CEBU INSTITUTE OF TECHNOLOGY UNIVERSITY

COLLEGE OF COMPUTER STUDIES

Software Project Management Plan

for

CyberKids

TABLE OF CONTENTS

1. Overview

1.1. Project Summary

1.1.1. Purpose, scope and objectives

The purpose of the *CyberKids* project is to provide an engaging and educational digital game tailored for elementary students (Grades 5–6), aimed at introducing and reinforcing fundamental cybersecurity concepts. The game leverages a gamified learning environment to foster awareness and understanding of digital safety in a fun, interactive way.

Scope:

The *CyberKids* platform comprises several key components, designed to offer both an interactive learning experience for students and a monitoring tool for educators. These include:

- **Module 1: Gamified Cybersecurity Game Levels** (Roblox)
    - Level 1 – Online Privacy & Safety: Sort digital information into “Safe to Share” vs. “Not Safe to Share.”
    - Level 2 – Password Security: Collect password fragments in a treasure-hunt mini-game and defend against a simulated hacker bot.
    - Level 3 – Phishing & Scam Awareness: Navigate a virtual escape room to identify phishing emails, fake login pages, and social-engineering scams.

- **Module 2: Student Real-Name Registration & Profile Viewer** (Roblox + Backend)
    - Allow students to register their real names—linked to their Roblox IDs—for teacher identification.
    - Provide a read-only profile view showing achievements and unlocked levels (no direct edits).

- **Module 3: Teacher Web Dashboard** (React.js + APIs)
    - Monitor student progress, scores, and mission completion in real time.
    - Generate downloadable CSV/PDF reports.
    - Lock or unlock individual game modules per student.
    - Reassign students to different levels as needed.
    - Provide a read-only profile view showing achievements and unlocked levels (no direct edits).
- **Module 4: Teacher Notification System** (React.js + WebSocket + APIs)
    - Create, edit, and delete in-dashboard notifications.
    - Real-time delivery via WebSocket with GET-based fallback on reconnect.
    - Mark notifications read/unread.

System Components:

- **Backend System** – Java Spring Boot RESTful APIs, Azure-hosted MySQL/PostgreSQL, JWT-based authentication, bcrypt password hashing.

- **Frontend Interfaces:**
    - React.js with Tailwind and shadcn/ui for the teacher dashboard; Roblox UI scripts (Lua) for the game client.

Objectives:

The main objectives of the *CyberKids* project are to:

- Provide **cybersecurity education** to Grades 5–6 students of **CIT-University** through a gamified learning approach.
- Ensure a **stable, responsive, and accessible** learning platform that is suitable for classroom and at-home use.
- Enable **teachers** to effectively monitor student learning, assess outcomes, and integrate cybersecurity topics into their lesson plans.
- Maintain full **compliance with educational standards** and the **Data Privacy Act of 2012 (RA 10173)** to safeguard student data and user interactions.

1.1.2. Assumptions and constraints

The planning and execution of the CyberKids project are built upon a set of foundational assumptions that must hold true for the system to function as intended. In parallel, the team must also work within certain constraints technical, legal, and operational that limit the project's scope and flexibility.

Assumptions:

The following assumptions are considered foundational to the success of the project and are expected to hold true throughout the project lifecycle:

- **Hardware Availability**: Students and teachers have access to desktop or laptop computers that meet the system's minimum requirements.
- **Basic Digital Literacy:** Students (Grades 5–6) are assumed to possess basic computing skills, including using a mouse and keyboard, navigating simple UIs, and following instructions.

- **Operating System Compatibility:** The system will run on Windows 10 or later, with support for Chrome, Firefox, and Edge browsers.
- **Internet Access:** A stable internet connection is available for features such as leaderboard updates and dashboard synchronization.
- **Teacher Involvement:** Teachers will actively use the dashboard for progress monitoring and integrating CyberKids into classroom activities.
- **Server Uptime:** The backend server and database services are assumed to be consistently online during school hours.

Constraints:

The project is also subject to several limitations that define its operational boundaries:

- **Platform Limitation:** The game is developed for desktop platforms only; no support is provided for mobile or tablet devices.
- **Performance Requirements:** The game must operate smoothly on low- to mid-range hardware with at least an Intel Core i3 processor, 4 GB RAM, and integrated graphics.
- **Data Privacy:** All user information must be protected under the Data Privacy Act of 2012 (RA 10173); no real personal data is collected.
- **No External Integration:** The system is self-contained; it does not integrate with social media platforms or third-party systems.

