIT-U provides the academic framework, technical oversight, and resource access necessary for project completion. The project is conducted in compliance with the university's research ethics and capstone guidelines.

• Faculty Adviser / Project Panel

The Faculty Adviser serves as both mentor and evaluator. They guide the technical and pedagogical direction of the project, review milestone deliverables, and participate in sprint evaluations. Final grading and approval are conducted by the Capstone Panel.

• Pilot Stakeholders (Teachers and IT Coordinators)

Teachers and IT staff assist in validating the educational value of the game. They participate in the pilot testing phase and provide feedback on gameplay relevance, interface clarity, scoring fairness, and curriculum alignment.

• Roblox Platform

As the primary game development and deployment environment, Roblox imposes development constraints, moderation policies, and platform guidelines. All features must comply with Roblox's Terms of Service, content rules, and technical limits for public deployment.

4.2 Internal structure

The CyberKids project team consists of students from Bachelor of Science in Information Technology at Cebu Institute of Technology – University. The following outlines the key roles and their responsibilities within the team:

1. **Project Manager**
   Oversees overall project planning, manages sprint schedules, tracks risks, and ensures milestones are met on time.

2. **Lead Developer**
   Leads the development of the game logic and backend integration using Roblox Studio and Spring Boot.

3. **UI/UX Designer**
   Designs user interfaces and game environments to create an engaging and age-appropriate experience for elementary students.

4. **Full Stack Developer**
   Develops the teacher web dashboard using React and manages communication between the game client and backend services.

5. **QA Tester**
   Conducts functional and usability testing to ensure the system operates smoothly and meets quality standards.

6. **Documentation Lead**
   Prepares and maintains project documentation, including technical reports, user manuals, and version control.

Team members may assume multiple roles as necessary throughout the project lifecycle. Weekly meetings are held to review progress, discuss challenges, and coordinate tasks.

4.3. Roles and responsibilities

This section outlines the major work activities and supporting processes involved in the **CyberKids** project and assigns them to the respective organizational units or team members responsible for their execution. Each role plays a vital part in the success of the project, ensuring that all technical, educational, and managerial tasks are completed on schedule and to the required standards.

The following matrix depicts the assignment of roles and responsibilities across the different work activities and supporting processes:

TEAM MEMBER	**ROLE & RESPONSIBILITIES**	
	SEM 1	**SEM 2**
Jesson Chyd Cultura	**Project Lead** - serves as the single point of accountability for CyberKids, overseeing scope, schedule, and stakeholder communication. **Backend Developer** - Responsible for designing, developing, and maintaining the server-side components of the system. Manages database integration, API development, and real-time communication infrastructure. **Frontend Developer** - Builds and optimizes the teacher web dashboard using React.js. **Game Developer** - implements and optimizes in-game mechanics, UI, and backend integrations within Roblox.	Continues these roles into Semester 2.

TEAM MEMBER	ROLE & RESPONSIBILITIES	
	SEM 1	**SEM 2**
	UI/UX Designer - Designs visual and interactive elements of the CyberKids game and teacher dashboard.	
Emmanuel Dedumo	**Game Designer** - Responsible for designing and creating the game's graphical user interface (GUI), animations, cutscenes, and visual elements to ensure an engaging and polished player experience within Roblox.	Continues these roles into Semester 2.
Ashley Josh Vequiso	**Backend Developer** - Handles backend development tasks including Microsoft Teams authentication integration, dynamic CRUD operations in Spring Boot for real-time data polling in the Level 1 Roblox game, and leaderboard management. **Frontend Developer** - Builds and optimizes the teacher web dashboard using React.js. **Game Developer** - implements and optimizes in-game mechanics, UI, and backend integrations within Roblox.	Continues these roles into Semester 2.

TEAM MEMBER	ROLE & RESPONSIBILITIES	
	SEM 1	**SEM 2**
Jhudiel Artezuela	**Game Designer** - Contributes by creating basic cutscenes. His work is minimal and focuses on very short, basic visual elements.	Inactive (no contribution)
John Kenneth Baguio	**Game Designer** - Creates advanced cutscenes with animations, camera effects, and visual enhancements. Also contributes to the game's story development, ensuring it complements the gameplay and user experience.	Continues these roles into Semester 2.

[Table 4.3.1] Role and Responsibilities

5. Managerial process plans

This section details the managerial approaches and processes applied to effectively plan, coordinate, and execute the CyberKids project. It covers how the team managed time and resources, organized roles, conducted training where necessary, and maintained work schedules. These structured processes ensured smooth progress, timely milestone delivery, and quick resolution of any issues during development.

5.1 Start-up plan

5.1.1. Estimation plan

To ensure realistic time and resource management, the CyberKids team used a collaborative and data-driven estimation process. This process factored in prior experience from Capstone 1, the complexity of integrating Roblox game mechanics with backend systems, and the dual development environment (game and web dashboard).

Estimation Process

- **Task Identification:**
  The team broke down the project into key tasks including Roblox game development, backend API creation, React-based teacher dashboard, UI/UX design, content creation (animations, cutscenes), and testing phases.

- **Initial Time Estimates:**
  Each member provided time estimates for their assigned components, considering factors such as game mechanics complexity, backend integration for dynamic content, and UI responsiveness for young users.

- **Group Discussion and Final Estimates:**
  Differences in initial estimates were resolved through team discussions. For example, game development time was adjusted after considering the design scope and integration needs with the Spring Boot backend and Railway hosting.

Shared Tracking Tool

A shared online spreadsheet was used to log task progress, update timelines, and track completion status. This centralized tool enabled transparency and allowed quick adjustments to the schedule as development progressed.

Time Estimate Reviews

Weekly team meetings were conducted to review the schedule, evaluate progress, and recalibrate estimates. Adjustments were made promptly when unforeseen challenges arose, such as UI refinements suggested by teachers during pilot testing or debugging backend API calls.

Time Estimates and Milestones

- **Weekly Time Allocation:**
  Team members committed between 12 to 48 hours weekly, balancing project demands with academic responsibilities.

- **Project Milestones:**
  - **Alpha Version Completion:** April – Core game mechanics and basic dashboard functionalities completed.
  - **Beta Version Completion:** June – Full integration with backend, UI/UX enhancements, and initial testing phases.

- **Final Version Completion:** October – Polished product including finalized animations, cutscenes, and bug fixes, ready for deployment and evaluation.

Testing and Bug Fixing

Dedicated time was reserved for iterative testing and bug fixing, including feedback cycles from university teachers during pilot testing. This phase ensured the CyberKids game was educational, engaging, and stable before release.

5.1.2 Staffing plan

The CyberKids project team consists of five student developers from Cebu Institute of Technology University (CIT-U). Each member contributed across all phases of the project, including planning, design, development, testing, and documentation, while focusing on their respective areas of expertise.

Team Structure and Staffing Timeline

- **Part-time Commitment During Semesters:**
  Team members balanced their academic responsibilities with project work, dedicating part-time hours to CyberKids development throughout the semester.

- **Full-time Effort During Breaks:**
  During semester breaks and critical deadlines, the team increased their commitment to full-time hours to ensure timely milestone completion.

- **Weekly Coordination Meetings:**
  Regular weekly meetings were held to review progress, plan upcoming tasks, and reassign responsibilities as needed to keep the project on track.

Skill Requirements

Team members were chosen based on their existing skills or their eagerness to develop proficiency in the following areas essential to the project:

- Game development using Roblox Studio and Lua scripting
- Backend programming with Spring Boot and API integration
- Frontend web development with React.js for the teacher dashboard
- UI/UX design principles focused on educational and age-appropriate interfaces
- Testing, debugging, and quality assurance
- Documentation and formal report writing

Staffing by Project Phase

Project Phase	Required Staff	Skills Needed	Duration
Planning & Design	All 5 members	Research, wireframing, tool design	2 weeks
Game Development	3-4 members	Roblox Studio, Lua scripting	4-6 weeks (part/full-time)
Backend Development	1-2 members	Spring Boot, API development, database management	4 weeks
Frontend Development	1-2 members	React.js, UI components	2-3 weeks
Testing & Debugging	4-5 members	Manual testing, bug fixes	Ongoing throughout project
Documentation	3-5 members	Report writing, formatting, user guides	Throughout project

Sources of Staff

All team members were recruited internally from CIT-U's academic program. No external contractors were involved. A faculty adviser provided mentorship, technical guidance, and regular feedback.

Flexibility and Reassignment

The team maintained flexibility to accommodate shifting workloads and availability. Tasks were reassigned dynamically during weekly meetings to address delays or reallocate resources efficiently.

Staff Retention

The core team remained consistent from project start to finish, ensuring continuity, smooth collaboration, and knowledge retention with zero turnover.

5.1.3. Resource acquisition plan

This section outlines the hardware, software, and network resources required for the successful development of the **CyberKids** project. It describes how these resources were acquired, allocated, and managed throughout the project lifecycle to ensure effective collaboration, development, and deployment.

A. Hardware Resources

Each team member utilized a personal laptop as their primary development environment. To support the requirements of Roblox Studio, Spring Boot backend development, and React-based frontend work, minimum technical specifications were established to ensure compatibility and performance.

