**EXECUTIVE SUMMARY**

The CTMS or the Classroom Task Management System is a web-based management system that is designed to streamline task tracking, collaboration and organization in an educational setting. This enables both teacher and students to efficiently manage assignments, deadlines and progress while fostering accountability and productivity.

**PROJECT DELIVERABLES CHECKLIST**

* Process Mapping: [ ] Complete (Refer to Section 11)
* Key Performance Indicators (KPIs): [ ] Complete (Refer to Section 12)
* Service Level Agreement (SLA): [ ] Complete (Refer to Section 13)
* Emerging Technology Integration: [ ] Complete (Refer to Section 14)
* Project Management using Asana: [ ] Complete (Access granted to Ma'am Jen, Sir Villy, Sir Joven, Sir Mark)
* Individual Member Contributions: [ ] Documented (Refer to Chapter 5)

**PROJECT REQUIREMENTS CHECKLIST**

* Web-Based Management System: [ ] Designed and Developed
* Needs of Mock Clients (Ms. Escalada, Mr. Alovera, Mr. Delsocora): [ ] Addressed
* Professionally Presented: [ ] Ensured
* Well-Documented: [ ] Ensured
* Backend Programming Language: Python: [ ] Used
* Database: MySQL: [ / Used
* Core Functions: Create, Read, Update, Delete (CRUD): [ ] Implemented
* Core Functions: User Login & Logout: [ ] Implemented
* Frontend UI and Interactivity: JavaScript: [ ] Used
* Frontend UI Design: Tailwind CSS: [ ] Used

**Table of Contents**

PAGE

[TITLE PAGE i](file:///C:\Users\cmrvn\Downloads\EMPLOYEE%20MANAGEMENT%20SYSTEM%20PAPERWORKS.docx#_Toc167139857)

[EXECUTIVE SUMMARY](file:///C:\Users\cmrvn\Downloads\EMPLOYEE%20MANAGEMENT%20SYSTEM%20PAPERWORKS.docx#_Toc167139858) 1

[PROJECT DELIVERABLES CHECKLIST](file:///C:\Users\cmrvn\Downloads\EMPLOYEE%20MANAGEMENT%20SYSTEM%20PAPERWORKS.docx#_Toc167139858) 2

[PROJECT REQUIREMENTS CHECKLIST](file:///C:\Users\cmrvn\Downloads\EMPLOYEE%20MANAGEMENT%20SYSTEM%20PAPERWORKS.docx#_Toc167139858) 3

[TABLE OF CONTENTS](file:///C:\Users\cmrvn\Downloads\EMPLOYEE%20MANAGEMENT%20SYSTEM%20PAPERWORKS.docx#_Toc167139858) 4

[CHAPTER 1: INTRODUCTION](file:///C:\Users\cmrvn\Downloads\EMPLOYEE%20MANAGEMENT%20SYSTEM%20PAPERWORKS.docx#_Toc167139858) 5

[CHAPTER 2: OVERVIEW OF THE STUDY](file:///C:\Users\cmrvn\Downloads\EMPLOYEE%20MANAGEMENT%20SYSTEM%20PAPERWORKS.docx#_Toc167139858) 6

[CHAPTER 3: SYSTEM REQUIREMENTS](file:///C:\Users\cmrvn\Downloads\EMPLOYEE%20MANAGEMENT%20SYSTEM%20PAPERWORKS.docx#_Toc167139858) 7

[CHAPTER 4: SYSTEM DESIGN & DEVELOPMENT](file:///C:\Users\cmrvn\Downloads\EMPLOYEE%20MANAGEMENT%20SYSTEM%20PAPERWORKS.docx#_Toc167139858) 8

[CHAPTER 5: INDIVIDUAL MEMBER CONTRIBUTIONS](file:///C:\Users\cmrvn\Downloads\EMPLOYEE%20MANAGEMENT%20SYSTEM%20PAPERWORKS.docx#_Toc167139858) 9

[CHAPTER 6: CONCLUSION & RECOMMENDATIONS](file:///C:\Users\cmrvn\Downloads\EMPLOYEE%20MANAGEMENT%20SYSTEM%20PAPERWORKS.docx#_Toc167139858) 10

