PROJECT MANAGEMENT SYSTEM

Author: Soroush Ariana
Software Engineering 2 Coursework
Module Leader: Dr. David Adama

MARCH 2024
Nottingham Trent University

Table of Contents

General Description

This is a project management console app coded in C# using object oriented techniques. It provides a tool for managing and monitoring various projects, tasks within these projects, and different members with different roles. Simplifying communication, increasing transparency, and offering precise real-time insights into project progress are the objectives. The solution supports the expanding team and varied project portfolio, which will ultimately result in enhanced productivity and more informed decision-making. It reduces the risk of inaccuracies and miscommunications that can arise from manual tracking methods. The system facilitates better collaboration among team members, especially in a distributed environment, by providing a platform for sharing updates, assigning tasks, and tracking progress transparently.

There are 3 different roles defined in the system. Administrator, team leader, and team member. Each user can do some specified tasks due to their role. Users can interact with the system via the console.

Everyone can log in and out the system via the same process, see the status of the projects, see progression messages of a project, and add a comment on a project. Of course for the last 2 cases, team members and team leaders are only able to do them if their name is already added to the list of the members working on that specific project.

In addition to the above, team members and team leaders can also announce that the task they are assigned to, is done.

In addition to the mentioned items, team leaders can do other affairs, which administrators are also capable of doing. These include creating a project, assigning a task, seeing members working on a project, see tasks of a project (and task assignments), add a member to a project, and add a task to a project.

Besides the affairs that every user can do and the items mentioned in the last paragraph which are mutual between administrators and team leaders, the administrators can do the following as well: creating a team member account, creating a team leader account, creating an administrator account, and mark a project as ‘completed’ and closing it.

It should be noted that to start using the system for the first time, at least the information of one administrator must already exist and be saved in the system. The system asks for the user’s email and password to let them log in. Hence, the one administrator account that already exists in our system is as follow:
email: arianasoroush@gmail.com
password: soroush@123

Cohesion and coupling

Our class design prioritizes high cohesion and low coupling, resulting in a modular and maintainable codebase that is easy to extend and modify as the project evolves. By organizing functionality within classes and minimizing dependencies between modules, I have created a robust foundation for the project management application, ensuring scalability, flexibility, and ease of maintenance.

Cohesion

I have organized the functionality within each class around specific responsibilities to achieve high cohesion. For example, the Project class encapsulates methods and properties related to managing projects, such as adding tasks, adding members to the project, and updating project status. Similarly, the Task class focuses on tasks' attributes and functionalities, such as setting status and assigning deadlines. This ensures that each class has a clear purpose and avoids unnecessary complexity.

In this writer's opinion, communicational cohesion is achieved. For instance, methods within each class perform similar tasks or operate on the same data to produce consistent outcomes, promoting readability and maintainability.

Coupling

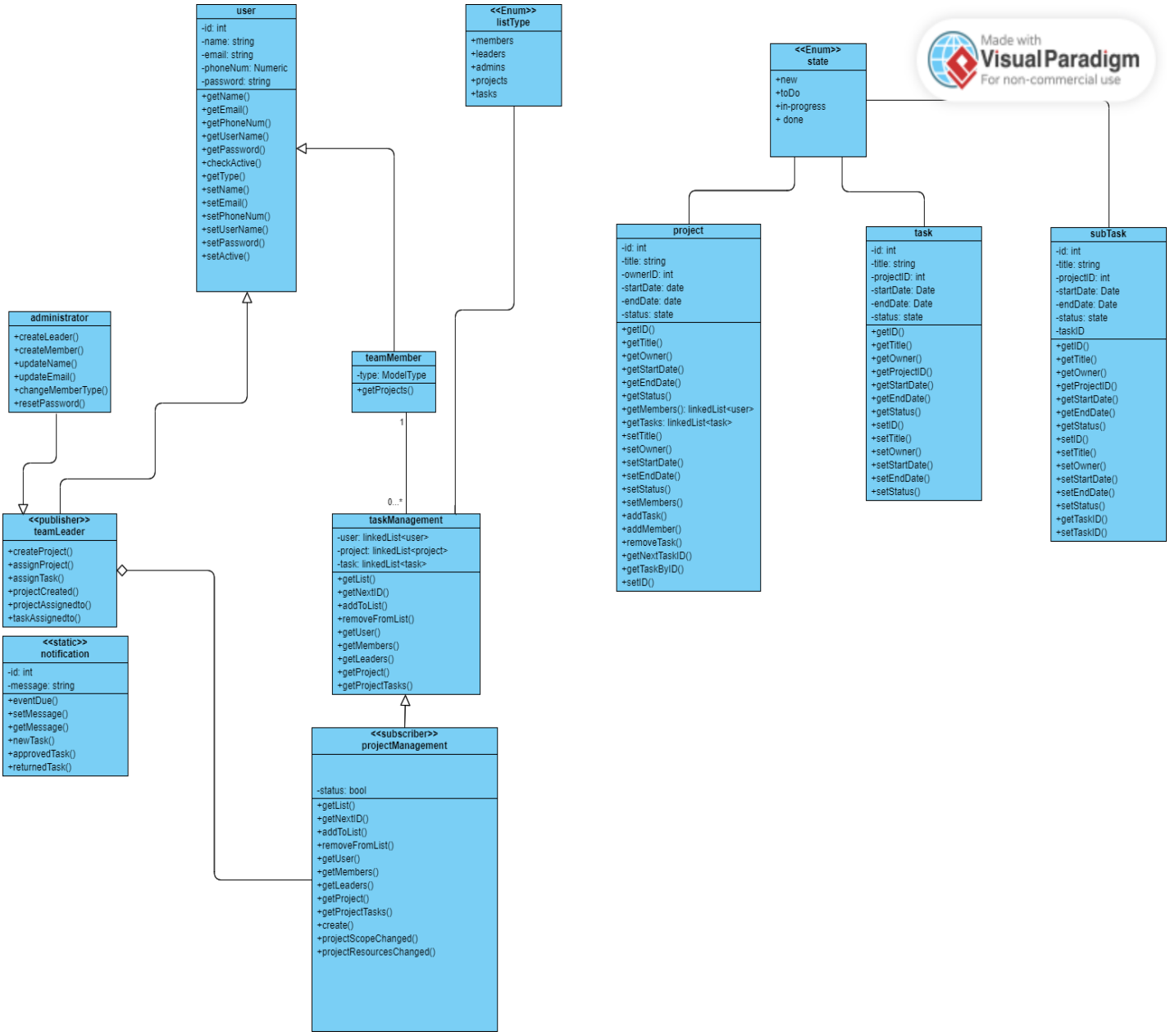
In our project management C# console app, I have employed the Observer design pattern to minimize coupling between classes and promote flexibility in our design.

