

**ASSIGNMENT 1**

**SUBJECT: CS438.**

**STUDENT NAME: MOHAMED ALNAJEH**

**STUDENT ID: 2190105529**

**STUDENT NAME: AHMED ALNAJEH**

**STUDENT ID: 2200110086**

**1.Problem description:**

The Collaboration and Management Tool (CMT) is designed to address the challenges faced by student teams in the Computer Science Department at the University of Tripoli when managing real-world projects. These projects, sponsored by local and regional companies, require students to follow a structured process involving tasks like requirements gathering, software design, and creating work products such as test plans, weekly reports, and project schedules. However, the current lack of a centralized system for managing these tasks and work products leads to inefficiencies, miscommunication, and difficulty in tracking progress. The CMT aims to provide a unified platform for teams to collaborate, manage tasks, share files, and track project progress effectively.

**1.1 What are we trying to do:**

The goal of the CMT is to create a **processing back-end** for collaboration and management tools that will help student teams:

* **Organize Projects:** Create, update, and manage projects with ease.
* **Track Tasks:** Assign tasks, set deadlines, and monitor progress.
* **Collaborate:** Share files and work together efficiently.
* **Generate Reports:** Automatically generate progress reports for projects and tasks.
* **Ensure Accountability:** Track individual contributions and ensure everyone is on the same page

**1.2 Current Practices and Limitations:**

Today, many university teams utilize a combination of email, spreadsheets, and various cloud services to manage their projects. This approach has several limitations:

* **Fragmentation**: Teams must switch between different applications, leading to lost information and time.
* **Lack of Integration**: Tools do not communicate with each other, making it difficult to get a comprehensive view of project status.
* **Communication Barriers**: Relying on email can lead to missed messages and unclear communication.
* **Inefficient Tracking**: Manually updating task statuses in spreadsheets can be tedious and error-prone, resulting in outdated information.

**2. Functional requirements:**

Here are the 12 functional requirements for the **CMT** project implementation.

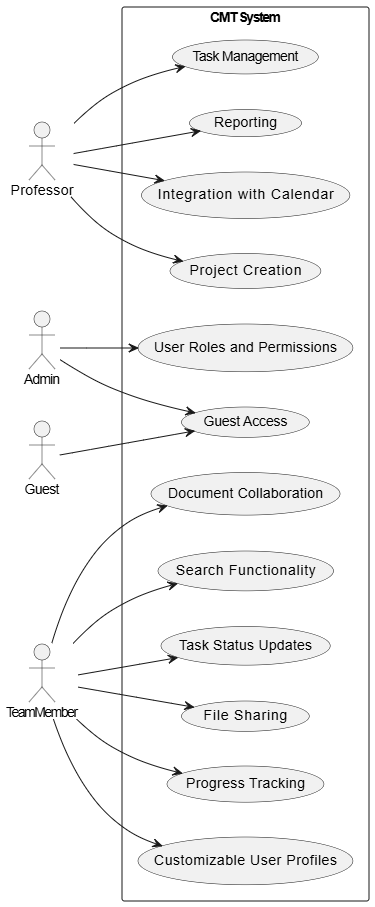
1. **Project Creation**:
   1. **Description:** Teams can create new projects with a title and description. Professors are responsible for setting overall deadlines for these projects**.**
   2. **Responsible Role**: **Team Members (**for creation**), Professors (**for setting deadlines**).**
2. **• Task Management**:
   1. **Description**: **Professors** can create, edit, and delete tasks within a project. **Team Members** can update the status of tasks assigned to them but cannot create or delete tasks.
   2. **Responsible Role**: **Professors** (for creating and managing tasks), **Team Members** (for updating assigned tasks)
3. **Task Status Updates**:
   1. **Description**: Team members can easily mark tasks assigned to them as complete or in progress to indicate their current status.
   2. **Responsible Role**: **Team Members**
4. **File Sharing:**
   1. **Description**: Users can upload, download, and manage files related to their projects and tasks assigned to them by professors.
   2. **Responsible Role: Team Members (**with permissions defined by **Professors)**
5. **Progress Tracking:**
   1. **Description**: The system allows team members to update the status of tasks, showing progress in a visual format (like a progress bar).
   2. **Responsible Role: Team Members (**for updating status**), Professors (**for oversight**).**
6. **Reporting**:
   1. **Description**: Users can generate reports on project progress, including completed tasks, pending tasks, and overall project status.
   2. **Responsible Role: Professors (**with input from **Team Members)**
7. **• User Roles and Permissions:**
   1. **Description**: Different user roles (Admin, Professors, Team Member, Guest) are defined, with specific permissions for each role.
   2. **Responsible Role: Admin**
8. **Integration with Calendar:**
   1. **Description**: The tool integrates with external calendar services to manage deadlines and meetings.
   2. **Responsible Role: Professors (with user-configurable settings)**
9. **Document Collaboration**
   1. **Description**: Team members can edit and share documents together .
   2. **Responsible Role: Team Members**
10. **Search Functionality:**
    1. **Description**: Users can search for projects, tasks, and files quickly using keywords.
    2. **Responsible Role: All Users**
11. **Customizable User Profiles:**
    1. **Description**: Allow users to personalize their profiles and settings.
    2. **Responsible Role: All users**
12. **Guest Access:**
    1. **Description:** The system allows guest users to view projects and tasks without needing an account.
    2. **Responsible Role: Admin (**to set permissions**)**