1.1.3. Project deliverables

The CyberKids development team will provide the following deliverables to the client in accordance with the project's scope and contractual obligations. These work products cover all essential components of the system, its documentation, and supporting materials necessary for evaluation, deployment, and future maintenance:

Core Software Deliverables

- **CyberKids Game (Final Build):**
  A fully functional, playable version of the CyberKids game, deployed on the **Roblox platform** and accessible through a secure, private game link.
- **Source Code and Assets Package:**
  A compressed archive containing:
  - Roblox Studio project files
  - Custom scripts and modules
  - Game models, animations, audio, and UI elements

Technical and Design Documentation

- **Software Requirements Specification (SRS):**
  A comprehensive document detailing the educational goals, game features, functional and non-functional requirements, and system environment.

- **Game Design Document (GDD):**
  Includes:
  - High-level software architecture
  - Gameplay mechanics and level design
  - User interaction flows and UX elements
  - Visual design specifications

End-User Documentation

- **Student User Guide:**
  Instructions for gameplay, control schemes, learning objectives, and navigation through the levels.

- **Teacher Manual:**
  Guidance for integrating CyberKids into lessons, including:
  - Monitoring tools
  - Suggested discussion prompts
  - Alignment with curriculum goals

Testing and Quality Assurance

- **Software Test Plan (STP):**
  Outlines the testing strategy, QA methodologies, and test environments.
- **Test Case Documentation and Results:**
  Includes detailed test cases, execution results, bug reports, and feedback from pilot testing sessions with students.

Project Communication Materials

- **Presentation Materials:**
  Demo slide decks, walkthroughs, and visuals used for project demos, stakeholder briefings, or educational board presentations.

Project Management Artifacts

- **Software Project Management Plan (SPMP):**
  The formal project plan, outlining milestones, resources, and management processes.
- **Gantt Charts and Schedules:**
  Detailed project timelines, task dependencies, and progress tracking.
- **Risk Assessments and Logs:**
  Identification of potential project risks, mitigation strategies, and ongoing monitoring.
- **Weekly Status Reports:**
  Summarized progress updates delivered to stakeholders throughout the development cycle.

Final Documentation and Handoff

- **Final Project Report:**
  A complete overview of the development lifecycle, challenges encountered, lessons learned, and reflections on outcomes.
- **Project Handoff Package:**
  Contains maintenance notes, setup instructions, and guidance for future improvements or expansion.

All deliverables will be submitted via a shared cloud folder (Google Drive) and, where appropriate, mirrored in a Github repository. Documents will be provided in PDF format, with source files in their original formats. Sensitive or student-related data will be anonymized and handled per institutional privacy requirements.

1.1.4. Schedule and budget summary

Milestone	Date (initiation / completion)
Project Planning	February 7, 2025
Requirements Gathering	February 13, 2025
Requirements Finalization	March 14, 2025
SRS Completion	March 21, 2025
SDD Completion	March 23, 2025
Development Kick-off	March 25, 2025
Testing Phase	April 15, 2025
Deployment and Evaluation	May 10, 2025

Budget Assumptions:

- The project uses only free or open-source tools (e.g., Roblox Studio, GitHub, MySQL, Java, React).
- No monetary compensation for developers academic projects only.
- Hosting is handled internally or via university-provided infrastructure.

1.2. Evolution of plan

The SPMP will undergo revisions at key milestones. Updates are managed using version control via Google Drive and GitHub repositories.

Version	Primary Author(s)	Description of Version	Date Expected
1.0	Baguio Cultura Dedumo Vequiso	Initial draft	March 20, 2025
1.1	Cultura	Post-SRS/SDS Integration	Apr 5, 2025
2.0	Cultura	Finalized	May 5, 2025

[Table 1.2] Evolution Plan

2. References

[1] IEEE Std 830-1998 – *Recommended Practice for Software Requirements Specifications*

[2] IEEE Std 1058-1998 – *Standard for Software Project Management Plans*

[3] ReactJS Glossary – https://legacy.reactjs.org/docs/glossary.html

[4] Roblox API Documentation – https://create.roblox.com/docs/reference/engine

[5] NIST Cybersecurity Framework – https://www.nist.gov/cyberframework

[6] CyberKids Software Requirements Specification, v1.2

[7] CyberKids Software Design Description, v1.1

[8] CIT-U Data Privacy Policy (RA 10173 Compliance Manual)

[9] System Design Description - https://drive.google.com/file/d/1GGWaddU9CwgVXmu7-Du0au1EHyjunzrQ/view?usp=drive_link

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 outlines the organizational structure for the CyberKids project, including both internal and external interfaces. It defines the roles and responsibilities within the project and clarifies the relationships with external entities.