[REFERENCES / BIBLIOGRAPHY](file:///C:\Users\cmrvn\Downloads\EMPLOYEE%20MANAGEMENT%20SYSTEM%20PAPERWORKS.docx#_Toc167139858) 11

[APPENDICE](file:///C:\Users\cmrvn\Downloads\EMPLOYEE%20MANAGEMENT%20SYSTEM%20PAPERWORKS.docx#_Toc167139858) 12

**CHAPTER I**

**INTRODUCTION**

**8.1 Project Context**

Living in an evolving educational landscape, classrooms are no longer confined to the physical

realm. The shift toward digital and blended learning environments has undeniably increased the

complexity of managing academic task and interactions. The Classroom Task Management System arises in this context as a tool to help both teachers and students when navigating these challenges.

As institutions integrate technology into everyday learning, educators face an array of responsibilities, including task assignment, deadline tracking, and real-time performance monitoring. With traditional methods often falling short, a CMTS provides a centralized platform that consolidates these tasks into a streamlined, user-friendly dashboard. This digital system not only automates checking of task but also promotes efficiency by reducing forgetful tasks and enhance productivity into student performance.

**8.2 Problem Statement**

During a busy week at the university, teachers like Ms. Jeanny Escalada, Mr. Joven Alovera, and Mr. Villy Delsocora face significant challenges in manually checking assignments, projects, and tasks for numerous students per class. This traditional, labor-intensive approach results in time-consuming, less productive, and inefficient work routines, one that can quickly become exhausting over time. Moreover, such an unsustainable workload may adversely affect the teachers' mental health, potentially leading to stress, depression, burnout, and even prompting resignations. However, by using a Classroom Task Management System, these problems may be effectively addressed. The CTMS automates routine tasks such as tracking submissions and provides real-time progress updates.

**8.3 Importance of the Project**

A Classroom Task Management System project is tremendously important because it doesn’t only address the problem faced by Ms. Escalada, Mr. Alovera and Mr. Dolsocora but also addresses many challenges faced by traditional classrooms. By automating routine tasks, such as tracking submissions and providing real-time progress updates, the system alleviates the burdens placed on educators who traditionally spend countless hours manually checking assignments and projects for large classes. This automation not only streamlines workflow and significantly boosts productivity but also help students to have an efficient and effective organizing and submission of task by allowing seamless data exchange with the use of Internet of Things or IoT. Ultimately, a CTMS project transforms an inefficient, labor-intensive process into an organized and sustainable management system, enabling educators to devote more time to interactive teaching and enhancing the overall educational experience while safeguarding their well-being.

**8.4 Project Objectives**

CTMS aims to achieve and provide a more efficient, more productive and more effective management of classroom tasks and addressing operational challenges for both the teachers and students.

* To allow seamless data exchange and online submission of activities with the help of IoT.
* To enhance accountability and progress tracking in allowing both teachers and students to monitor the status of assignments.
* To provide a unified platform where all assignments, projects, and announcements are stored, ensuring that students and teachers can easily locate and review content at any time.
* To present precise timelines for each task, reducing ambiguity and helping students manage their time effectively.

**8.5 Significance of the Study**

**Mock Clients**

Instead of having a manual checking of classroom tasks personally, our system enhances Ms. Escalada, Mr. Alovera and Mr. Delsocora work efficiency by having a more reliable, productive and effective way in checking the tasks submitted by their students.

**Students/Developers**

For our team, students and developers, this project gave us hands-on experience and by developing our skills in problem-solving, task managing and collaboration skills, database integration, frontend and backend development with the use of Tailwind CSS, JavaScript and Django Framework.

**Community/Industry (if applicable)**

The system not only streamlines academic tasks but also promotes skills that are valued beyond the classroom, such as digital literacy, effective time management, and critical problem-solving. These skills are transferable to community projects and volunteer initiatives, cultivating a culture of lifelong learning that benefits both individuals and the community at large.

**8.6 Scope and Limitations**

**Scope**

The study focuses on designing and implementing a web-based classroom task management system that allows teachers to assign tasks, track student progress, and generate performance reports. The system also allows students to experience a more organized tasks list along with deadlines. Students may also pass their tasks and assignments online.