**3. Stretch goals:**

Here are some stretch goals that we are hoping to implement:

* **• Messaging Platform:**
  + Implement a real-time messaging feature for team members to communicate about projects and tasks, enhancing collaboration.
* **• Task Prioritization:**
  + Allow users to prioritize tasks within projects, helping teams focus on the most critical work first.
* **• Basic Notifications:**
  + Provide essential notifications for task updates and project changes to keep team members informed.
* **• Task Checklists:**
  + Introduce simple checklists within tasks to help users track progress on subtasks easily.
* **• Mobile App:**
  + Develop a mobile application that allows users to access the platform on the go, keeping them connected and productive anytime, anywhere

**4. Use Case diagram of CMT:**

****

**5. Use Case description of CMT:**

### 1. ****Use Case: Project Creation****

* **Actors**: Professor
* **Description**: Professors can create new projects by providing a title and description. They are also responsible for setting the overall deadlines for these projects.
* **Preconditions**:
  + The professor is logged into the system.
  + The professor has the necessary permissions to create a project.
* **Postconditions**:
  + A new project is created and saved in the system.
  + The project is visible to the assigned team members.
* **Basic Flow**:
  1. The professor selects the option to create a new project.
  2. The system displays a form for entering the project title and description.
  3. The professor fills in the project title and description.
  4. The professor sets the project deadline.
  5. The professor submits the form.
  6. The system saves the project and notifies the assigned team members.
* **Exceptions**:
  1. **Missing Required Fields**: If the professor leaves any required field blank, the system displays an error message and prompts the professor to fill in the missing information.
  2. **Invalid Deadline**: If the professor sets a deadline in the past, the system displays an error message and asks for a valid date.

### 2. ****Use Case: Task Management****

* **Actors**: Professor
* **Description**: Professors can create, edit, and delete tasks within a project. Team members can update the status of tasks assigned to them but cannot create or delete tasks.
* **Preconditions**:
  + A project has been created.
  + The professor is logged into the system.
* **Postconditions**:
  + Tasks are created, edited, or deleted as specified.
  + Team members can view and update the status of their assigned tasks.
* **Basic Flow**:
  1. The professor selects a project.
  2. The professor chooses to add a new task, edit an existing task, or delete a task.
  3. For adding a task:
     + The professor fills in the task details (title, description, deadline).
     + The professor assigns the task to one or more team members.
     + The system saves the task.
  4. For editing a task:
     + The professor modifies the task details.
     + The system updates the task.
  5. For deleting a task:
     + The professor confirms the deletion.
     + The system removes the task from the project.
* **Extensions/Variations**:
  1. The professor can assign multiple team members to a single task.
* **Exceptions**:
  1. **Task in Progress**: If the professor tries to delete a task that is already in progress, the system displays a warning and asks for confirmation.
  2. **Invalid Deadline**: If the professor sets a task deadline that is earlier than the project deadline, the system displays an error message.

### 3. ****Use Case: Task Status Updates****

* **Actors**: Team Member
* **Description**: Team members can update the status of tasks assigned to them (e.g., mark as complete or in progress).
* **Preconditions**:
  + A task has been assigned to the team member.
  + The team member is logged into the system.
* **Postconditions**:
  + The task status is updated in the system.
  + The updated status is visible to the professor and other team members.
* **Basic Flow**:
  1. The team member selects a task assigned to them.
  2. The team member updates the task status (e.g., marks it as "In Progress" or "Complete").
  3. The system saves the updated status.
* **Extensions/Variations**:
  1. The team member can add comments or notes when updating the task status.
* **Exceptions**:
  1. **Missing Deliverables**: If the team member tries to mark a task as complete without providing required deliverables, the system prompts them to upload the necessary files.
  2. **Unauthorized Update**: If a team member tries to update a task not assigned to them, the system denies the request.