Minimum System Requirements:

- **Processor:** Intel Core i3 (8th Generation or higher)
- **RAM:** Minimum 4 GB (8 GB or more recommended)
- **Storage:** At least 2 GB of available disk space
- **Operating System:** Windows 10 or newer
- **Graphics:** Integrated GPU (Roblox Studio does not require dedicated graphics)
- **Network:** Stable internet access for online collaboration, cloud services, and development tools

> **Compliance:** All team members validated their hardware against the baseline requirements prior to the start of development to avoid performance bottlenecks during game simulation, compilation, or testing.

B. Software Tools and Platforms

Tool/Software	Purpose	Source	Access Type
Roblox Studio	Game development platform	Roblox Corporation	Free
React.js	Frontend development for teacher dashboard	Microsoft	Free
Spring Boot	Java framework for backend APIs	Open-source	Free
VS Code	An integrated development environment	Microsoft	Free
Postman	A tool for testing and managing RESTful API requests	Postman	Free
Docker Desktop	A platform to containerize and run the application components	Docker	Free
MySQL Workbench	A relational database management system used to store and manage all user data	Oracle Corporation	Free
Git and GitHub	Version control and collaboration	Git/GitHub.com	Free
Railway Database	Online database for storing scores, progress	Railway	Paid
Render	A cloud service used to deploy and host the backend API, providing reliable server uptime and scalability.	Render	Free
Vercel	A cloud platform for deploying and hosting the frontend React.js application with fast global delivery.	Vercel	Free

C. Paid Cloud Resource

Cloud Database Hosting – Azure Database for MySQL

While most tools used were free, the project required a reliable and scalable cloud database to support dynamic gameplay and web dashboard functionalities. The team initially utilized **Microsoft Azure** for this purpose during the early development phase but later transitioned to **Railway.app** for cost-efficiency.

Platform	Azure Database for MySQL (Capstone 1)
Use Case	Real-time player data, leaderboard, teacher dashboard integration
Plan	Basic Tier
Duration	Capstone 1
Cost	₱2,000 total (₱320/member approx.)
Management	Access and configuration managed by backend developer and team lead

Platform	Railway.app (Capstone 2)
Use Case	Continued backend support and database polling
Plan	Starter Plan
Duration	Capstone 2
Cost	₱500 total
Management	Maintained by backend developer

D. Network and Infrastructure Resources

- **Internet Access:**
  Stable internet connectivity was critical for accessing cloud tools, performing real-time collaboration, and conducting online research and testing. Each team member ensured they had consistent access to broadband internet.

- **Remote Collaboration:**
  The majority of the development work was conducted remotely, using Google Meet and group chats for communication and GitHub for code sharing and version control.

- **On-Site Facilities:**
  The university's computer laboratories were made available when needed for group sessions, collaborative testing, and evaluations, though these were used sparingly due to remote-first project dynamics.

5.1.4. Project staff training plan

This section outlines the training activities conducted to prepare the project team for the development of the **CyberKids** system. While some team members brought prior experience, others required focused learning to gain proficiency in specific tools and frameworks. The training plan ensured skill alignment, improved collaboration, and supported the delivery of a high-quality educational game and teacher dashboard.

Training Objectives

The primary objectives of the staff training plan were:

- To ensure each team member had sufficient knowledge of the tools and technologies used in the project.
- To enhance productivity by promoting shared technical knowledge and problem-solving techniques.
- To prepare members to independently contribute to specialized project components such as the Roblox game, backend system, web dashboard, and documentation.

Training Timeline and Structure

- **Initial Training Period:**
  Conducted during the **planning phase** (first 1–2 weeks), focusing on onboarding and familiarization with Roblox Studio, Git, and project tools.

- **Ongoing Training:**
  Continued throughout development via:
  - Weekly technical discussions during team meetings
  - Shared resources (links, documents, tutorials)
  - Peer support and informal mentorship

- **Mentorship:**
  Experienced team members (especially those with prior exposure to game development or backend technologies) provided peer assistance and resolved blockers in real-time.

Training Outcomes

- Team members developed the necessary competencies to confidently build and manage their assigned components.
- Shared understanding of tools and workflows improved collaboration and code consistency.
- Time spent on training significantly reduced implementation delays and rework during core development phases.

5.2 Work plan

5.2.1. Work activities

The project was broken into clear phases. Each phase included tasks that built on each other.

A	B	C	D	E	F
Phase	**Task**	**Start Date**	**End Date**	**Duration**	**Milestone**
Planning	Task breakdown, team roles, scheduling	2/1/2025	2/7/2025	1 week	Team organization completed
Design	Game design, wireframes, dashboard mc	2/8/2025	2/18/2025	1.5 weeks	Game layout ready
Game Dev - Module 1: Privacy	Module 1 development	3/25/2025	4/7/2025	2 weeks	Module 1 complete
Backend Dev	Login, scoring, API setup	3/18/2025	3/31/2025	2 weeks	Backend integrated
UI/Dashboard	Build teacher dashboard (React.js)	4/1/2025	4/10/2025	1.5 weeks	Dashboard functional
Testing	Unit testing, integration testing	4/11/2025	4/24/2025	2 weeks	Testing completed
Bug Fixing	Debug modules, improve performance	4/25/2025	5/1/2025	1 week	Final polish
Documentation	Finalize SPMP, User Manual, Reports	4/22/2025	5/3/2025	1 week	Final documents submitted

5.2.2. Schedule allocation

The CyberKids project was developed over two academic semesters, from March to early May 2025, with the team balancing school requirements and project responsibilities. The development schedule was aligned with the institutional academic calendar, allowing for efficient allocation of time and resources without disrupting coursework.

All project work was conducted by a five-member undergraduate team from Cebu Institute of Technology – University (CIT-U), with members taking on multiple roles to meet both technical and managerial demands. Collaboration was managed virtually, using shared development environments and version control platforms.

Hardware and Working Environment

Each team member utilized a personal laptop that met the minimum system requirements for development tasks including game design, frontend/backend programming, and testing. The team operated remotely, with occasional in-person coordination meetings, although the school's computer laboratory remained available for group sessions if necessary.

Software Tools and Services

Development tools were selected based on cost-effectiveness, relevance, and accessibility:

- **Game development**: Roblox Studio and Lua
- **Backend development**: Spring Boot (Java)
- **Frontend development**: React.js
- **Database and hosting**: Railway (₱500 PHP paid service)
- **Version control**: Git and GitHub
- **Project documentation**: Google Docs and Drive

Only one paid service—**Railway**—was used for hosting the backend and database services. The subscription cost was covered by the team collectively and managed by the backend developer and team lead.

Schedule Milestones

The project timeline was divided into major phases with corresponding deliverables:

Milestone	Target Completion
Project Planning and Training	March 10, 2025
SRS Finalization	March 24, 2025
SDD Completion	March 30, 2025
Level 1 Game Development (Roblox)	March 25 – April 15
Backend & Database Integration	April 1 – April 20
Dashboard UI (Frontend) Development	April 10 – April 25
Testing & Bug Fixing	April 25 – May 2
Final Documentation & Polishing	May 3, 2025

Time Management and Workload

Each member committed between **12 to 48 hours per week**, depending on academic workload. Weekly meetings ensured consistent tracking of deliverables, reassignment of tasks when necessary, and timely resolution of development blockers.

5.2.3. Resource allocation

The resource allocation for the *CyberKids* project is aligned with the scope of work outlined in the Work Breakdown Structure (WBS). This section identifies the human and technical resources assigned to each major task or module of the system. All team members are students enrolled in the Capstone Project class and are expected to devote approximately **10–12 hours per week** on project-related activities during the semester.

Personnel Allocation

Module	**Work Activity**	**Personnel Assigned**	**Skill Level**	**Hours/Week**
Module 1: Gamified Cybersecurity Game Learning Module	• Roblox Studio • Backend (Spring Boot)	Cultura Dedumo Vequiso	• Intermediate (Lua Scripting) • Advance (Java REST api)	11
Module 2: Student Real-Name Integration	• Roblox Studio • Backend (Spring Boot)	Cultura	• Intermediate (Lua Scripting) • Advance (Java REST api)	3
Module 3: Teachers Dashboard	• Frontend (React) • Backend (Spring Boot)	Cultura Vequiso	• Advance (React js + Springboot)	8
Module 4: Notification System	• Frontend (React) • Backend (Spring Boot)	Cultura	• Advance (React js + Springboot)	5

Technical Resources Allocation

Resource Type	**Details**	**Allocated For**
Development Environment	Roblox Studio, Visual Studio Code	Game Design
Backend Framework	Spring Boot (Java), MySQL	APIs, Leaderboards, Scoring Systems, Timer System
UI Libraries	React.js, ShadCN UI Component, Tailwind	Teacher Dashboard
Data Storage	Azure Database for MySQL (local and hosted)	All Modules
Testing Tools	Postman	API & Logic Testing
Document Tools	Google Docs, Visual Paradigm	Documentation & Diagrams
Simulation/Testing Devices	Student-owned laptops (min spec: 8GB RAM, i5)	Development and local deployment

Summary of Total Weekly Resource Hours

Role	**Team Member(s)**	**Estimated Weekly Hours**
Frontend Developers	Cultura Dedumo Vequiso	48 hours (4 × 12 hrs)
Backend Developers	Cultura Vequiso	24 hours (2 × 12 hrs)
Documentation & Testing	Baguio Cultura Dedumo Vequiso	Shared across total weekly hours
Project Tools & Platforms	All Members	Included in weekly hours

All resources will be adjusted based on weekly sprint feedback and the project timeline set during the planning phase. Any changes to personnel availability or tool access will be recorded and approved during the weekly team meetings.