The system is equipped with features such as a calendar view for deadlines, and an analytic dashboard to visualize task completion trends. The technologies used include HTML, CSS, Python with Django, and MySQL, allowing for a reliable, measurable and organized platform that fills the holes in a traditional classroom task operation. Transforming manual checking and submission of paper-based tasks into a digitalize and automated method.

**Limitations**

Classroom Task Management System acknowledges constraints such as relying on internet dependency, automated notification alert for new task and deadlines, automatic grade recorder, display setting where user may choose either light and dark mode

**CHAPTER II**

**OVERVIEW OF THE STUDY**

**9.1 System Description**

The **Classroom Task Management System (CTMS)** is a web-based platform is developed to improve how tasks and assignments are managed in an educational setting. It is designed to help both teachers and students stay organized, productive, and connected.

In many classrooms, managing multiple assignments, deadlines, and student progress using manual methods can be time-consuming and error-prone. The System addresses this problem by providing a centralized platform where teachers can create, assign, and monitor tasks in real-time. Students can view their assignments, submit work, and track their progress easily.

* **Dashboard Module:** Show upcoming tasks, recent activity, and task summary and display academic performance indicators.
* **Student Tasks Module:** View assigned tasks, submit work and track progress.
* **Authentication Module:** Student account login/logout and Password management and user session control
* **Notification Module:** Real-time reminders for upcoming deadlines and System messages or task status updates
* **Progress Tracking Module:** Visual tracking of completed vs. pending tasks and Graphs or charts to show student progress over time.

This system helps students manage academic tasks by providing a centralized platform to view assignments, track deadlines, monitor progress, and receive feedback or rating from teachers. It streamlines task management with real-time notifications, automatic updates, and reminders to support productivity and improve academic performance.

**9.2 Core Functionalities**

The Classroom Task Management System provides the following essential features and capabilities focused on helping students effectively manage their academic tasks and responsibilities:

* **User Authentication and Authorization:** Secure login system with role-based access control for different user types (Student, and Teacher). The authentication mechanism ensures secure access using password hashing, session management, and logout functionality. Permissions are assigned based on the user’s role within the system.
* **Student Task Management:** Core functionality allowing students to add, view, update, and delete academic tasks or assignments. Each task can include a title, description, subject, due date, priority level, and status (e.g., pending, in progress, completed). The task interface promotes an organized, checklist-style workflow for students.
* **Task Scheduling and Calendar Integration:** Integrated calendar view that maps out all tasks by due date, helping students visualize their workload and plan accordingly. Tasks are color-coded based on priority or subject and can be dragged and rescheduled in the calendar interface.
* **Progress Tracking and Performance Monitoring:** The system tracks task completion rates, overdue assignments, and progress over time. A progress dashboard shows visual indicators of student task completion, helping students monitor their academic performance.
* **Notification System:** Real-time notifications and reminders alert students about upcoming deadlines, overdue tasks, and newly added assignments. Notifications are delivered through in-app pop-ups and optional email alerts.
* **Subject and Class Management:** Admins or teachers can create and manage academic subjects or classes, which are then associated with specific tasks. This helps categorize assignments and enables subject-specific tracking and filtering.
* **Dashboard Overview:** Student dashboard displays an overview of tasks provided by the instructor.
* **Analytic and Reporting:** Built-in analytics dashboard for students and teachers, displaying visual reports such as task submission rates, on-time completion statistics, most active students, and common overdue patterns. These insights support academic planning and timely intervention.
* **Document Attachment and Submission:** Students can upload files (e.g., homework, project reports) as part of task submissions. Each task retains a record of attached documents and submission timestamps for teacher reference.
* **Key Performance Indicators (KPIs):** Automated tracking of academic engagement and productivity metrics such as Task Completion Rate, On-Time Submission Rate, and Average Task Response Time. These KPIs help measure student performance and system efficiency

**CHAPTER III**

**System Requirements**

**10.1 Functional Requirements**

**10.1.1 System Functional Requirements**

**FR-01** The system shall require user authentication via username and password before accessing any functionality.

**FR-02** The system shall enforce account-based access control for Teacher/Admin and student, restricting access to authorized functionalities of the system.

