Name & Title: - Archival Material Management System

Introduction :- A Graphic Design Project Management System is a software tool designed to help graphic design teams manage their projects from start to finish. The system typically includes features such as task management, project timelines, resource allocation, and communication tools to help teams collaborate and stay on track. By using a Graphic Design Project Management System, designers and project managers can streamline their workflow, ensure timely delivery of projects, and maintain quality standards. This ultimately helps design teams to be more productive and efficient in their work.

Enums Used:

Status_t:

Enum Name	Description
NEW	Indicates that a task or project is in a new state.
IN_PROGRESS	Indicates that a task or project is currently in progress.
REVIEW	Indicates that a task or project is being reviewed or evaluated.
DONE	Indicates that a task or project is completed or finished.

Update_Status_t:

Enum Name	Description
SUCCESS	The update operation was successful.
NOT_FOUND	The item to be updated was not found.
INVALID_STATUS	The status provided for the update operation is invalid.

Project_type_t:

Enum Name	Description
LOGO	Represents a project type for a logo design.
BROCHURE	Represents a project type for a brochure design.
WEBSITE	Represents a project type for a website design.
PACKAGING	Represents a project type for a packaging design.

Structures Used:

Client_t:

Property	Data Type	Description
name	char[50]	The name of the client. This is a string of up to 50 characters.
email	char[50]	The email address of the client. This is a string of up to 50 characters.

Project_t:

Property	Data Type	Description
name	char[50]	A string representing the name of the project.
duration	int	An integer representing the duration of the project in days.
type	project_type_t	A custom data type representing the type of the project (e.g., software, hardware, marketing, etc.).
status	status_t	A custom data type representing the status of the project (e.g., in progress, completed, cancelled, etc.).
client	client_t	A custom data type representing the client associated with the project.

Cost_t:

Property	Data Type	Description
hours	int	Represents the number of hours worked.
rate	float	Represents the hourly rate of the work.
cost_t	struct	A data type that combines the properties of hours and rate to represent the cost of work.

Problem Description:

The following functions will be used for the assignment:

Function Name	Parameters	Return Type	Explanation
add_project	project_t projects[], int *num_projects, char name[], int duration, project_type_t type, status_t status, char client_name[], char client_email[]	int	This function adds a new project to the given array of projects. It takes in the project array, the number of projects, the name, duration, type, status, client name, and client email of the new project. If a project with the same name already exists in the array, it returns 0, otherwise it adds the new project to the array and increments the number of projects by 1, and returns 1.
find_project_by_name	project_t projects[], int num_projects, char name[]	int	This function searches for a project in the given array of projects by name. It takes in the project array, the number of projects, and the name of the project to search for. If the project is found, it returns the index of the project in the array, otherwise it returns -1.
update_project_status	project_t projects[], int num_projects, char name[], status_t new_status	update_status_t	This function updates the status of a project in the given array of projects. It takes in the project array, the number of projects, the name of the project to update, and the new status of the project. If the project is not found in the array, it returns NOT_FOUND. If the new status is the same as the current status, it returns SUCCESS. If the new status is not valid for the current status of the project, it returns INVALID_STATUS. Otherwise, it updates the status of the project and returns SUCCESS.
update_project_duration	project_t projects[], int num_projects, char name[], int new_duration	int	This function updates the duration of a project in the given array of projects. It takes in the project array, the number of projects, the name of the project to update, and the new duration of the project. If the project is not found in the array, it returns 0. Otherwise, it updates the duration of the project, and if the project is in progress or review and the new duration is less than the current duration, it sets the project status to review. It

			returns 1 to indicate success.
calculate_total_project_cost	project_t projects[], int num_projects	cost_t	This function calculates the total cost of all projects in the given array of projects. It takes in the project array and the number of projects. It calculates the total hours and rate based on the duration and type of each project, and returns the total cost as a cost_t structure.
delete_project_by_name	project_t projects[], int *num_projects, char name[]	int	This function deletes a project from the given array of projects by name. It takes in the project array, a pointer to the number of projects, and the name of the project to delete. If the project is not found in the array, it returns 0. Otherwise, it shifts all projects after the one to be deleted down by one position, decrements the number of projects by 1, clears the memory used by the deleted project, and returns 1 to indicate success.