5.2.4. Budget allocation

The **CyberKids** project was developed by a student team with limited financial resources. As a capstone initiative, most tools and platforms used were free, open-source, or already owned by team members. However, two key areas incurred actual monetary costs during development:

1. **Backend and database hosting using Railway**, to enable dynamic data processing.
2. **MS Teams Integration Setup**, which required temporary access to premium services for development and testing (though later deprecated in production).

Summary of Project Expenses

Category	Item / Resource	Estimated Cost (PHP)	Remarks
Backend Hosting	Railway (Hobby Plan)	₱500	Used for hosting Spring Boot backend and MySQL database during testing and deployment
Third-Party Integration	Microsoft Teams API Access	₱2000	Temporary subscription during development of Teams integration (no longer used in final system)
Development Equipment	Personal Laptops	₱0	All team members used their own development machines
Game Development Engine	Roblox Studio	₱0	Free-to-use game engine for development and publishing
Development Tools	IntelliJ IDEA (Community), React.js, GitHub	₱0	All frameworks and tools used were free and open-source
Documentation & Storage	Google Drive, Docs, Sheets	₱0	Used for collaborative writing and file sharing

Cost Management Notes

- All paid expenses were **self-funded** by team members.
- Costs were minimized by leveraging **free-tier services** and **open-source tools** wherever possible.
- The MS Teams integration, although deprecated, was still considered part of the official development timeline and budget.

Budget Considerations

This lightweight financial plan reflects the resource-conscious nature of student-led software development, while still maintaining access to necessary tools for professional-grade output.

5.3. Control plan

This section outlines the metrics, reporting mechanisms, and control procedures used to measure and ensure that the **CyberKids** project adheres to defined requirements, schedules, resource plans, and quality standards. These controls are aligned with the policies and procedures followed in our Capstone Project course and the standards set by our development methodology.

5.3.1. Requirements control plan

Product requirements for *CyberKids* are maintained in the **Software Requirements Specification (SRS)** document and tracked using a version-controlled system (Google Docs with history tracking and manual versioning logs). Any proposed changes to the requirements must follow a **formal change control procedure**:

- **Change Request Form (CRF):** Submitted by any team member or stakeholder.
- **Change Evaluation:** The development team evaluates the request's impact on scope, timeline, quality, and resources.

  Approval Process: All requirement changes must be reviewed and approved during weekly team meetings by a designated **Requirements Review Board**, composed of the project manager and at least two senior developers.

Control mechanisms include:

- **Requirements Traceability Matrix (RTM):** Used to trace each requirement to its design, implementation, and test case.
- **Impact Analysis:** Each change request will undergo an analysis for technical feasibility and potential ripple effects.
- **Prototype Reviews:** UI and game mechanics prototypes are validated by stakeholders and iterated based on feedback.
- **Version Control:** Requirements documents and updates are maintained in a shared project drive with proper version naming

5.3.2. Schedule control plan

The project schedule is maintained in a **shared Gantt chart** (Google Sheets) and updated weekly by the project manager. Each major milestone (e.g., Module Completion, User Testing, Final Presentation) is broken down into individual tasks assigned to team members.

Mechanisms for schedule control include:

- **Weekend Stand-Up Meetings:** Held every Saturday to report task completion, blockers, and planned work.
- **Progress Tracking:** Each task status is tracked (Not Started, In Progress, Completed) and visually updated on the Gantt chart.
  Milestone Reviews: Conducted after each module is completed to ensure all features are working as planned and integrated properly.
- **Corrective Actions:** If delays are detected, re-planning occurs through a **Schedule Control Meeting** led by the project manager. Tasks may be redistributed based on availability and urgency.

Tools used:

- **Google Sheets:** For Gantt chart and task status updates.
- **Clickup:** For task breakdown, checklists, and ownership tracking.
- **Calendar Reminders:** For coordinating meetings and development sprints.

All schedule progress is assessed objectively through delivered code, successful module tests, and peer reviews to ensure quality alongside timeliness.

5.3.3. Budget control plan

The team will keep track of the monthly Azure Database subscription cost and make sure it is paid on time. If the cost increases or becomes hard to manage, the team will discuss possible solutions such as downgrading the plan or switching to a free option. The team will make sure the budget stays within what was planned for the whole project.

5.3.4. Quality control plan

To ensure the quality of the *CyberKids* project, the team will conduct regular reviews and testing throughout the development process. Weekly meetings will be held to check progress, review work, and identify any issues early.

Each module will be tested through unit and integration testing to make sure it works properly. Peer reviews will be done to check the code and documents before they are finalized. Mentors will also help monitor the quality of the project by giving feedback during scheduled consultations.

A simple checklist will be used to guide the team in maintaining quality in both the game and documentation.

5.3.5. Reporting plan

The **Reporting Plan** defines the communication structure for sharing updates on the CyberKids project. It ensures that both internal and external stakeholders are kept informed about project progress, challenges, and achievements. The reporting structure supports effective decision-making and timely interventions when needed.

General Reporting

Effective communication is vital for the success of the project. The CyberKids team will leverage modern digital tools to ensure that all stakeholders are kept updated on the project's status and progress.

- **Communication Tools:**
  - Google Docs: For creating and sharing project documents, ensuring all team members have access to the latest updates and deliverables.
  - Clickup: For collaborative tracking of tasks, deadlines, and progress, making it easy to monitor real-time updates.
  - Group Chats (e.g., Messenger and MS Teams): For quick communication, discussions, and informal updates among team members.

- **Weekly Reports:**

  Each week, the team will compile a report that covers the following:
  - Status of Requirements: A summary of how the project requirements are being met, highlighting any changes or challenges.
  - Deadlines: Review of upcoming deadlines, ensuring that milestones are on track or identifying potential delays.
  - Issues: Any challenges or obstacles encountered during the week, with potential solutions or requests for assistance.
  - Completed Tasks: A summary of tasks completed in the past week, showing progress toward the final goals.

Internal Reporting

The internal reporting process ensures that all team members remain aligned with project objectives and timelines. Weekly updates will be provided in team meetings, ensuring that any issues or blockers are promptly addressed.

- **Weekly Team Updates:**
  Each team member will provide a report during the weekly meeting, which will include:
  - Task Status: Updates on the progress of their assigned tasks.
  - Blockers: Any challenges or obstacles that are preventing progress.
  - Planned Work: A preview of what the team member intends to work on in the upcoming week.

External Reporting

- At the end of each project phase, the team will prepare a progress report for mentors and project advisers.
- A final presentation and report will be submitted to the Capstone instructor as part of the completion requirement.
- All key documents (SRS, SDD, SPMP) will be shared through the school's project submission platform or via Google Drive.

5.3.6. Metrics collection plan

The **Metrics Collection Plan** establishes how the CyberKids project team will track progress, monitor productivity, and identify areas for improvement. This plan is designed to ensure that the team stays on target with respect to both time and quality while providing insights into potential areas for adjustment.

Weekly Progress Tracking

To monitor the team's progress and performance, each team member will update a shared project tracker every Friday. The tracker will be used to record key metrics related to task completion, time management, and any challenges faced during the week.

The tracker will include the following columns:

- Tasks Assigned: A list of the tasks assigned to the team member during the week.
- Tasks Completed or Pending: A status update on the tasks – whether they have been completed or are still in progress.
- Problems or Blockers: A summary of any challenges or obstacles encountered that impacted task completion.
- Hours Planned vs. Hours Actually Spent: A comparison of the initially estimated hours for each task versus the actual time spent on it.
- Next Steps: Notes on what the team member plans to work on in the following week.

Review Process

The Team Leader will review the data entered into the tracker every Saturday before the weekend meeting. This review process will serve to:

- Summarize the overall progress made by the team in the past week.
- Identify any issues or trends that may need to be addressed.
- Ensure that the project remains on track with respect to deadlines and quality expectations.

Metrics Utilization

The collected metrics will serve several key purposes:

- Team Productivity: By comparing planned vs. actual hours, the team leader can evaluate how efficiently the team is working and identify potential bottlenecks or overestimations.
- Tracking Delays: Metrics on task completion and blockers will help identify delays early on, allowing the team to take corrective actions as needed.
- Identifying Areas for Attention: Any recurring challenges or tasks that take longer than expected can be flagged for further discussion, training, or reallocation of resources.

5.3.7 Risk management plan

The **Risk Management Plan** outlines how potential risks to the CyberKids project will be identified, monitored, and mitigated. This plan will help ensure that the team stays on track, can effectively manage obstacles, and minimize delays to the project timeline.

Development Risks

To prevent development delays and maintain progress, the team will meet every week to track task completion and ensure that all tasks are on schedule. Key measures include:

- Set Deadlines: Every task will have clearly defined deadlines to avoid delays and keep the development process organized.
- Communication: Team members are expected to communicate any issues or delays to the Team Leader immediately, allowing for quick resolution or adjustments to the project plan.