**FR-03** The system shall support self-registration and password recovery for non-administrative users.

**FR-04** The system shall display a calendar that shows assignment deadlines and scheduled tasks.

**10.1.2 Teacher’s Account Requirements**

**FR-05** The system shall allow teachers to create, modify, and delete assignments and tasks.

**10.1.3 Student’s Account Requirements**

**FR-06** The system shall allow students to view and submit assignments/tasks.

**FR-07** The system shall allow students to

**FR-07** The system shall display a calendar that shows assignment deadlines and scheduled tasks.

**10.2 Non-Functional Requirements**

|  |  |
| --- | --- |
| **Category** | **Requirement Description** |
| Performance | The system shall load the dashboard within 3 seconds. |
| Performance | The system shall load at least 50 concurrent users. |
| Security | **The system shall perform account-based access control.** |
| Security | The system shall protect against CSRF, XSS, and SQL injection. |
| Reliability | The system shall back up all data daily. |
| Reliability | The system shall maintain 95-98% uptime. |
| Usability | The system shall be available 95-98% of the time during the academic year |
| Stability | The system shall operate without frequent errors during typical usage. |
| Portability | The system shall be accessible and functional across common, modern web browsers. |
| Maintainability | The system shall use a modular, well-documented architecture with robust version control and logging to enable efficient updates and fixes. |

**10.3 Hardware Requirements**

**10.3.1 Teacher Hardware Requirements**

|  |  |
| --- | --- |
| **Component** | **Minimum Specification** |
| **Processor** | Multi-Core CPU(4+cores) |
| **Memory (RAM)** | 8 GB DDR4 RAM Minimum |
| **Storage** | 100GB SSD |
| **Display** | 1920x1080 resolution or higher (21.5” monitor recommended) |
| **Browser** | Google Chrome, Microsoft Edge |
| **Internet** | 50 Mbps Stable Connection |

**10.3.2 Student Hardware Requirements**

|  |  |
| --- | --- |
| **Component** | **Minimum Specification** |
| **Processor** | Multi-Core CPU(4+cores) |
| **Memory (RAM)** | 8 GB DDR4 RAM Minimum |
| **Storage** | At least 500 MB free (for caching, file downloads, etc.) |
| **Display** | 1024x768 resolution or higher |
| **Browser** | Google Chrome, Microsoft Edge |
| **Internet** | 50 Mbps Stable Connection with 2 Mbps download speed |

**10.4 Software Requirements**

|  |  |  |
| --- | --- | --- |
| **Category** | **Component** | **Version** |
| Server Environment | Operating System | Linux Ubuntu 22.04 LTS / Windows Server 2019+ / macOS 12+ |
| Server Environment | Web Server | Nginx 1.18+ / Apache 2.4+ |
| Server Environment | WSGI Server | Gunicorn 20.1.0+ / uWSGI 2.0.19+ |
| Server Environment | Database | MySQL 8.0+ / MariaDB 10.6+ |
| Server Environment | Programming Language | Python 3.10+ |
| Server Environment | Framework | Django 5.2.1 |
| Server Environment | Python Libraries | mysqlclient 2.2.0, Pillow 10.1.0, python-dotenv 1.0.0 |
| Server Environment | Node.js | v20.0+ |
| Server Environment | NPM Packages | Tailwind CSS 3.4.17, Autoprefixer 10.4.21, PostCSS 8.5.3 |
| Client Environment | Web Browsers | Chrome 96+, Edge 96+, Safari 15+ |
| Client Environment | Operating System | Windows 10+, macOS 12+, modern Linux distro |
| Development Environment | IDE | VS Code with Python/JS extensions |
| Development Environment | Version Control | Git 2.35+ |
| Development Environment | Database Tool | MySQL CLI / MySQL Workbench |
| Development Environment | Virtual Environment | Python venv or virtualenv |
| Project Management Tool | Project Management Tool | Asana |

**CHAPTER IV**

**SYSTEM DESIGN & DEVELOPMENT**

**11.1 Back-end Development**

**11.1.1 Programming Language: Python with Django Framework**

The Classroom Task Management System uses **Python** with the **Django framework** for its

backend due to Python’s readability and Django’s robust features like ORM, built-in admin, and

strong security. Key libraries include mysqlclient for MySQL database connectivity, Pillow for

image processing, and python-dotenv for managing environment variables. This stack ensures

the system is secure, scalable, and easy to maintain.