4.1. External structure

The CyberKids project interacts with several key external entities, each playing a crucial role in the successful development, deployment, and evaluation of the project:

- **Parent Organization:**

  The project is conducted under the **Cebu Institute of Technology University (CIT-U)**, which provides the overarching academic framework, resources, and ensures compliance with institutional policies and standards for educational projects.

- **Faculty Advisor / Project Evaluator:**

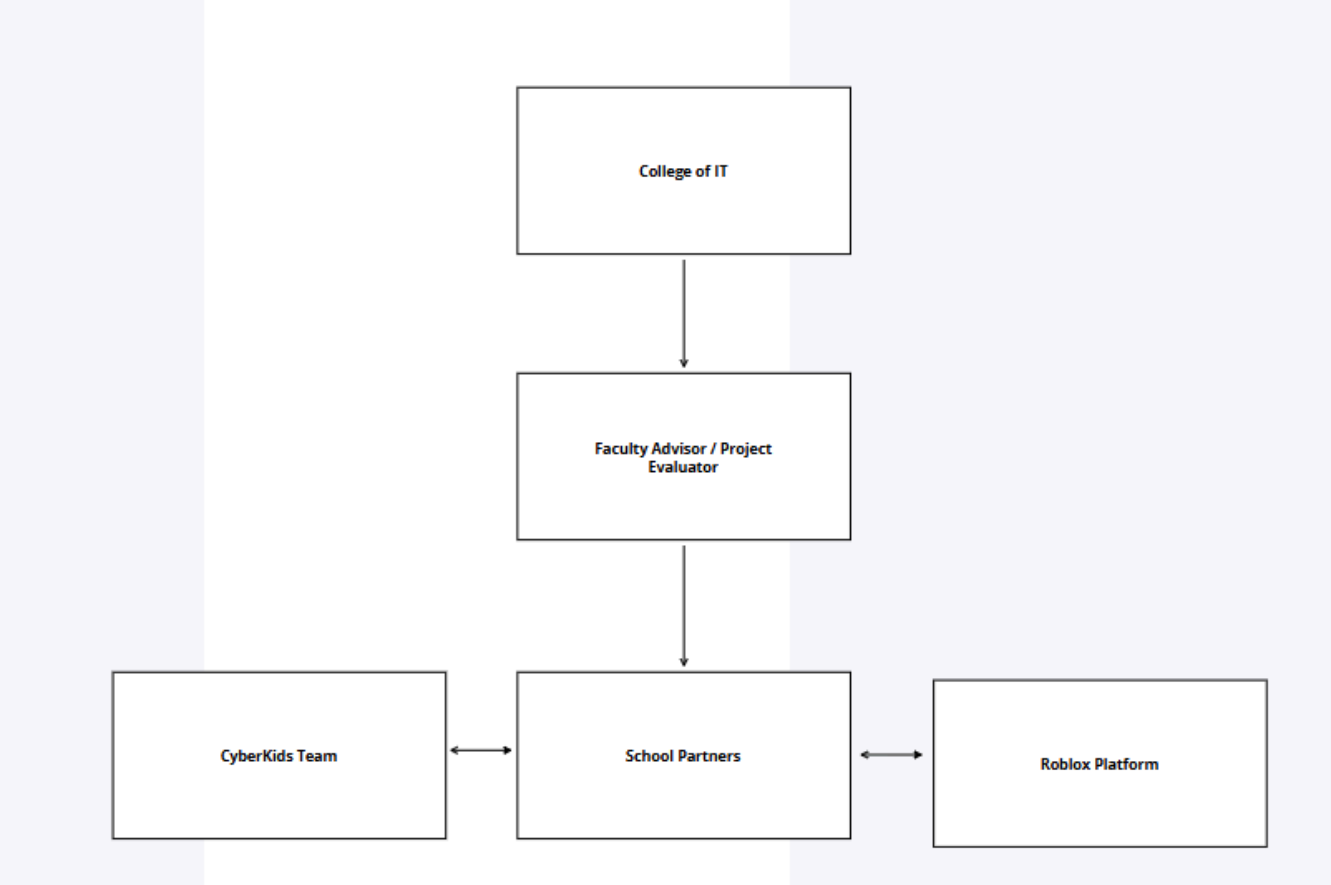
  The **Faculty Advisor** acts as the academic supervisor, offering guidance, expertise, and mentorship throughout the development process. The advisor will also assess project milestones, providing feedback on technical and educational aspects of the project, and conduct final evaluations upon completion.