Technical Risks

The CyberKids project is subject to various technical challenges that could hinder progress. Key technical risks include:

- Bugs and Technical Issues: Bugs in the Roblox Studio or Spring Boot backend could cause delays. To manage this risk, the team will conduct regular testing during development to identify and address bugs early.
- Internet and Hardware Issues: Interruptions due to internet connectivity or hardware failures can delay progress. Team members should have backup devices or alternative work solutions in case of hardware failure.
- Platform or API Updates: If Roblox or any integrated API tools release updates that cause errors or incompatibilities, the team will promptly research solutions and update the code as necessary to maintain functionality.

Team Management Risks

Team dynamics can present risks if members are not contributing or if there are performance issues. To mitigate this, the team will:

- Monitor Member Progress: If a team member is falling behind, the issue will be discussed in a meeting, and corrective actions will be agreed upon to ensure progress is maintained.
- Support and Collaboration: The team will provide assistance to any member who needs help in completing their tasks. Peer support will help ensure that all team members stay on track and meet deadlines.
- Role Reassignments: If necessary, temporary role reassignments will be made to redistribute tasks and keep the project moving forward without significant delays.

Project Completion Risks

To minimize the risk of project failure and ensure successful completion, the following measures will be implemented:

- **Backup Strategy:** The team will regularly **back up all project files** in shared drives (e.g., Google Drive) to prevent data loss in case of technical failures.
- **Mentor Feedback:** The team will receive **regular reviews and feedback** from mentors to ensure quality and project direction remain aligned with project goals.
- **Contingency Time: Contingency time** will be built into the project schedule to account for any unexpected delays, providing a buffer to address unforeseen issues.

Risk Monitoring and Mitigation

Throughout the project, the team will consistently monitor these risks and adapt the plan as necessary to address new challenges. The regular risk assessment process ensures that the team can make informed decisions, take corrective actions, and keep the project on track for successful completion.

5.3.8 Project closeout plan

The Project Closeout Plan ensures that all activities related to the CyberKids project are formally concluded in an organized and documented manner. This phase includes the completion and submission of final deliverables, presentation to stakeholders, documentation archiving, and team retrospection. It guarantees that the project is not only finalized but also properly handed off for potential future maintenance or academic reference.

1. Final Deliverables Submission

Upon completion, the team will submit all required deliverables through the designated university submission channels, as well as the project's shared repositories:

- **GitHub Repository** – Contains the complete source code, version history, and deployment instructions.
- **Google Drive Folder** – Centralized archive for formal documentation and presentation materials.

Submitted Artifacts:

- Final Source Code (Roblox Game, Backend APIs, React Dashboard)
- Software Project Management Plan (SPMP)
- Software Requirements Specification (SRS)
- Software Design Document (SDD)
- Software Test Document (STD)
- User & Maintenance Guide (Basic instructions for future users or developers)
- Capstone Presentation Materials (slide deck, demo video, technical demo files)

2. Final Presentation

The final version of the CyberKids application will be demonstrated during the official Capstone Final Presentation Day. The team will present:

- Project overview and objectives
- Demonstration of key features (gameplay, cutscenes, dashboard, backend integration)
- Development process, challenges, and solutions
- Outcomes, lessons learned, and potential for future development

The presentation will serve as a formal evaluation opportunity for advisors, faculty panelists, and stakeholders (e.g., teachers or IT coordinators).

3. Documentation & Archiving

All project files, resources, and codebases will be securely archived for future access and reference. This includes:

- Final PDFs of all documents
- Editable formats (e.g., .docx, .pptx) for possible revisions or reuse
- Code repositories with setup and deployment notes
- Screenshots, gameplay videos, and UI mockups

Storage Locations:

- Official GitHub repository (public or private, as required)
- Google Drive (institutional or personal backup)
- Offline backup (USB/hard drive maintained by project lead)

4. Project Debrief and Reflection

To close the project from an internal standpoint, the team will conduct a final **project retrospective meeting**. This session will include:

- **Team Reflection:**
    - Successes and major achievements
    - Challenges faced and how they were overcome
    - Lessons learned across technical, managerial, and collaborative aspects
- **Recommendations for Future Teams:**
    - Advice on tools, platforms, and workflows
    - Pitfalls to avoid (e.g., scope creep, misaligned roles)
    - Suggested improvements in documentation or testing

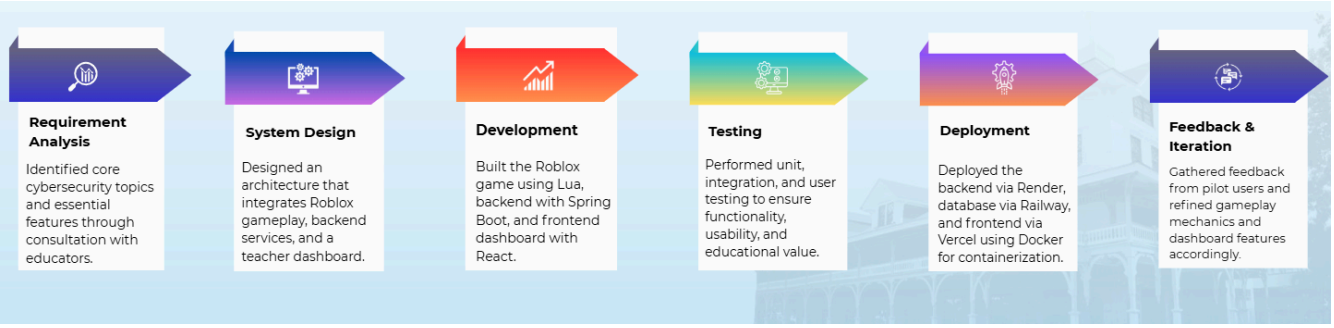
Meeting outcomes will be summarized and added to the final SPMP as part of the project's continuous improvement record.

6. Technical process plans

This section outlines the technical strategies, tools, and methodologies applied in the development of the CyberKids project. It defines the chosen development model, work product generation methods, technical infrastructure, and the product acceptance strategy. Emphasis is placed on the iterative and agile-based processes that supported frequent feedback, rapid prototyping, and incremental delivery.

6.1 Process Model

Agile Iterative Development Approach



The **CyberKids project** followed an **agile iterative approach**, structured into **four main sprint cycles**. Each sprint focused on delivering a key functional component (game level, backend integration, dashboard features) and included clear checkpoints for planning, execution, testing, and reflection.

Due to the academic context and part-time nature of student work, each sprint lasted **2–3 weeks**, depending on the complexity of the features involved.

Below is the detailed breakdown of how each phase of the agile cycle was adapted and applied to the project:

1. Sprint Planning (Every Start of Iteration)

- **Goal**: Define sprint objectives, deliverables, and task breakdown.
- **Activities**:
    - Selected prioritized features or modules based on project roadmap.
    - Assigned tasks based on expertise and availability.
    - Defined user stories and acceptance criteria for each feature.
    - Estimated effort in hours using informal consensus.
- **Tools Used**: ClickUp (task board), Google Sheets (Sprint Backlog), Messenger group chat.

2. Requirement Refinement & Design

- **Goal**: Detail out the user stories, technical needs, and visual or game designs before coding.
- **Activities**:
    - Created mockups for game levels or dashboard screens.
    - Refined gameplay mechanics and backend logic.
    - Confirmed technical feasibility.
    - Collaborated with Teacher Rosario (when applicable) for validation of educational goals.
- **Artifacts Produced**:
    - Paper or digital mockups (Figma, draw.io, etc.)
    - Short design outlines or diagrams (e.g., flowcharts, sequence diagrams).
    - Updated SRS/GDD with refined details.

3. Incremental Development

- **Goal**: Code the features and modules defined in the current sprint.
- **Activities**:
    - Game development in Roblox Studio using Lua.
    - Backend API development in Spring Boot (hosted on Railway).
    - Frontend development in React.js (hosted on Vercel).
    - Cutscene animations, GUI creation, and story progression design.
- **Pair Programming** (in some cases): More experienced members helped others, especially for integrations.
- **Version Control**: All work was pushed and reviewed on GitHub using branches per feature.

4. Internal Testing & Peer Review

- **Goal**: Ensure new features are working correctly and integrated smoothly.
- **Activities**:
    - Unit testing of game scripts and backend functions.
    - Integration testing (e.g., checking score saving, leaderboard sync).
    - Peer reviews of code, logic, and UI/UX.
    - Logged bugs in shared tracker (Google Sheets or ClickUp).
- **Testing Types**:
    - Manual gameplay testing (peer-run).
    - Backend endpoint testing via Postman.
    - Data validation through live dashboards.

5. Sprint Review / Feedback Session

- **Goal**: Showcase what was built, gather feedback, and validate progress.
- **Activities**:
    - Demonstrated the working features to the team and faculty mentor (if available).
    - Discussed what worked well and what needed improvement.
    - Collected feedback for gameplay feel, clarity, and dashboard data flow.
    - Minor UI/UX tweaks were often made during this phase.

6. Sprint Retrospective

- **Goal**: Reflect on the sprint and plan how to improve.
- **Activities**:
    - Identified blockers and delays (e.g., absences, integration challenges).
    - Adjusted work distribution for the next sprint.
    - Documented key learnings and action points.
- **Real Examples**:
    - Task reassignment when one designer became inactive.
    - Schedule adjustments around midterms and final exams.
    - Agreement to simplify animations to meet deadlines.

6.2 Methods, tools, and techniques

This section describes the methodologies, programming languages, frameworks, tools, and standards used in the development, testing, documentation, integration, deployment, and maintenance of the CyberKids project. The project comprises two core components: the CyberKids Roblox Game and the Teacher Dashboard Web Application, along with a centralized backend and database system.