**11.1.2 Database: MySQL**

The **Classroom Task Management System** uses **MySQL** to manage and store structured data such

as student profiles, academic tasks, deadlines, and progress tracking. Django’s ORM simplifies

database interactions by allowing developers to work with Python objects instead of writing raw

SQL. The database is designed to support core features like **user accounts**, **task creation**, **status**

**updates**, and **submission tracking**.

**11.1.3 Core Functions Implemented: Create, Read, Update, Delete (CRUD)**

The system implements full **CRUD operations** (Create, Read, Update, Delete) for managing key

data like tasks and user accounts. These functions allow administrators and users to create new

records, view and update existing data, and remove outdated entries. Django views handle these

operations by interacting with models and responding through defined routes in urls.py, ensuring

organized and efficient data management.

1. **1.4 Core Functions Implemented: User Login & Logout**

The **Classroom Task Management System** uses **Django’s built-in authentication system** to

securely manage student and admin accounts. It includes essential features such as **user**

**registration**, **login/logout**, and **secure password hashing** to protect user credentials. Django’s

**session management** ensures that users remain logged in as they navigate through the system,

providing a smooth and personalized experience.

To maintain proper control over system access, **role-based permissions** are implemented,

allowing **students** to manage their own tasks and **administrators** to oversee system-wide data and

user activity. This ensures that each user only has access to the features appropriate to their role.

**11.1.5 Backend Architecture and Design Choices**

The **Classroom Task Management System** is built using Django’s **Model-View-Template (MVT)**

architecture for clear separation of logic, data, and UI. It functions as a **monolithic application**,

ideal for small to medium educational setups. Key parts of the system use **RESTful APIs** for

smooth frontend-backend communication. Django's built-in **security features** protect against

common web threats, ensuring **secure handling of user data** like login credentials and academic

records.

**11.2 Front-end Development**

**11.2.1 UI and Interactivity (JavaScript)**

JavaScript is used to make your system more dynamic and responsive, for example, validating student login forms instantly before submission, filtering the task list in real time as students type keywords, and paginating long lists of assignments without reloading the page. When a student submits a task, JavaScript sends the data asynchronously to the backend and updates the submission history instantly. Teachers can also use JavaScript to dynamically load student lists, edit tasks inline, and rate submissions, all without full page refreshes. This creates a seamless, interactive experience for both students and teachers using clean, efficient vanilla JavaScript.

**11.2.2 UI Design (Tailwind CSS)**

Tailwind CSS is leveraged to craft a clean, responsive, and intuitive interface by applying utility classes directly within HTML, enabling fast development and consistent styling across all components such as the login page, task dashboard, and submission forms. Responsive design principles ensure the layout adapts smoothly to different screen sizes, and the styling approach helps maintain a cohesive visual experience for both students and teachers as they navigate the system.

**11.2.3 Frontend Architecture and Design Choices**

The frontend is structured as a traditional multi-page application (MPA) using Django templates to render each page, such as login, dashboard, task list, and submission views. Rather than relying on a full single-page application (SPA) framework, interactivity is enhanced with JavaScript for dynamic features like real-time task filtering, AJAX-based form submissions, and live updates to submission history. This approach keeps development streamlined while still delivering a responsive, user-friendly experience. It also allows you to fully leverage Django’s templating engine and Tailwind CSS for consistent styling and efficient UI development across both student and teacher interfaces. Let me know if you'd like to visualize this flow or connect it to your backend logic.