- **School Stakeholders (Teachers and IT Coordinators):**

  Teachers and **IT Coordinators** from the university will be actively involved during the **pilot testing phase**, where they will provide valuable feedback on the educational quality, user experience, game clarity, and student engagement. Their input will help refine the project before it is fully deployed.

- **Roblox Platform:**

  The **Roblox platform** serves as the hosting environment for the CyberKids game. The project is subject to Roblox's platform guidelines, policies, and constraints regarding game development, publishing, and content management. These external rules and technical limits must be considered during the game design and deployment phases.



4.2. Internal structure

The **CyberKids** project is managed by a dedicated student team that collaborates closely to design, develop, test, and document the educational game based on the Roblox platform. The team follows a flat organizational structure that encourages open communication, collaboration, and efficiency, with clearly defined roles and responsibilities.

The internal team structure is organized as follows:

- **Project Manager (PM):**
  Oversees the entire project, ensuring that milestones are met, timelines are followed, and resources are allocated effectively. The PM also acts as the main point of contact for external stakeholders.

- **Game Developers:**
  Responsible for the creation of the game's core functionality, using **Lua** for scripting and **Roblox Studio** for game development. This role also involves debugging and refining the game's interactive elements to ensure a smooth player experience.

- **Frontend Developers:**
  Focus on the design and implementation of the **teacher dashboard**, using **React.js** for the web interface. This role includes ensuring that the dashboard is user-friendly and meets the needs of teachers for monitoring student progress.

- **Backend Developers:**
  Manage the server-side logic, implementing the **Java Spring Boot** framework for developing REST APIs and database management using **MySQL**. They are responsible for ensuring smooth data flow between the game and the teacher dashboard.

- **Educational Content Designers:**

  Experts in cybersecurity who work alongside the game developers to create meaningful educational content. They ensure that the game's levels and challenges effectively teach the target audience (Grades 5-6) key cybersecurity concepts.

- **Quality Assurance (QA) Specialists:**

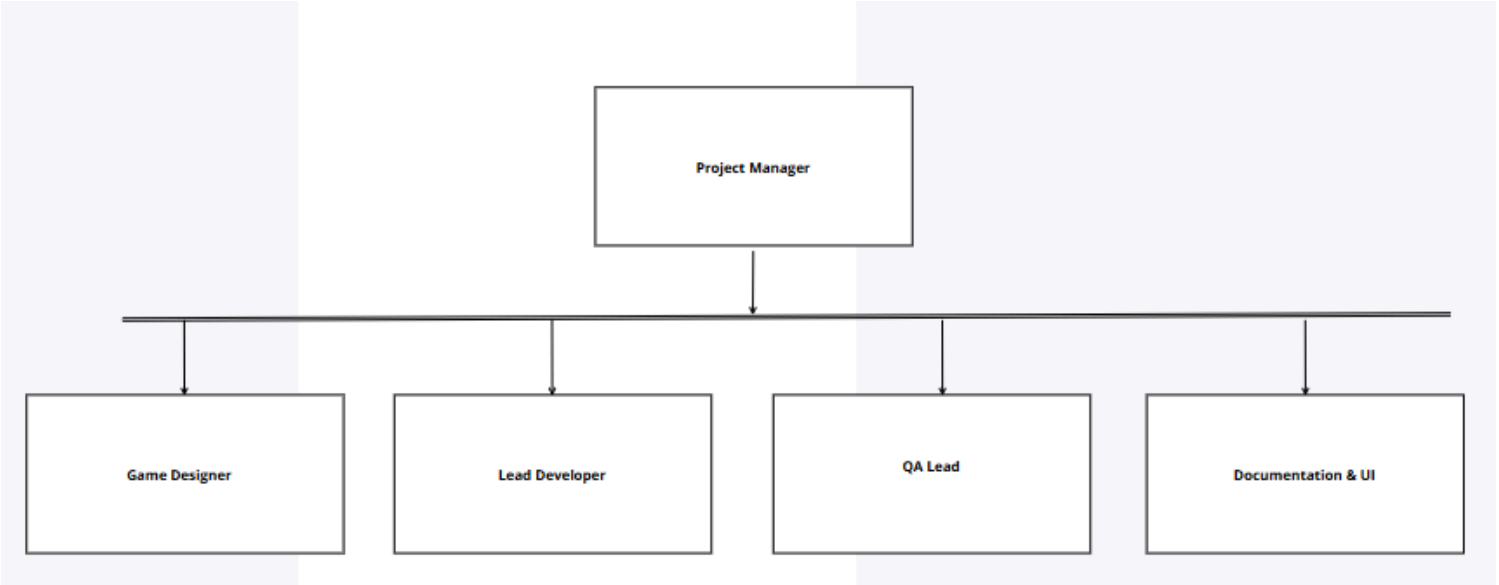
  Test the game and teacher dashboard to ensure functionality, usability, and educational effectiveness. They are also responsible for identifying bugs, performance issues, and ensuring compliance with project standards.