6.2.1 Game Mission Module – Tools / Technologies

A. Methods and Techniques

Category	Tools / Technologies
Requirements Elicitation	- Discussions with educators to determine key educational content to gamify. - Review of classroom behavior to align with learning goals.
Design	- Use of Roblox Studio GUI tools to layout scenes, characters, and logic. - Gameflow diagram to visualize level progression. - Wireframes for in-game interfaces.
Implementation	- Lua scripting for event handling, scoring, collision, and level logic. - Modular scripting for reusable game mechanics (e.g., obstacle logic, timers).
Testing	- Manual testing within Roblox Studio for each level. - Group testing with sample students for usability and engagement feedback.

Integration	- Backend score submission via REST API calls to Spring Boot service. - Real-time syncing of student achievements.
Deployment	- Game published to Roblox platform (private access initially for internal testing).
Documentation	- In-game help screens. - Markdown-based technical documentation for scripting logic. - User feedback logs from testing sessions.

B. Tools and Technologies

Category	**Tools / Technologies**
Game Engine	Roblox Studio – Scripting, level design, animations
Scripting Language	Lua – Game logic and mechanics
Design and Layout	Roblox built-in editor + external asset imports (e.g., 3D models, audio)
Source Control	Git + GitHub – Version control of scripts and shared media
Testing	Roblox playtest mode, student trial sessions
API Integration	REST API calls to Teacher Dashboard backend
Documentation	Markdown, internal wiki

C. Standards & Procedures

- Game structure follows modular design per level (Level 1: Logging in, Level 2: Cyberbullying, etc.).
- All educational content is validated by teacher input.
- Lua code adheres to Roblox Lua style guidelines.
- All levels include an exit point that saves player score to the backend before logout.

6.3 Infrastructure Plan

The development and deployment of the CyberKids project, including the Game Mission Module and the Teacher Dashboard Module, will be carried out using cloud-based development tools, distributed team workflows, and remote deployment environments. The infrastructure supports all phases of the software lifecycle—from requirements gathering to delivery.

6.3.1 Game Mission – Infrastructure

For the Game Mission, the development environment will be established using individual workstations equipped with mid-to-high performance specifications (minimum Intel Core i5, 8GB RAM) running on Windows 10 or higher. The primary development tool, Roblox Studio, will be installed locally but integrated with the Roblox cloud for asset storage and version control. A stable internet connection is essential to access the Roblox asset repository and test multiplayer features during development. Developers will work remotely and utilize the university's online communication platforms and cloud storage services (such as Google Drive) for collaboration and documentation. The infrastructure also includes a virtual sandbox testing environment through Roblox's playtest feature, allowing developers to simulate gameplay behavior before live deployment. All development-related tasks will be conducted in individual home offices, supported by university-provided licenses and accounts when necessary.

6.3.2 Teacher Dashboard – Infrastructure

For the Teacher Dashboard, the infrastructure comprises both local and cloud-based environments. Developers will use personal computers with modern operating systems (Windows/macOS) connected to a reliable internet network to access version control (GitHub), cloud-hosted backend services (Render), and a frontend deployment platform (Vercel). The backend will run on Spring Boot, connecting to a cloud-based Azure MySQL database to manage teacher and student data. The frontend and backend will be integrated and tested using remote CI/CD pipelines, ensuring that updates are automatically deployed to the test and live environments. University cloud storage and collaboration tools will be used to manage requirements documents, user feedback logs, and bug tracking. The team will maintain infrastructure access policies by securing all platforms with account-based permissions and using encrypted environment variables to store credentials and configuration secrets.

6.3 Product Acceptance Plan

This section outlines the formal process for accepting the deliverables of the CyberKids project. The acceptance plan includes objective criteria and procedures to determine the quality and completeness of the developed components. It ensures that both the development and acquiring organizations have a shared understanding of the standards and responsibilities for product validation. Acceptance will be determined based on formal inspections, testing, and demonstrations agreed upon by both parties.

6.3.1 Game Missions

The **Game Missions** is the core interactive component of the CyberKids platform, designed using the Roblox game engine to teach cybersecurity concepts to Grade 5 and Grade 6 students. This module includes levels such as *Data Leak Investigation*, *Password Fortress Defense*, and *Cyber Escape Room*.

To ensure product acceptance, the module will undergo formal evaluations at the end of each development iteration. The acceptance criteria include functional correctness of the game mechanics, accuracy in progress tracking, proper saving of game states, and successful implementation of unlocking logic. Teacher Rosario, representing the acquiring organization, will conduct structured playthroughs with selected students to verify usability, educational impact, and engagement quality.

All core features—such as drag-and-drop categorization, phishing detection, and level-based challenges—must function seamlessly and reflect real-time updates on student profiles. Each iteration will be reviewed through scheduled demonstration sessions, followed by documented test results. Upon meeting all technical and functional requirements, an acceptance certificate will be signed, and feedback will be collected for future refinement. Any requested adjustments will be addressed before final delivery.

6.3.2 Teacher Dashboard Module

The **Teacher Dashboard** serves as the analytical and monitoring tool for instructors. Developed as a web application using React (frontend deployed via Vercel) and Spring Boot (backend deployed via Render) with an Azure MySQL database, this module provides real-time insights into student performance, leaderboard rankings, and time-on-task metrics.

Acceptance of the Teacher Dashboard will be based on the module's ability to accurately collect, aggregate, and display data across multiple student users. Evaluation will include tests for API integration, filtering accuracy by grade level, responsiveness of the user interface, and overall system performance during classroom-scale usage. Real and dummy data will be used to simulate normal operational conditions.

A final demonstration will be conducted for Teacher Rosario to walk through the dashboard features. The acceptance process includes both qualitative feedback and quantitative verification of performance indicators. Once the instructor verifies the system's functionality and data reliability, formal sign-off will occur through a written acceptance document. This ensures that the tool is ready for deployment and classroom use in support of monitoring student engagement with the CyberKids game.

7. Supporting process plans

This section outlines the essential supporting processes that ensure the CyberKids software project is developed and maintained with a high level of quality and traceability. These processes span the entire development lifecycle and include configuration management, quality assurance, verification and validation, documentation control, problem resolution, and subcontractor coordination (if applicable). These plans help safeguard the integrity of the project's deliverables and ensure alignment with stakeholder expectations, timelines, and resource constraints.

7.1. Configuration management plan

The **Configuration Management Plan** for the CyberKids project ensures that all project artifacts—codebases, documentation, assets, and configurations—are consistently identified, tracked, and controlled throughout the project lifecycle. It establishes a systematic approach to version control, change management, release documentation, and communication of changes across the team.

All software components, including those related to the Game Mission Module (developed on Roblox Studio using Lua) and the Teacher Dashboard Module (developed using React, Spring Boot, and MySQL), will be maintained using Git-based version control systems. GitHub will serve as the primary configuration management repository, enabling collaboration, issue tracking, branching, and pull request reviews.

Baseline versions of critical modules (e.g., UI prototypes, database schemas, level mechanics, dashboard features) will be established at key milestones and labeled using version tags (e.g., v1.0, v1.1). Once a work product is baselined, any changes must follow the formal change control process, including:

- **Change Request Submission** – Initiated by any developer or project member using GitHub Issues.
- **Change Impact Analysis** – Led by the project lead to assess technical, schedule, and quality implications.
- **Change Approval** – Reviewed during the team's weekly planning meeting or through asynchronous approval via GitHub pull request reviews.
- **Update and Merge** – Changes are merged into the main branch only after successful peer review and testing.
- **Notification** – Stakeholders are informed of approved changes via GitHub notifications and an internal project Slack channel.

A configuration log documenting all changes, version history, and status will be maintained in GitHub's release notes and internal documentation records. Automated continuous integration (CI) tools (such as GitHub Actions) will be used to enforce code standards and build verification upon each merge to the main branch.

This configuration management process ensures transparency, minimizes integration issues, and maintains alignment between deliverables and requirements across the modules of CyberKids.

7.2. Verification and validation plan

The **Verification and Validation Plan** for CyberKids will define the scope, techniques, tools, and responsibilities for V&V activities. These activities will be integrated throughout the project lifecycle to ensure quality, correctness, and fitness for purpose. The V&V activities are essential to confirm that the software system aligns with both functional and non-functional requirements, user expectations, and technical specifications.

To ensure systematic verification and validation, the following activities will be implemented across the project:

- **Traceability Analysis**: This process will ensure that all requirements (functional and non-functional) are linked throughout the lifecycle, from design to implementation, and ultimately to testing. For both the Game Missions and Teacher Dashboard modules, each requirement will be traceable from the initial specification to its corresponding design, code, and test cases.
- **Evaluation**: Evaluation will be performed at various stages to ascertain whether each component, module, and system meets its specification and user needs. Both the Game Missions and Teacher Dashboard modules will undergo comprehensive evaluations during design, implementation, and testing phases.
- **Interface Analysis**: As both modules rely on user interactions and interfaces, interface analysis will ensure that data exchanges, UI components, and user flows are consistent, correct, and meet the design specifications.
- **Testing**: Extensive testing will be carried out at multiple levels, including component testing, integration testing, system testing, and acceptance testing. Testing will cover all modules and interfaces, ensuring full functional and non-functional verification.
    - **Component Testing**: Focused on verifying individual software components in isolation.
    - **Integration Testing**: Ensures that different components work together as expected.
    - **System Testing**: Comprehensive testing of the integrated system to verify that the system meets the overall requirements.
    - **Acceptance Testing**: Formal testing performed with the customer to validate the system’s readiness for deployment.