### 4. ****Use Case: File Sharing****

* **Actors**: Team Member
* **Description**: Team members can upload, download, and manage files related to their projects and tasks.
* **Preconditions**:
  + A task or project has been assigned to the team member.
  + The team member is logged into the system.
* **Postconditions**:
  + Files are uploaded or downloaded as specified.
  + The files are accessible to the professor and other team members (based on permissions).
* **Basic Flow**:
  1. The team member selects a task or project.
  2. The team member chooses to upload or download a file.
  3. For uploading:
     + The team member selects a file from their device.
     + The system uploads the file and associates it with the task or project.
  4. For downloading:
     + The team member selects a file from the list of available files.
     + The system downloads the file to their device.
* **Extensions/Variations**:
  1. The team member can organize files into folders.
* **Exceptions**:
  1. **File Size Limit**: If the file size exceeds the limit, the system displays an error message and prompts the team member to upload a smaller file.
  2. **Invalid File Type**: If the file type is not supported, the system displays an error message.

### 5. ****Use Case: Progress Tracking****

* **Actors**: Team Member, Professor
* **Description**: Team members can update the status of tasks, and the system displays progress in a visual format (e.g., progress bar). Professors can oversee the progress.
* **Preconditions**:
  + Tasks have been assigned to team members.
  + The team member or professor is logged into the system.
* **Postconditions**:
  + The progress of tasks is updated and displayed visually.
* **Basic Flow**:
  1. The team member updates the status of a task (e.g., marks it as complete).
  2. The system updates the progress bar or visual indicator for the project.
  3. The professor views the progress report.
* **Extensions/Variations**:
  1. The system can send automated progress notifications to the professor.
* **Exceptions**:
  1. **No Updates**: If no tasks have been updated, the system displays a message indicating no progress changes.

### 6. ****Use Case: Reporting****

* **Actors**: Professor
* **Description**: Professors can generate reports on project progress, including completed tasks, pending tasks, and overall project status.
* **Preconditions**:
  + A project has been created, and tasks have been assigned.
  + The professor is logged into the system.
* **Postconditions**:
  + A report is generated and available for download or viewing.
* **Basic Flow**:
  1. The professor selects a project.
  2. The professor chooses to generate a report.
  3. The system compiles data on completed tasks, pending tasks, and overall progress.
  4. The system displays the report in a readable format (e.g., PDF or table).
* **Extensions/Variations**:
  1. The professor can customize the report format (e.g., include only completed tasks).
* **Exceptions**:
  1. **No Data**: If no tasks have been completed, the report indicates that no progress has been made.

### 7. ****Use Case: User Roles and Permissions****

* **Actors**: Admin
* **Description**: The admin defines and manages user roles (Admin, Professor, Team Member, Guest) and their permissions.
* **Preconditions**:
  + The admin is logged into the system.
* **Postconditions**:
  + User roles and permissions are updated in the system.
* **Basic Flow**:
  1. The admin selects the option to manage roles and permissions.
  2. The system displays a list of existing roles and permissions.
  3. The admin adds, edits, or deletes roles and permissions.
  4. The system saves the changes.
* **Extensions/Variations**:
  1. The admin can assign specific permissions to individual users.
* **Exceptions**:
  1. **Role in Use**: If the admin tries to delete a role that is currently assigned to users, the system displays a warning and asks for confirmation.

### 8. ****Use Case: Integration with Calendar****

* **Actors**: Professor
* **Description**: The system integrates with external calendar services to manage deadlines and meetings.
* **Preconditions**:
  + The professor is logged into the system.
  + The professor has configured their external calendar.
* **Postconditions**:
  + Deadlines and meetings are synced with the external calendar.
* **Basic Flow**:
  1. The professor sets a deadline or schedules a meeting in the system.
  2. The system syncs the deadline or meeting with the external calendar.
  3. The professor receives a notification in their external calendar.
* **Extensions/Variations**:
  1. The system can send reminders for upcoming deadlines or meetings.
* **Exceptions**:
  1. **Calendar Not Configured**: If the external calendar is not configured, the system prompts the professor to set it up.