- **Documentation Specialists:**

  Responsible for producing clear and comprehensive documentation, including the Software Requirements Specification (SRS), Game Design Document (GDD), and user guides for both students and teachers.

- **Configuration and Release Management:**

  Ensures the consistent version control of the project's assets, source code, and documentation, using tools like **GitHub**. This role also manages the deployment process and ensures that the game is published on the **Roblox platform** in accordance with guidelines.



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:

s

TEAM MEMBER	ROLE & RESPONSIBILITIES	
	SEM 1	**SEM 2**
Jesson Cultura	**Project Lead** - serves as the single point of accountability for CyberKids, overseeing scope, schedule, and stakeholder communication while architecting both frontend and backend solutions.	Oversee the finalization of all project components in Semester 2, including documentation, testing, and game development. This ensures a polished, fully functional product ready for deployment and evaluation.
Emmanuel Dedumo	**Game Developer** - implements and optimizes in-game mechanics, UI, and backend integrations within Roblox, translating design specifications into functional code.	Focus on finalizing the game design and implementation in Semester 2, ensuring smooth gameplay and integration with other system components.
Ashley Vequiso	**Game & Web Developer** - builds and maintains both the Roblox game client and the React.js teacher dashboard, ensuring seamless integration with the Spring Boot backend.	Finalize both the game and web application in Semester 2, ensuring seamless integration and optimal performance across platforms.

TEAM MEMBER	ROLE & RESPONSIBILITIES	
	SEM 1	**SEM 2**
Jhudiel Artezuela	**Game Des`igner** - focuses exclusively on creating and refining the game's graphical user interface, ensuring it is intuitive, visually appealing, and enhances player experience.	Focus on finalizing the game design and implementation in Semester 2, ensuring smooth gameplay and integration with other system components.
John Kenneth Baguio	**Game Designer** - focuses exclusively on creating and refining the game's graphical user interface, ensuring it is intuitive, visually appealing, and enhances player experience.	Focus on finalizing the game design and implementation in Semester 2, ensuring smooth gameplay and integration with other system components.

[Table 4.3.1] Role and Responsibilities

5. Managerial process plans

This section outlines the managerial processes used to plan, manage, and execute the **CyberKids** project. It includes details about the project's time and resource estimation, team organization, training, work activities, and scheduling strategies. By implementing structured processes, the project team ensured that tasks were efficiently allocated, milestones were met, and any challenges were promptly addressed.

5.1. Start-up plan

5.1.1. Estimation plan

In order to manage time and resources effectively, the team adopted a practical method for estimating the effort required for each task. This involved reviewing previous experiences, breaking down tasks into manageable components, and agreeing on time estimates through collaborative discussion. The process also included regular review and updates of time estimates based on the progress made.

Estimation Process:

1. **Task Identification:**
   The team listed all major tasks involved in the project, such as game design, development, testing, and documentation.
2. **Initial Time Estimates:**
   Each team member estimated the time it would take to complete their assigned tasks. These estimates were based on prior experience with similar tasks, the perceived complexity of the task, and known challenges.
3. **Group Discussion and Final Estimates:**
   After individual estimates were provided, the team held a meeting to discuss and align the estimates. Any discrepancies between estimates were addressed through collaborative discussion to reach an agreement on final time estimates for each task.