7.2.1 Game Missions V&V

The Game Missions Module, developed in Roblox Studio, will undergo rigorous V&V to confirm that each interactive level correctly implements the educational objectives and integrates seamlessly with backend services. Traceability analysis will link each specified requirement—such as drag-and-drop classification, level-unlocking logic, and progress persistence—to the corresponding design schemas, Lua scripts, and test cases. Evaluation activities will assess whether game mechanics align with pedagogical goals and performance criteria, using both expert walkthroughs and student playtesting sessions. Interface analysis will verify that in-game UI elements and API calls to record progress satisfy the defined functional contracts.

Testing will proceed in four escalating stages:

- **Component Testing**: Individual scripts, UI widgets, and API endpoints will be tested in isolation within Roblox's Play Test environment.
- **Integration Testing**: The interplay between game logic and backend services (RESTful progress storage) will be validated in a staged environment.
- **System Testing**: Full playthroughs of each game level will be executed to ensure end-to-end functionality under realistic conditions.
- **Acceptance Testing**: Teacher Rosario and a select group of students will perform structured gameplay sessions, verifying both usability and educational efficacy.

V&V Plan for Game Missions

This section explains out V&V plan for each phase of software development.

Phase	**V&V Input**	**V&V Tasks**	**V&V Output**
Requirements	SRS, Use cases for game levels	Traceability analysis to link requirements to Lua scripts and design artifacts; Requirements evaluation; Interface analysis; Test plan creation	Traceability matrix; Requirements phase test plan
Design	Gameflow diagrams, UI mockups	Design traceability analysis; Design evaluation; Interface consistency checks; Test design generation	Design phase test plan; Validation reports
Implementation	Lua script source, compiled game builds	Code traceability analysis; Script evaluation; Interface analysis; Unit test generation; Component test execution	Component test cases and results; Test logs
Integration	Staging backend and game build	Integration test execution for gameplay–API interactions; Data flow validation	Integration test reports; Issue logs
System	Fully integrated game environment	End-to-end system testing; Performance and stability checks under sample class load	System test reports; Performance benchmarks
Acceptance	Final game build, test student group	Acceptance test execution with Teacher Rosario and students; Usability and pedagogical validation	Signed acceptance certificate; Feedback summary

7.2.2 Teacher Dashboard V&V

The Teacher Dashboard, implemented with React on the frontend and Spring Boot/Azure MySQL on the backend, requires its own V&V regime to guarantee accurate data representation and secure, responsive operation. Traceability analysis will map each dashboard requirement—such as data filters, summary tiles, and student detail views—to specific React components, REST API endpoints, and database schemas. Evaluation will involve expert reviews of UI/UX mockups and performance audits of query response times. Interface analysis will confirm that all API contracts (input parameters, JSON payloads) between the dashboard and backend services adhere to the OpenAPI specification.

Testing will follow the same four stages:

- Component Testing: Unit tests for React components and JUnit tests for Spring Boot controllers ensure each element functions correctly in isolation.
- Integration Testing: Combined tests of frontend fetching data from backend endpoints validate end-to-end data flow.
- System Testing: Full dashboard workflows, including login, data filtering, and chart rendering, will be executed in a staging environment.
- Acceptance Testing: Teacher Rosario will perform scripted scenarios—viewing class metrics, filtering by grade, and exporting reports—to confirm readiness for deployment.

V&V Plan for Teacher Dashboard

This section explains out V&V plan for each phase of software development.

Phase	**V&V Input**	**V&V Tasks**	**V&V Output**
Requirements	SRS, Dashboard use cases	Traceability analysis linking requirements to components and APIs; Requirements evaluation; Interface analysis; Test plan creation	Traceability matrix; Requirements phase test plan
Design	Figma wireframes, API specifications	Design traceability analysis; UI/UX evaluation; Interface contract checks; Test design generation	Design phase test plan; Validation reports
Implementation	React component code, Spring Boot source	Code traceability analysis; Code evaluation; Interface analysis; Unit and component test generation	Component test cases and results; Test logs
Integration	Staging frontend & backend integration	Integration test execution for UI–API interactions; Data consistency validation	Integration test reports; Issue logs
System	Deployed staging version	End-to-end system testing of dashboard workflows; Performance and security checks	System test reports; Performance/security metrics
Acceptance	Final deployed dashboard, sample data	Acceptance test execution with Teacher Rosario; Usability and functional validation	Signed acceptance certificate; Feedback summary

7.3. Documentation plan

This section defines the strategy for producing, reviewing, and distributing all project documentation for CyberKids. Each deliverable and non deliverable work product will be assigned an author and a reviewer, follow university or IEEE templates, and be baselined under version control in the team's GitHub repository. Initial drafts ("review copies") will be circulated for feedback, and final baseline versions will be approved by Teacher Rosario (adviser) before distribution to the development team, stakeholders, and archival in the project Wiki.

7.3.1 Game Missions Module

Throughout the development of the Game Missions Module, the team will produce, review, and baseline the following artifacts:

- **Statement of Work (SOW):** Outlines module scope, objectives, deliverables, and acceptance criteria.
- **Software Project Management Plan (SPMP):** Captures the management approach specific to the game module, including schedules, resources, and risk controls.
- **Software Requirements Specification (SRS):** Details the functional requirements for each game level (Data Leak Investigation, Password Fortress Defense, Cyber Escape Room).
- **Supplementary Specification (SS):** Describes nonfunctional requirements such as performance targets, usability goals, and security constraints.
- **Architecture Document:** Provides a high-level view of the game's component structure, including server-client interactions and asset pipelines.
- **Architecture Tradeoff Analysis Model (ATAM):** Evaluates design alternatives (e.g., asset streaming vs. preloading) against quality attributes.
- **Mini-Software Risk Evaluation (SRE):** Identifies technical and schedule risks (e.g., Roblox API limitations) and defines mitigation plans.
- **Use Case Diagram & Descriptions:** Graphical and textual scenarios capturing actor interactions (students, game engine).

- **Detailed Level Design (DLD):** Includes sequence diagrams (Lua script flows) and class diagrams for core game objects.
- **Entity Relationship Diagram (ERD):** Models the schema for storing player progress and scores in the backend database.
- **System Integration Plan:** Describes the process for integrating Roblox Studio builds with the Spring Boot API and database.
- **Acceptance Confirmation Documentation:** Records formal sign-off by Teacher Rosario after each level's demonstration.
- **Status Reports:** Biweekly progress updates, covering completed tasks, next steps, and risk status.
- **Test Scripts & Test Results:** Detailed test cases for component, integration, system, and acceptance testing, with executed test logs.
- **Risk Management Statement:** Ongoing log of identified risks, their probability and impact assessments, and mitigation actions.
- **Defect Log:** Records all bugs found during testing, their severity, current status, and resolution.
- **Metrics Log:** Tracks metrics such as defect density, test coverage, and student engagement statistics.
- **Inspection Reports:** Results from peer reviews of requirements, design, code, and test artifacts conducted each iteration.

7.3.2 Teacher Dashboard Module

For the Teacher Dashboard Module, the following documents will be created and maintained under version control:

- **Statement of Work (SOW):** Defines dashboard scope, deliverables, and acceptance criteria for monitoring Grades 5–6.
- **Software Project Management Plan (SPMP):** Details planning, scheduling, and risk management specific to the dashboard.
- **Software Requirements Specification (SRS):** Specifies functional requirements—class overview, detailed student metrics, filters, and exports.
- **Supplementary Specification (SS):** Covers nonfunctional requirements like performance (response times), security (role-based access), and availability.
- **Architecture Document:** Describes the three-tier architecture (React frontend, Spring Boot backend, Azure MySQL).
- **Architecture Tradeoff Analysis Model (ATAM):** Compares deployment options (monolithic vs. microservices) and their trade-offs.
- **Mini-Software Risk Evaluation (SRE):** Identifies and mitigates risks such as API versioning issues or database schema changes.
- **Use Case Diagram & Descriptions:** Illustrates teacher and admin interactions with the dashboard.
- **Detailed Level Design (DLD):** Sequence diagrams for API calls and class diagrams for React components and Java services.
- **Entity Relationship Diagram (ERD):** Models tables for users, roles, scores, and logs in Azure MySQL.
- **System Integration Plan:** Outlines CI/CD processes linking GitHub commits to Vercel and Render deployments.
- **Acceptance Confirmation Documentation:** Formal sign-off by Teacher Rosario upon successful UAT of dashboard features.
- **Status Reports:** Sprint summaries with completed stories, impediments, and next sprint goals.

- **Test Scripts & Test Results:** Comprehensive test cases for unit, integration, UI, performance, and security tests, with results logged.

- **Risk Management Statement:** Catalog of dashboard-specific risks and mitigation strategies.
- **Defect Log:** Tracking of all issues found during development and testing, with resolution status.
- **Metrics Log:** Collection of metrics such as page-load times, API error rates, and test coverage.
- **Inspection Reports:** Documentation of peer review findings for requirements, design documents, and code at each milestone.