### 9. ****Use Case: Document Collaboration****

* **Actors**: Team Member
* **Description**: Team members can collaboratively edit and share documents within the system.
* **Preconditions**:
  + A task or project has been assigned to the team member.
  + The team member is logged into the system.
* **Postconditions**:
  + Documents are edited and shared among team members.
* **Basic Flow**:
  1. The team member selects a document to edit.
  2. The system opens the document in an editable format.
  3. The team member makes changes and saves the document.
  4. The system updates the document and notifies other team members.
* **Extensions/Variations**:
  1. Multiple team members can collaborate on the same document in real-time.
* **Exceptions**:
  1. **Document Locked**: If two team members try to edit the document simultaneously, the system locks the document for one user and notifies the other.

### 10. ****Use Case: Search Functionality****

* **Actors**: All Users
* **Description**: Users can search for projects, tasks, and files using keywords.
* **Preconditions**:
  + The user is logged into the system.
* **Postconditions**:
  + Search results are displayed based on the keywords.
* **Basic Flow**:
  1. The user enters a keyword in the search bar.
  2. The system searches for matching projects, tasks, and files.
  3. The system displays the search results.
* **Extensions/Variations**:
  1. The user can filter search results by type (e.g., projects, tasks, files).
* **Exceptions**:
  1. **No Results**: If no results are found, the system displays a "No results found" message.

### 11. ****Use Case: Customizable User Profiles****

* **Actors**: All Users
* **Description**: Users can personalize their profiles and settings.
* **Preconditions**:
  + The user is logged into the system.
* **Postconditions**:
  + The user’s profile and settings are updated.
* **Basic Flow**:
  1. The user selects the option to edit their profile.
  2. The system displays the profile settings.
  3. The user updates their profile information (e.g., name, profile picture, preferences).
  4. The system saves the changes.
* **Extensions/Variations**:
  1. The user can link their profile to external accounts (e.g., Google, LinkedIn).
* **Exceptions**:
  1. **Invalid File**: If the user uploads an invalid file (e.g., unsupported image format), the system displays an error message.

### 12. ****Use Case: Guest Access****

* **Actors**: Guest
* **Description**: Guest users can view projects and tasks without needing an account.
* **Preconditions**:
  + The guest has access to the system (no login required).
* **Postconditions**:
  + The guest can view projects and tasks.
* **Basic Flow**:
  1. The guest navigates to the system.
  2. The system displays a list of available projects and tasks.
  3. The guest selects a project or task to view.
  4. The system displays the details of the selected project or task.
* **Extensions/Variations**:
  1. The guest can request access to restricted content.
* **Exceptions**:
  1. **Restricted Access**: If the guest tries to access restricted content, the system prompts them to log in or request access.

**6. Non-functional Requirements:**

### 6.1. Performance

**Description:** The system must respond to user requests within three seconds for 95% of transactions during peak usage times.

### 6.2. Usability

**Description:** The application should have an intuitive interface that allows users to complete key tasks (like registration and task creation) without requiring more than three steps.

### 6.3. Security

**Description:** User passwords must be stored securely using encryption and must adhere to minimum complexity requirements (e.g., at least eight characters, including upper- and lower-case letters, numbers, and special characters).

**7.** **Risks:**

### 7.1. ****Risk: Calendar Integration Might Not Work Smoothly****

* **What Could Happen**: If the system doesn’t sync properly with external calendars (like Google Calendar or Outlook), deadlines and meetings might not show up correctly, causing confusion.
* **How to Fix It**:
  + Test the calendar integration early to make sure it works with popular services.
  + Have a backup plan, like letting professors manually enter deadlines if the sync fails.

### 7.2. ****Risk: People Might Accidentally Delete Important Tasks****

* **What Could Happen**: If a professor or team member accidentally deletes a task, it could disrupt the entire project.
* **How to Fix It**:
  + Add a confirmation step before deleting anything (e.g., “Are you sure you want to delete this task?”).
  + Keep a log of deleted tasks so they can be restored if needed.

### 7.3. ****Risk: File Uploads Could Crash the System****

* **What Could Happen**: If too many users upload large files at the same time, the system might slow down or crash.
* **How to Fix It**:
  + Set a limit on file sizes (e.g., no files larger than 100MB).
  + Use cloud storage (like Google Drive or AWS) to handle large files instead of storing them on the system itself.