4. **Shared Tracking Tool:**

   A shared spreadsheet was used to track all task time estimates and progress. The spreadsheet was updated regularly to reflect changes in estimates, the completion status of tasks, and adjustments to deadlines.

5. **Time Estimate Reviews:**

   During weekly team meetings, time estimates were reviewed, and updates were made as necessary. If tasks were taking longer or shorter than initially expected, new time estimates were discussed and adjusted accordingly.

Time Estimates and Milestones:

- **Weekly Time Allocation:**

  Team members committed **12–48 hours per week** depending on their school load and project deadlines.

- **Project Milestones:**

  - **Alpha Version Completion:** March
  - **Beta Version Completion:** April
  - **Final Version Completion:** May

- **Testing and Bug Fixing:**

  Time was allocated for post-development tasks such as testing, bug fixing, and implementing minor improvements. This was an ongoing process that ran alongside the core development stages.

5.1.2. Staffing plan

The CyberKids project team is made up of five student developers from CIT-University. Each member contributed to planning, designing, developing, testing, and documenting the project. The team worked together across all phases of the project but focused on specific skill areas.

Team Structure and Staffing Timeline

- During the academic semester, members worked part-time, balancing schoolwork and project duties.
- During semester breaks, the team committed to full-time hours to meet key deadlines.
- Weekly team meetings were held to plan, check progress, and reassign tasks if needed.

Skill Requirements

Each team member was selected based on their familiarity or willingness to learn the following skills:

- Game Development using **Roblox Studio and Lua scripting**
- Backend programming using **Spring Boot (Java)**
- Web development using **React.js** for building dashboards
- Basic **UI/UX design** principles
- **Testing and debugging** skills
- **Documentation and writing skills** for formal project reports

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

Sources of Staff

- All team members were selected from within the same academic program.
- No external hiring was done.
- Mentorship and feedback were provided by a faculty adviser.

Flexibility and Reassignment

- If a task became delayed or a team member was unavailable, others would assist or take over.
- Tasks were assigned weekly based on availability, interest, and progress in previous tasks.

Staff Retention

- The same core team was retained from start to finish to maintain consistency and team coordination.
- No staff turnover occurred during the course of the project.

5.1.3. Resource acquisition plan

This section explains all the tools, equipment, and services we needed to complete the CyberKids project. It also shows how we got those resources, how we used them, and who was responsible for managing them.

Hardware Resources

Each team member used a personal laptop as their main development device. To make sure everyone could run Roblox Studio, code, and test efficiently, the laptops had to meet minimum technical requirements.

Minimum Laptop Requirements:

- Processor: Intel Core i3 (8th Gen or better)
- RAM: 4 GB minimum (8 GB recommended)
- Storage: At least 2 GB of free disk space
- Operating System: Windows 10 or higher
- Graphics: Integrated GPU (no dedicated graphics card required)
- Internet Access: Required for online tools, meetings, and database connection

The team ensured that all laptops were functional and compatible with the development tools before starting the project.

Software Tools and Platforms

To develop the CyberKids system, the team used free or shared software tools, except for the database, which required a paid plan.

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
Azure Database	Online database for storing scores, progress	Microsoft	Free

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

Database Resource (Paid)

For data storage, we chose Azure Database for MySQL because of its stability, availability, and ability to scale for future use. It was used to:

- Store player scores and progress
- Manage teacher dashboard data
- Securely connect the backend system with the front end

To use Azure, the team created a shared student plan and contributed financially to pay for the cloud database. This was the only paid service we needed.

- Azure Database Plan: Basic Tier
- Cost: Shared by the team (₱320 per member, actual cost depending on usage)
- Access: Managed by the backend developer and team leader
- Security: Protected with passwords, SSL connection, and IP restrictions

Resource Management and Responsibility

- Each member ensured their laptop was ready and updated with the required tools.
- The team leader was responsible for tracking which resources were needed.
- The backend developer handled the Azure database setup, connection, and monitoring.
- All team members used Google Drive to store and access shared documents.

Other Resources

- Internet access was needed by all team members, especially for:
    - Online development (e.g., Azure database)
    - Remote collaboration via Google Meet or Messenger
    - Research, tutorials, and documentation access
- The school's computer lab was available for group meetings and performance testing if needed, but most development was done remotely.