7.4. Quality assurance plan

This section provides a quality assurance plan for the **CyberKids** project, ensuring that the software meets its commitments as defined in the Software Requirements Specification (SRS), Software Project Management Plan (SPMP), and any supporting plans. This quality assurance plan outlines the procedures, roles, and responsibilities for ensuring the project adheres to specified standards, procedures, and guidelines throughout its lifecycle.

7.4.1 Quality Assurance Objectives

The primary objective of this quality assurance plan is to ensure that the CyberKids software:

1. Meets the functional and non-functional requirements outlined in the SRS.
2. Adheres to the project management guidelines and schedules described in the SPMP.
3. Is developed following best practices, standards, and quality processes throughout the software development lifecycle.
4. Delivers a product that is reliable, efficient, and secure for the target audience (Grades 5-6 students) and their teacher, Teacher Rosario.

7.4.2 Quality Assurance Process Overview

The quality assurance activities will be integrated into each phase of the project. They include, but are not limited to, the following:

- **Analysis**: Detailed examination of the software requirements, designs, and code to ensure compliance with specifications.
- **Inspections**: Formal reviews of requirements, designs, and code to identify issues early and ensure adherence to the quality standards.
- **Reviews**: Regular project status reviews, design walkthroughs, and code reviews to assess progress and quality.

- **Audits**: Periodic audits to verify adherence to the project plan, requirements, and quality standards.
- **Assessments**: Continuous assessment of the development process and product quality to identify improvements.

7.4.3 Key Quality Assurance Procedures

- **Requirements and Design Reviews**: A review process will be conducted for both the Software Requirements Specification (SRS) and the design documents (including Use Case Diagrams, ERDs, and Sequence Diagrams). The aim is to ensure that the project scope, user needs, and system design are properly defined, understood, and adhered to.
- **Code Inspections and Reviews**: Code inspections will be done regularly to ensure compliance with coding standards, readability, and maintainability. Peer reviews will be conducted to assess code quality, catch errors early, and improve code efficiency.
- **Testing**: A thorough testing procedure will be established, including unit testing, integration testing, system testing, and acceptance testing. These tests will verify that the system functions as intended and meets the specified requirements.
- **Configuration Management**: Configuration management will be implemented to ensure that all project artifacts are properly tracked, versioned, and stored. This will include source code, documentation, and test scripts. Regular backups will be maintained to safeguard project materials.
- **Risk Management**: Risks related to the software's development will be identified, documented, and addressed promptly. The Risk Management Statement will outline specific risks, their mitigation strategies, and their impact on project timelines and deliverables.
- **Audit and Compliance Checks**: Periodic internal and external audits will be carried out to ensure compliance with the project's quality standards and guidelines. Audits will cover both the software product and the process used to develop it.

7.4.4 Relationship with Other Processes

The quality assurance activities are interconnected with other processes in the software development lifecycle:

- **Verification and Validation (V&V)**: Quality assurance procedures will support the verification and validation processes to ensure the software meets all functional and non-functional requirements. This will include reviewing test results, validating user requirements, and ensuring the product performs as expected.
- **Review and Audit Processes**: Quality assurance reviews and audits will be aligned with regular project reviews and audits to assess adherence to the project plan, requirements, and quality standards. This ensures transparency and accountability throughout the project.
- **System Engineering**: The quality assurance plan will collaborate with the system engineering team to ensure that the system architecture and components meet quality requirements. This will involve both design-level reviews and system-level integration testing.

7.4.5 Quality Assurance Roles and Responsibilities

The following roles are responsible for quality assurance activities:

- **Quality Assurance Manager**: Oversees the quality assurance process, ensuring all procedures are followed and that deliverables meet the required standards.
- **Project Manager**: Ensures that the project team adheres to the project management plan (SPMP) and integrates quality assurance processes into the overall project timeline.
- **Developers**: Participate in code reviews, testing, and design inspections to ensure the software is built according to the specified standards and requirements.
- **Testers**: Execute the testing procedures, report defects, and validate the software against the specified requirements.
- **Teacher Rosario (Adviser)**: Will be involved in reviewing the user interfaces and ensuring the project meets the educational goals for Grades 5-6 students.

7.4.6 Tools and Techniques

The following tools and techniques will be employed to support quality assurance:

- **Project Management Tools**: Tools like Trello or Asana will be used for task tracking, project planning, and progress reporting.
- **Version Control Systems**: Git will be used to manage source code versions and ensure proper configuration management.
- **Automated Testing Tools**: Tools such as Selenium or Jest will be used to automate unit and integration testing to ensure accuracy and efficiency.

- **Code Analysis Tools**: Tools like SonarQube will be used to analyze code quality, detect bugs, and assess code coverage.
- **Documentation Tools**: Confluence or Google Docs will be used to document test plans, audit reports, and other quality assurance documents.

7.4.7 Continuous Improvement

Continuous improvement will be achieved through regular feedback loops, where quality assurance activities will be reviewed and refined after each project phase. Lessons learned from audits, inspections, and test results will be incorporated into subsequent project phases to enhance the software development process and product quality.

This **Quality Assurance Plan** will ensure that the CyberKids project maintains high standards throughout its development, delivering a high-quality product that meets the educational needs of Grades 5-6 students and supports Teacher Rosario’s goals effectively.

7.5. Reviews and audits

This section defines the structured review and audit activities for the CyberKids project. Reviews and audits are integral to ensuring that the project remains aligned with its objectives, complies with defined processes, and delivers high-quality software. The following review types will be conducted on the schedule and with the resources detailed below.

1. **Joint Acquirer–Supplier Reviews**

   At the conclusion of each major phase (requirements, design, implementation, testing, and delivery), a formal review will be held jointly between the development team and the acquirer, represented by Teacher Rosario. These sessions will include a demonstration of deliverables, comparison against the acceptance criteria, and sign-off on phase completion. The reviews will be scheduled two weeks before each phase milestone, with preparation led by the Project Manager and presentation by the module leads. Teacher Rosario's feedback and any corrective actions will be recorded and addressed in the subsequent iteration.

2. **Management Progress Reviews**

   Weekly status meetings chaired by the Project Manager will monitor progress against the project schedule and budget. These management reviews will cover earned value metrics, risk updates, and resolution of any impediments. Attendance will include the Project Manager, Scrum Master, QA Manager, and key technical leads. Minutes will be distributed within 24 hours, and action items will be tracked in the project's issue-tracking system.

3. **Developer Peer Reviews**

   All code, design documents, and test plans will undergo peer review before merging into the main branch. Using GitHub pull requests, each submission will be reviewed by at least two developers, checking for correctness, adherence to coding standards, and potential defects. Peer reviews will be completed within three business days of submission. Reviewers will annotate feedback directly in the pull request; authors will resolve comments and obtain final approvals before integration.

4. **Quality Assurance Audits**

   Midway through each sprint and at the end of each major release, the QA Manager will perform an audit of the quality processes. This audit will verify that V&V activities (test execution, defect logging, traceability maintenance) are current and complete. Audit findings will be documented in a quality audit report, reviewed by the Project Manager, and any nonconformities will be assigned to team members for resolution within one sprint cycle.

5. **Acquirer-Conducted Reviews and Audits**

   In addition to joint phase reviews, Teacher Rosario may conduct ad-hoc audits of project artifacts (such as the SRS, game prototypes, or the teacher dashboard) to ensure pedagogical alignment and usability. These audits will be scheduled as needed, with at least three business days' notice to the development team. Findings will be formally documented, and action items will be integrated into the task backlog for prioritization.

External Agencies

No external regulatory or certification agencies are required for this educational capstone project. All approvals and audits will be conducted internally by Cebu Institute of Technology University stakeholders, including Teacher Rosario and the project adviser.

7.6. Problem resolution plan

This section defines the structured approach for capturing, analyzing, prioritizing, and resolving all software problems identified during the CyberKids project. Problems include defects, configuration issues, process gaps, and change requests. The plan specifies the tools, roles, workflows, and metrics needed to efficiently address and track problems from initial report through final closure.

Problem Reporting and Logging

All team members, testers, and stakeholders—especially Teacher Rosario—will report problems via the project's GitHub Issues tracker. Each issue must include a clear title, description, steps to reproduce (for defects), severity level (Critical, Major, Minor), module affected (Game Missions or Teacher Dashboard), and any relevant attachments (screenshots, logs). The Scrum Master is responsible for ensuring that newly reported issues are triaged within 24 hours and properly categorized.

Triage and Prioritization

During the daily stand-up, the Development Team Lead and QA Manager will perform initial triage, assigning each issue to one of three queues: Blocker (blocks development or testing), High (significant functional impact), or Normal (minor or cosmetic). The Configuration Manager will verify that each issue is linked to the appropriate baseline version or branch. The Change Control Board (CCB), consisting of the Project Manager, QA Manager, and a representative from Teacher Rosario, will meet weekly to review Blocker and High priority issues, authorize any scope or schedule adjustments, and assign target resolution sprints.

Analysis and Resolution

Assigned developers will analyze each issue's root cause—whether code defect, design gap, or environment configuration—and estimate effort for correction. All proposed fixes require peer review via GitHub pull requests that reference the original issue number. Once code changes pass automated unit and integration tests, the fix is merged into the development branch and deployed to the staging environment for validation by QA and verification by the reporter.