### 7.4. ****Risk: Progress Bars Might Not Update Correctly****

* **What Could Happen**: If the progress bar doesn’t update when tasks are marked as complete, professors and team members might think the project is behind schedule.
* **How to Fix It**:
  + Test the progress tracking feature thoroughly to make sure it updates in real-time.
  + Add a manual refresh button so users can update the progress bar if needed.

**8.** **Team Process description:**

**8.1. Software Toolset**

Here’s the list of tools along with their **justifications**:

|  |  |  |
| --- | --- | --- |
| **Tool** | **Purpose** | **Justification** |
| **XAMPP** | Local server for PHP, MySQL, and Apache. | XAMPP is easy to set up and widely used for PHP-based projects. It provides a local environment for testing. |
| **PHP** | Backend programming language. | PHP is well-suited for web development and integrates seamlessly with HTML and CSS. |
| **HTML/CSS** | Frontend development for creating the user interface. | HTML and CSS are standard for building responsive and visually appealing web pages. |
| **MySQL** | Database management system. | MySQL is lightweight, easy to use, and integrates well with PHP for storing project and task data. |
| **Git/GitHub** | Version control and collaboration. | Git allows team members to work on different features simultaneously without conflicts. GitHub provides a centralized repository for collaboration. |
| **Visual Studio Code** | Code editor for writing and debugging code. | VS Code is lightweight, supports PHP, HTML, and CSS, and has extensions for debugging and version control. |

**8.2. Build System:**

I have chosen the following tools to streamline my workflow:

|  |  |  |
| --- | --- | --- |
| **Tool** | **Purpose** | **Justification** |
| **Composer** | Dependency management for PHP. | Composer helps manage PHP libraries and ensures all team members use the same versions of dependencies. |
| **NPM (Optional)** | Package manager for frontend dependencies (e.g., Bootstrap, jQuery). | If you use JavaScript libraries, NPM can help manage them. |
| **XAMPP** | Local server for running and testing the application. | XAMPP provides a complete environment for PHP, MySQL, and Apache, making it easy to test the application locally. |

### ****3. Test-Automation Infrastructure****

For testing your PHP-based application, you can use the following tools:

|  |  |  |
| --- | --- | --- |
| **Tool** | **Purpose** | **Justification** |
| **PHPUnit** | Unit testing framework for PHP. | PHPUnit is the most popular testing framework for PHP. It allows you to write and run unit tests for your backend logic. |
| **Selenium** | Automated browser testing for frontend. | Selenium can automate testing of your HTML/CSS frontend to ensure it works across different browsers. |

### ****4. Weekly Schedule with Measurable Milestones****

Here’s a **4-week schedule** with **measurable milestones** for your team. Adjust the tasks based on your team size and roles.

#### **Week 1: Project Setup and Basic Structure**

* **Milestone**: Set up XAMPP, Git repository, and project structure.
* **Tasks**:
  + Install XAMPP and set up the local server.
  + Create a GitHub repository and invite team members.
  + Set up the basic folder structure (e.g., css/, js/, php/, assets/).
  + Create a basic index.php file with a navigation bar (using Bootstrap).
  + Write a simple PHP script to connect to the MySQL database.

#### **Week 2: User Authentication and Project Creation**

* **Milestone**: Implement user authentication and project creation functionality.
* **Tasks**:
  + Create a MySQL database for users and projects.
  + Implement user registration and login (PHP + MySQL).
  + Create a form for professors to create new projects (title, description, deadline).
  + Store project data in the database.
  + Display a list of projects on the dashboard.

#### **Week 3: Task Management and File Sharing**

* **Milestone**: Implement task management and file sharing features.
* **Tasks**:
  + Create a MySQL table for tasks.
  + Allow professors to create, edit, and delete tasks.
  + Allow team members to update task status (e.g., In Progress, Complete).
  + Implement file upload and download functionality (store files in a folder or database).
  + Display uploaded files for each task.

#### **Week 5: Progress Tracking and Reporting**

* **Milestone**: Implement progress tracking and reporting features.
* **Tasks**:
  + Add a progress bar to show the completion status of tasks in a project.
  + Allow professors to generate a report showing completed and pending tasks.
  + Test the entire system (PHPUnit for backend, Selenium for frontend).
  + Fix any bugs and prepare for final submission.

**References:**

1. **System analysis book. : Naima Elbadri.**
2. **Michigan Tecnological university.**
3. **Public information from : deepseek**
4. **Canvas platform as an example for the CMT undergoing project.**