**11.3 Development Tools and Technologies**

**11.2 Languages**

* Python (Backend)
* JavaScript (Frontend Interactivity)
* HTML (Frontend Structure)
* SQL (Database Queries)
* CSS (Custom styles and Tailwind CSS framework)

**11.2 Frameworks**

* Django 4.2.8 (Python Backend Framework)
* Vanilla JavaScript (No complex frontend framework - DOM manipulation and AJAX)

**11.2 Databases**

* MySQL

**11.2 Libraries**

* Backend: Django 4.2.8, mysqlclient 2.2.0, Pillow 10.1.0, python-dotenv 1.0.0
* Frontend: Chart.js (for data visualizations), Font Awesome (for icons)
* CSS: Tailwind CSS 3.4.17, Autoprefixer 10.4.21, PostCSS 8.5.3

**11.2 Other Tools**

* Integrated Development Environment (IDE): Visual Studio Code
* Version Control System: Git
* Database Management Tool: MySQL Command Line Interface
* Project Management Tool: Asana
* Build Tools: Node.js and npm (for Tailwind CSS compilation)

**Testing and Quality Assurance**

**Back-End Testing Strategies**

The backend testing approach ensures the reliability and security of core features such as task management, user roles, and data integrity. Unit tests are written using Django’s built-in testing framework to validate model methods and constraints for example, checking that task deadlines cannot be set in the past or that student submissions are linked to valid tasks. Integration tests verify that CRUD operations work across models like Task, Submission, and UserProfile, ensuring that teachers can create, edit, and delete tasks, and students can submit and update their work. API endpoint testing confirms that views return correct HTTP responses, enforce authentication, and handle data correctly—such as verifying that only teachers can access grading endpoints. Authentication and role-based access control are tested to ensure students and teachers have access only to their respective features (e.g., students cannot create tasks). Data validation tests check form inputs, submission deadlines, and grading logic to maintain system integrity. Tools like unittest or pytest can be used to automate and manage these tests efficiently.

**4.4.2 Front-End Testing Strategies**

Frontend testing focused on delivering a consistent, responsive, and intuitive experience for both students and teachers. UI/UX testing was performed manually to verify layout structure, navigation flow, and visual consistency using Tailwind CSS components. Responsiveness testing ensured the interface adapted smoothly across desktops, tablets, and mobile devices. Cross-browser testing was conducted on Chrome, and Edge to confirm consistent behavior and appearance. Form validation testing covered both client-side and server-side checks for login, task submission, and profile updates. JavaScript functionality testing validated dynamic features like real-time task filtering, AJAX-based submissions, and DOM updates without full page reloads. Finally, User Acceptance Testing (UAT) simulated student and teacher workflows—such as submitting tasks, grading, and managing profiles—to confirm usability and feature completeness across all roles.

**4.4.3 Testing Results and Bug Fixes**

Testing and debugging played a crucial role in enhancing system reliability and user experience. Early issues with foreign key constraints during task or user deletions were resolved by refining cascade behavior and adding confirmation prompts to prevent accidental data loss. Client-side form validation was improved to give students and teachers immediate feedback—reducing unnecessary server load and improving usability. Authentication flows were streamlined with better session handling and redirect logic, ensuring smooth transitions after login or logout. Visual inconsistencies were addressed by standardizing Tailwind CSS usage, improving error displays, and ensuring full responsiveness across devices. Performance was also optimized using Django’s select\_related and prefetch\_related to minimize database queries and speed up page loads. These refinements collectively strengthened the system’s robustness and made it more intuitive for both students and teachers to manage tasks, submissions, and profiles efficiently.

4.5 Process Mapping

4.5.1 System’s Core Process Visualization

Classroom Task Management System, the core processes can be clearly mapped and documented through detailed process flow diagrams to ensure clarity, maintainability, and scalability. These include:

1. User Authentication Process – Begins at the landing page, where students and teachers log in. Based on their role, they are redirected to their respective dashboards with session management in place.
2. Student Workflow – After login, students access their dashboard to view assigned tasks, filter them using a search bar, and paginate through the list. They can view task details, submit work via a form, and track their submission history. They also have access to profile management features like editing their profile, changing their password, and logging out.
3. Teacher Workflow – Teachers log in to access a dashboard where they can create, edit, and delete tasks. They manage a student list, add or update student records, and grade submitted tasks. Like students, they can also manage their profile and password, and log out securely.
4. Task Management Process – Covers the full lifecycle of a task: creation by teachers, assignment to students, student submission, and teacher grading. Each step includes validation, status updates, and data persistence.
5. Submission & Grading Process – Students submit tasks through a form that supports file uploads and deadline checks. Teachers review submissions, assign grades or feedback, and the system logs this history for future reference.
6. Dashboard Overview Process – Aggregates real-time data such as pending tasks, upcoming deadlines, recent submissions, and grading summaries, tailored to each user role.