Test Cases :

Test case name	Input	Output	Explanation
testAddProjectUniqueName	project_t projects[10] = {0}; int num_projects = 0; char name[] = "Project 1"; int duration = 5; project_type_t type = LOGO; status_t status = NEW; char client_name[] = "Client 1"; char client_email[] = "client1@example.com";	1	Test that a project can be added to the project array when the name is unique.
testAddProjectDuplicateName	project_t projects[10] = {0}; int num_projects = 0; char name[] = "Project 1"; int duration = 5; project_type_t type = LOGO; status_t status = NEW; char client_name[] = "Client 1"; char client_email[] = "client1@example.com";	0	Test that a project with a duplicate name cannot be added to the project array.
testAddProjectMaxProjects	project_t projects[5] = {0}; int num_projects = 9; char name[] = "Project 6"; int duration = 5; project_type_t type = LOGO; status_t status = NEW; char client_name[] = "Client 1"; char client_email[] = "client1@example.com";	1	Test that a project can be added to the project array when the maximum number of

			projects is not exceeded.
testFindProjectNotFound	int num_projects = 3; char name[] = "Project 1"; project_t projects[num_projects] = {...};	-1	Test that the function returns -1 when the project name is not found in the project array.
testFindProjectFound	int num_projects = 3; char name[] = "Project 1"; project_t projects[num_projects] = {...};	0	Test that the function returns the index of the project when the project name is found in the project array.
testFindProjectByName_Found	project_t projects[10] = {...}; int num_projects = 3; char name[] = "Project 2";	1	Test that the function returns the index of the project when the project name is found in the project array.
testUpdateProjectStatusNotFound	project_t projects[2] = {...}; int num_projects = 2; char name[] = "Project 3"; update_status_t result = update_project_status(projects, num_projects, name, IN_PROGRESS);	NOT_FOUND	Test that the function returns NOT_FOUND when the project name is not found in the project array.
testUpdateProjectStatusNoChange	project_t projects[2] = {...}; int num_projects = 2; char name1[] = "Project 1"; update_status_t result = update_project_status(projects, num_projects, name1, NEW);	SUCCESS	Test that the function returns SUCCESS when the project status is updated to the same status.
testUpdateProjectStatusInvalidStatus	project_t projects[2] = {...}; int num_projects = 2; char name[] = "Project 1"; update_status_t result = update_project_status(projects, num_projects, name, INVALID_STATUS);	INVALID_STATUS	Test that the function returns INVALID_STATUS when the status value is not valid.
testUpdateProjectDurationNotInList	projects array with 2 projects, num_projects is 2, name is "Project 3", and duration is 30	result is 0, projects[0].duration is 10, projects[1].duration is 20	Testing the update_project_duration() function when trying to update the duration of a project that is not in the projects array. The function should

			return 0 to indicate that the project was not found and the duration should remain unchanged for all projects.
testUpdateProjectDurationSuccess	projects array with 1 project, num_projects is 1, name is "Project 1", and duration is 20	result is 0, projects[0].duration is 20	Testing the update_project_duration() function when successfully updating the duration of a project in the projects array. The function should return 0 to indicate success and the duration of the project should be updated.
testUpdateProjectDurationNegativeDuration	projects array with 2 projects, num_projects is 2, name is "Project 1", and duration is -5	result is 1, projects[0].duration is -5	Testing the update_project_duration() function when trying to update the duration of a project with a negative value. The function should return 1 to indicate that the duration is invalid and the duration should be updated to the negative value.
testCalculateTotalProjectCostSingleBrochure	projects array with 1 project of type BROCHURE and 4 hours	expected_cost is {120, 6000.0}, actual_cost is {120, 6000.0}	Testing the calculate_total_project_cost() function when calculating the cost of a single BROCHURE project. The expected cost is calculated as hours * BROCHURE_RATE and should match the actual cost returned by the function.

testCalculateTotalProjectCostSingleWebsite	projects array with 1 project of type WEBSITE and 8 hours	expected_cost is {320, 16000.0}, actual_cost is {320, 16000.0}	Testing the calculate_total_project_cost() function when calculating the cost of a single WEBSITE project. The expected cost is calculated as hours * WEBSITE_RATE and should match the actual cost returned by the function.
testCalculateTotalProjectCostMultipleProjects	projects array with 4 projects of different types and durations	expected_cost is {740, 37000.0}, actual_cost is {740, 37000.0}	Testing the calculate_total_project_cost() function when calculating the cost of multiple projects with different types and durations. The expected cost is calculated as the sum of the costs of each project and should match the actual cost returned by the function.
testDeleteProjectByName_Success	An array of 3 projects with names "Project 1", "Project 2", and "Project 3", and a name of "Project 2"	The number of projects should decrease by 1, and the function should return 1. The deleted project should not be found in the array, and the other projects should remain in the correct order.	This test case verifies that the delete_project_by_name() function can successfully delete a project from an array of projects by name.
testDeleteProjectByName_NotFound	An array of 3 projects with names "Project 1", "Project 2", and "Project 3", and a name of "Project 4"	The number of projects should not change, and the function should return 0. The array should remain unchanged.	This test case verifies that the delete_project_by_name() function handles the case where the project to be deleted is not found in the array.

testDeleteProjectByName_Successful	An array of 3 projects with names "Project 1", "Project 2", and "Project 3", and a name of "Project 2"	The function should return 1, and the number of projects should decrease by 1. The remaining projects should be in the correct order.	This test case verifies that the delete_project_by_name() function can successfully delete a project from an array of projects by name.