Verification and Closure

QA Engineers will execute regression and acceptance tests against each resolved issue in the staging environment. If the fix meets the acceptance criteria without introducing regressions, the CCB will approve the ticket for production deployment. Upon successful deployment and final confirmation from Teacher Rosario (if applicable), the issue is marked "Closed." All changes and closure comments are recorded in the issue tracker.

Metrics and Continuous Improvement

Effort spent on problem analysis, rework, and resolution is logged as time entries in GitHub Issues. The Scrum Master compiles weekly metrics—average time to close, reopened issue rate, and defect density per module—and presents them in the sprint review. These metrics inform process refinements, such as adjusting test coverage or strengthening peer-review practices, thereby driving continuous improvement throughout the CyberKids development lifecycle.

7.7. Subcontractor management plan

The **CyberKids** project is designed and developed primarily by the internal project team under the supervision of **Teacher Rosario**, the Grade 5–6 adviser. However, if any third-party subcontractors are engaged to support the development—such as for the creation of specialized graphics, sound effects for the game modules, or external testing—this section defines the criteria and process for their selection, engagement, and oversight.

Subcontractor Selection Criteria

Subcontractors will only be considered if they:

- Demonstrate prior experience in educational software or gamified learning tools.
- Provide a portfolio or demo showcasing relevant past work.
- Can commit to the project timeline and communication standards.
- Offer competitive pricing and flexible revision terms.
- Are able to comply with data privacy and child-friendly content regulations.

Subcontractors will be evaluated based on:

- Technical capability and specialization in the required service.
- Adherence to cybersecurity and educational content standards.
  Timeline guarantees and milestone-driven delivery model.
- Availability for coordination meetings or clarifications.

Management Plan for Subcontractors

Each subcontractor engagement will follow a mini-SPMP tailored to the scope of work. The plan will include the following key components:

- **Requirements Management**: Subcontractors will be given precise task specifications (e.g., “animated sprites for level 1 mission”) derived from the project’s SRS and DLD documents. A walkthrough session will be conducted by the Project Manager to clarify expectations.
- **Monitoring Technical Progress**: Subcontractors must deliver progress updates weekly via email or shared drive folders. Major deliverables (e.g., all graphics for Mission 2) will undergo team review before acceptance.
- **Schedule and Budget Control**: Deliverables are milestone-based, with partial payments made upon acceptance. Late or low-quality outputs will result in penalties or replacement. A strict timeline with buffer periods will be established.
- **Product Acceptance Criteria**: Each deliverable must:

    - Match the agreed-upon resolution, file format, or quality.
    - Integrate seamlessly into the Unity game engine or the teacher dashboard.
    - Receive final approval from the Design Lead and Teacher Rosario.

- **Risk Management Procedures**:
    - Backup vendors will be identified for critical assets (e.g., main character sprites).
    - Delays beyond one week trigger escalation and potential subcontractor replacement.
    - All subcontractor work must be free from licensing issues and must not contain AI-generated content without team consent.

7.8. Process improvement plan

Process improvement efforts in the **CyberKids** project are divided into two focused areas: **Game Missions** and the **Teacher Dashboard**. Each module has distinct users and workflows, so separate improvement strategies are outlined for each. Improvements are based on feedback cycles, error tracking, and team reflections throughout the development lifecycle.

7.8.1 Game Missions Module (Grades 5–6)

The Game Missions were designed to provide interactive and educational content in Math, Science, and Filipino. Given the nature of young users, gameplay logic, design clarity, and ease of navigation were key areas under evaluation.

Planned Assessments & Improvements:

1. **Usability Testing Feedback from Students**
    - Conducted during classroom simulations with Grades 5–6 learners.
    - Observations included moments of confusion in navigation and overwhelming level design in early builds.
    - Improvements: Simplified menus, added tutorial popups, and clearer visual cues in missions.

2. **Bug Frequency & Tracking Logs**
    - Frequent visual or audio glitches were logged by the development team.
    - Improvements: Introduced a checklist-based QA process before merging any update to reduce overlooked bugs.

3. **Content Engagement Metrics**
    - Tracked how long students stayed on each subject area and whether missions were completed.
    - Improvement Action: Adjusted difficulty curve and rewards system to better maintain attention span.

4. **Team Reflection Meetings**
    - Weekly reviews by student developers to log what delayed progress.
    - Key Takeaway: Switching from individual file uploads to a shared asset library improved efficiency in design integration.

Documentation: Lessons learned and actionable steps are documented in the Game Missions PIP (Process Improvement Plan) file.

7.8.2 Teacher Dashboard Module (for Teacher Rosario)

The Teacher Dashboard enables Teacher Rosario to monitor student progress, view scores, update content, and send announcements. This admin-oriented module requires clarity, reliability, and flexibility.

Planned Assessments & Improvements:

1. **Faculty Feedback Review**
    - Teacher Rosario provided walkthrough feedback during review sessions.
    - Observations: Initial dashboard was too text-heavy and lacked quick overview metrics.
    - Improvement: Introduced a summary panel for recent scores, and graphical charts for better insight.

2. **Error & Change Request Logs**
    - Midway through development, data filtering in the dashboard caused confusion.
    - Improvement: Revised data table designs to allow dynamic filtering and pagination.

3. **Integration Testing Logs**
    - Problems emerged when syncing student scores with mission data.
    - Improvement: Enhanced backend validations and introduced error logging for any mismatch cases.

4. **Role Review & Task Delegation**
    - It was noted that some frontend devs were unfamiliar with admin interface design.
    - Process Improvement: Conducted a short team skill session to align design expectations for admin tools.

Documentation: All updates and strategies for better development workflow are captured in the Teacher Dashboard PIP (Process Improvement Plan) document.

8. Additional Plans

To ensure that the *CyberKids* software system meets its functional and non-functional requirements while aligning with institutional standards, several additional plans have been established. These plans address product safety, user privacy, training needs, system installation, integration, transition, maintenance, and ongoing support. The system is divided into two major components: the **Game Missions Module** for Grade 5–6 students, and the **Teacher Dashboard Module**, designed specifically for Teacher Rosario as the class adviser.

In terms of **safety and privacy**, the CyberKids system is built with student protection as a top priority. No sensitive or personally identifiable information (PII) is collected from students. User profiles rely solely on first names and class codes, ensuring anonymity and compliance with school privacy policies. All data transactions are encrypted using secure HTTPS protocols.

The **user training plan** is tailored to each module's audience. For students, training is embedded directly within the game through intuitive tutorial levels, visual instructions, and interactive guidance. This design ensures that young learners can navigate and enjoy the missions with minimal external assistance. Meanwhile, Teacher Rosario receives access to a comprehensive quick-start guide and tooltip-based onboarding within the dashboard interface, complemented by a one-on-one orientation session.

For **installation and deployment**, CyberKids is delivered as a web-based application, requiring no local installation. This simplifies rollout and ensures compatibility with standard school-issued tablets and desktop computers. The system is optimized to run on low-specification devices to promote inclusivity and accessibility.

The **support and maintenance plan** outlines clear protocols for issue reporting and resolution. Teacher Rosario can log technical issues through the dashboard, which are automatically forwarded to the development team. Minor updates and patches are scheduled monthly, while more critical fixes are handled on a case-by-case basis. A defect log tracks all reported issues to ensure transparency and traceability.

Regarding **integration**, the dashboard connects directly with the game module’s backend, providing real-time updates on student performance and activity logs. This seamless integration reduces manual

record-keeping and improves data accuracy.

To manage system transition, manual tracking processes traditionally used by Teacher Rosario are being phased out in parallel with the new system's adoption. A short dual-tracking phase ensures data integrity before fully shifting to the digital platform.

Lastly, the **product support plan** includes continued documentation updates, teacher re-orientation for future academic years, and the capability to onboard new teacher-users should the system expand in scope. A product maintenance roadmap has also been drafted to anticipate future needs such as content updates, feature enhancements, and scalability improvements.Together, these additional plans ensure that the CyberKids system is not only technically sound but also educationally meaningful, user-centric, and sustainable over the long term.

9. Plan Annexes

To maintain clarity and focus in the main body of this Software Project Management Plan (SPMP), supporting materials and detailed references are included in the annexes. These documents provide in-depth information that supports planning, execution, and evaluation of the **CyberKids** project, while preventing the main text from being overloaded with technical details.

The annexes include:

- **Annex A** – Statement of Work (SOW)
- **Annex B** – Software Requirements Specification (SRS)
- **Annex C** – Supplementary Specification (SS)
- **Annex D** – Detailed Level Design (DLD) Documents (Sequence and Class Diagrams)
- **Annex E** – Entity Relationship Diagram (ERD)
- **Annex F** – Use Case Diagrams and Descriptions
- **Annex G** – Test Scripts and Test Results
- **Annex H** – Risk Management Statement and Defect Log
- **Annex I** – Architecture and ATAM Evaluation
- **Annex J** – Inspection Reports and Status Reports
- **Annex K** – System Integration Plan and Acceptance Confirmation
- **Annex L** – Quality Assurance Documentation

- **Annex M** – Problem Resolution and Process Improvement Plans
- **Annex N** – Subcontractor Management Details (if applicable)
- **Annex O** – Metrics Log

These annexes are stored in the project documentation repository and are referenced where relevant throughout the SPMP. They provide detailed support for planning decisions, project tracking, and traceability throughout the Software Development Life Cycle (SDLC).

10. Index