5.1.4. Project staff training plan

Before working on CyberKids, the team needed to learn how to use certain tools and technologies. Some members already had experience, while others learned during the project.

Training Goals:

- Make sure everyone understands the tools used in development
- Improve teamwork by sharing knowledge and resources
- Prepare members to work on specific parts like the game, backend, dashboard, or testing

Topic	Purpose	Training Method
Roblox Studio + Lua	Build the game modules and user interaction	YouTube tutorials, sample practice
React.js	Build the teacher dashboard (UI)	Industry Elective course at CIT-U
Spring Boot (Java)	Create backend APIs and database access	Application Development course at CIT-U

Git & GitHub	Save code, collaborate safely	Hands-on practice, mentor sessions
Azure Database	Connect backend to online storage	Online documentation

Training Schedule:

- Initial training: 1–2 weeks during the planning phase
- Ongoing learning: Weekly short sessions or shared resources
- Peer support: Experienced members helped others throughout development

The training helped the team become confident and efficient in building the system.

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 across two partial semesters (March to May), with overlapping school activities, and is expected to be finalized by early May 2025. Resource planning was done in accordance with the academic calendar, availability of team members, and institutional support. All work was completed by a single development team composed of five undergraduate students who shared tasks across multiple roles.

Hardware resources were limited to personal laptops with baseline technical specifications suitable for game development and backend programming. No additional computer laboratory access was required, though it remained available for occasional group sessions.

Software tools and platforms were selected based on accessibility, relevance to the project goals, and affordability. All core development tools—Roblox Studio, IntelliJ IDEA Community Edition, React.js, and GitHub—were freely available. The only paid service used

was the Azure Database for MySQL, for which the team jointly covered the cost under a basic student-tier plan.

The project relied entirely on online collaboration, using GitHub, and virtual communication platforms. No physical materials or external personnel were involved. Time allocation and resource use were tightly controlled through shared timelines and weekly check-ins, ensuring that tasks were delivered according to milestone deadlines.

We managed our schedule through completion of major development artifacts:

- The Software Requirements Specification (SRS) was completed by late March 2025.
- The Software Design Document (SDD) was finalized by the end of March.
- All game development tasks were completed using Roblox Studio from March 25 to April 20.
- Backend integration, dashboard creation, and testing occurred throughout April.
- Final polishing and documentation were finished by May 3.

By using personal laptops, free tools, and organized weekly work cycles, the team successfully allocated and managed resources without requiring outside support.

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

As a student capstone project, *CyberKids* does not have a formal budget. Most tools and platforms used are free or open-source. However, the team is currently saving up for one essential expense:

Work Activity	Item/Resource	Estimated Cost	Remarks
Backend/API Hosting	Azure Database Flexible Server	₱1,250	Required for stable and reliable database access during development and testing

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** outlines the steps the team will follow to conclude the CyberKids project, ensuring that all deliverables are properly submitted and that the team reflects on the development process.

Submission of Deliverables

At the end of the project, the team will:

- Submit all project deliverables to the shared Google Drive and GitHub repository. These deliverables will include:
    - Source code
    - Documentation (SRS, GDD, SPMP, etc.)
    - A simple maintenance guide to assist future teams or users in maintaining the application.

Final Presentation

- The final version of the CyberKids application will be presented during the Capstone finals day. This presentation will showcase the completed game and its educational features, allowing stakeholders to review the final product.

Archiving and Documentation

- After the final presentation, all project files will be archived for future reference. The documentation will be finalized and stored securely, ensuring it’s available for potential future maintenance or project continuation.

Debrief Meeting

- The team will hold a short debrief meeting to reflect on the overall project experience. During this meeting, the team will discuss:

    - What went well during the project.
    - What could be improved in future projects.
    - Key lessons learned throughout the development process.

6. Technical process plans

This section of the **Software Project Management Plan (SPMP)** outlines the technical approach and processes the team will follow throughout the development of the CyberKids project. It includes the chosen development process model, technical methods, tools, and techniques used to create the work products, as well as the infrastructure and product acceptance plan.

6.1 Process Model

General Iterative Approach

The CyberKids project adopts an iterative and incremental development process, combining the Rational Unified Process (RUP) for engineering tasks and a tailored agile-inspired methodology for project management. Each module will go through a consistent cycle of requirements elicitation, high-level and detailed design, implementation, and verification, ensuring frequent evaluations and feedback incorporation. The project utilizes Roblox Studio for game-based components and a modern React-based web stack (deployed on Vercel, with backend on Render and database on Azure MySQL) for the teacher dashboard.