These processes can be visualized in flowcharts and documented in files like:

* documentation/process\_flow\_visualization.html – for visual diagrams
* documentation/technical\_process\_mapping.md – for detailed logic and data flow

4.5.2 Inputs, Outputs, activities, and Decision Points

Each main process can be structured with clearly defined components to ensure clarity, traceability, and seamless integration across the platform: Task Management Process Inputs: Teacher-created task details, deadlines, and assigned students Activities: Task creation, editing, deletion, and assignment Decision Points: Validate deadline logic, task uniqueness, and assignment scope Outputs: Task records, assignment logs, and task visibility on student dashboards; Submission Management Process Inputs: Student submissions, file uploads, and task references Activities: Submission handling, deadline validation, and file storage Decision Points: Check submission timing, file format, and task linkage Outputs: Submission records, submission history, and grading eligibility; Feedback Process Inputs: Submitted tasks, and teacher feedback Activities: Review submissions, and provide comments Decision Points: Validate, feedback completeness Outputs, feedback logs, and performance summaries; User & Role Management Process Inputs: Registration data, role assignments (student/teacher), and profile updates Activities: Account creation, profile editing, password changes Decision Points: Role-based access validation, data completeness Outputs: User profiles, authentication tokens, and access logs; Dashboard & Analytics Process Inputs: Task data, submission records, grading history Activities: Data aggregation, deadline tracking, and performance visualization Decision Points: Determine overdue tasks, identify ungraded submissions Outputs: Real-time dashboards, alerts, and user-specific insights.

4.6 Key Performance Indicators (KPIs)

Classroom Task Management System, three core KPIs are used to assess its effectiveness: task completion to monitor student engagement and timely submissions, feedback turnaround time and instructional responsiveness, and system performance metrics to track uptime and page responsiveness ensuring the platform remains academically supportive, technically stable, and user-focused.

Defined KPIs

**CHAPTER V**

**INDIVIDUAL MEMBER CONTRIBUTIONS**

**Member 1: Clores, Carl Marvin Nabor**

**Assigned Role(s):** Project Manager, Researcher  
**Backend Contributions:**

* n/a

**Frontend Contributions:**

* Checking of functional designs
* Assisted in designing the website’s user interface by selecting appropriate color schemes, typography, and layout structures to enhance usability and visual appeal.

**Other Contributions:**

* Project planning and task delegation using Asana.
* Drafting, organizing and finalizing the project.
* Wrote most parts of the project paper.
* Helped compile user guide screenshots and visual documentation.

**Member 2: Daria, Carl Vincent Delfin**

**Assigned Role(s):** Researcher  
**Backend Contributions:**

**Frontend Contributions:**

**Other Contributions:**

**Member 3: Dela Cruz, Mc Jersey Arlante**

**Assigned Role(s):** UI/UX Designer   
**Backend Contributions:**

**Frontend Contributions:**

* Assisted in designing the website’s user interface by selecting appropriate color schemes, typography, and layout structures to enhance usability and visual appeal.
* Helped the developers in adjusting our layout.
* Making of the Logo.

**Other Contributions:**

* Wrote some parts of the project paper (overview, functional requirements, non-functional requirements, hardware and software requirements)
* Participated in refining of the project paper, ensuring it reflected the technical and research aspects of the system.

**Member 4: Lawag, John LesternVillasoto**

**Assigned Role(s): Backend Developer, System Analyst, Tester**  
**Backend Contributions:**

**Frontend Contributions:**

**Other Contributions:**

**Member 5: Simpas, Joshua Hurtada**

**Assigned Role(s): Frontend Developer, Quality Analyst (Tester)**  
**Backend Contributions:**

**Frontend Contributions:**

**Other Contributions:**

**CHAPTER VI**

**Conclusion and Recommendation**

**Project Outcome**

CMTS or the Classroom Task Management System successfully addressed the crucial operational challenges faced by our mock clients by delivering an automated solution

**Future Recommendations**