Each iteration will contain the following mini-phases:

- **Detail Design**
- **DLD Review**
- **DLD Inspection**
- **Code**
- **Code Review**
- **Compile**
- **Code Inspection**
- **Unit Test**

1. Game Mission Module

- **Requirements Phase**

In this phase, the team will identify critical cybersecurity concepts that form the foundation of the game missions, including responsible data sharing, password creation, and phishing awareness. The requirements will be validated through consultation with Teacher Rosario to ensure age-appropriateness and curriculum alignment. The scope includes three main levels: Data Leak Investigation, Password Fortress Defense, and Cyber Escape Room.

- **High-Level Design Phase**

This phase will involve crafting the overall architecture of the game missions. Each level will be planned out with clear mechanics such as drag-and-drop classification, object collection, and phishing email identification. Roblox Studio will be confirmed as the development environment, and architectural considerations for backend interaction (for progress saving and scoring) will be finalized.

- **Implementation Phase**

The game levels will be implemented in Roblox using Lua scripting. Each level will feature interactive mechanics tailored to the intended cybersecurity lesson. A RESTful API will be integrated to log student progress (time spent, levels completed) into a centralized backend system. Level unlocking will be governed dynamically by stored progress data.

- **Integration and System Test Phase**

Unit and integration testing will be conducted for each level to verify functionality, usability, and flow. The REST API will be tested for accurate data capture and retrieval. Cross-level transitions and persistent storage will be validated to ensure the game provides a seamless educational experience.

- **Delivery Phase**

The game will be deployed as part of the CyberKids system and made accessible to students. Feedback will be gathered from real classroom testing with students under Teacher Rosario's supervision. Necessary adjustments based on engagement and comprehension will be made before the final rollout.

2. Leaderboard and Scoring System

- **Requirements Phase**

The team will define how scores are calculated, categorized, and displayed. Scores will be based on completion time, level difficulty, and badges earned. Requirements for grade-specific leaderboards (Grade 5 and Grade 6) and achievement tracking will be documented

- **High-Level Design Phase**

A scalable design for the leaderboard UI will be created using React. The backend design will include tables to track user progress, timestamps, and badge history. Filtering logic for sorting by grade and achievement type will be incorporated into the plan.

- **Implementation Phase**

The frontend will be developed using React and styled for real-time interactivity. The backend will be implemented using Spring Boot and deployed on Render, with data stored in Azure MySQL. The system will support real-time updates and secure retrieval of top scores and achievement status for each student.

- **Integration and System Test Phase**

The leaderboard will be tested for correctness and responsiveness after game playthroughs. Filtering and sorting logic will be validated against test cases. Stress testing will simulate concurrent student access to ensure stability and accuracy in rankings.

- **Delivery Phase**

Upon successful testing, the leaderboard will be integrated into the student dashboard. Teachers will be shown how to interpret leaderboard data to reward high-performing students. The system will be deployed to Vercel and linked to the game backend for real-time updates.

3. Teacher Dashboard

- **Requirements Phase**

Requirements gathering will be done in collaboration with Teacher Rosario to determine necessary analytics, such as student progress summaries, leaderboard views, and time spent per level. The teacher dashboard is read-only and does not allow direct data manipulation to ensure data integrity.

- **High-Level Design Phase**

The dashboard layout will be sketched using mockups, with features like summary tiles, sortable student performance tables, and filters for grade or achievement type. API endpoints for accessing student statistics will be planned out for efficient data retrieval from the backend.

- **Implementation Phase**

The frontend will be developed using React and deployed via Vercel. The backend services will be built in Spring Boot and hosted on Render, connected to an Azure MySQL database. Teachers will be able to view real-time performance data for all students across all modules.

- **Integration and System Test Phase**

The dashboard will undergo verification using both mock data and live game-generated statistics. Its responsiveness and accuracy will be validated. Teacher account permissions and UI flow will be tested to ensure a smooth user experience.

- **Delivery Phase**

The dashboard will be delivered and demonstrated to Teacher Rosario with accompanying documentation. A teacher guide will be created to help interpret key metrics, encouraging data-driven decision making in the classroom.

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.

8.0 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