The Observer pattern facilitates communication between objects in a loosely coupled manner. In our implementation, classes some classes act as subjects, while other classes act as observers. This design allows observers to subscribe to and receive notifications about changes in the state of subjects without being tightly coupled to them.

By adopting the Observer pattern, I have decoupled the classes involved in managing projects, tasks, and users. Subjects are responsible for managing their state and notifying observers of any changes, while observers react to these changes without needing to be aware of the internal details of the subjects. This promotes modularity and extensibility in our codebase.

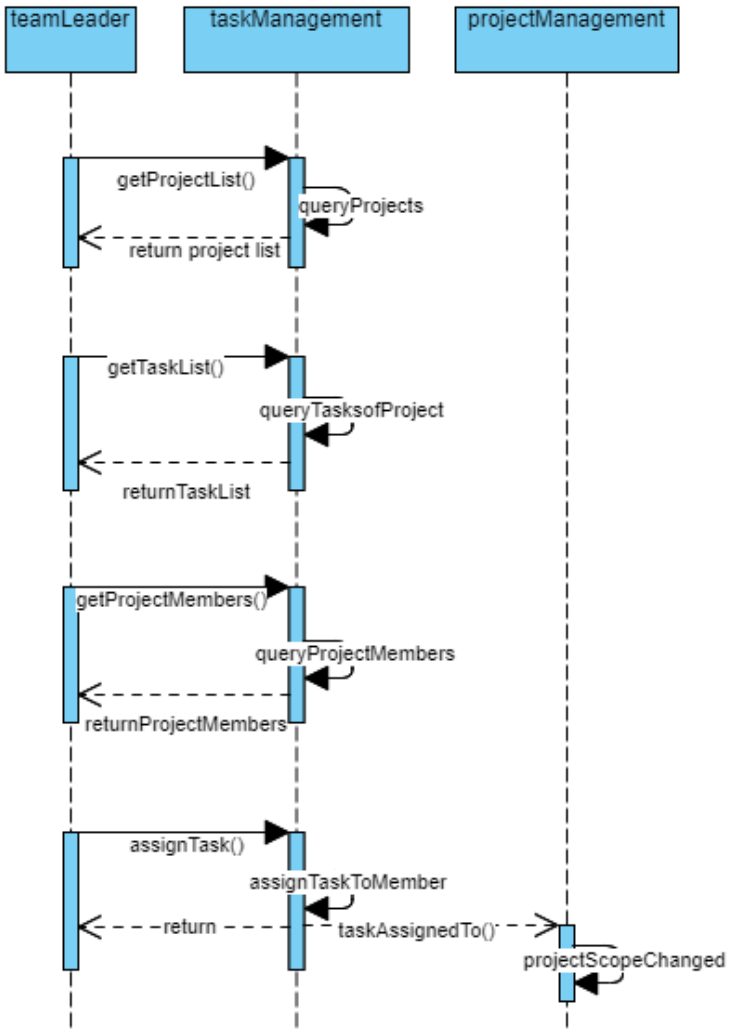
Additionally, the design minimizes external coupling by encapsulating interactions with external entities, such as file systems, within dedicated components. This ensures that changes to external dependencies have minimal impact on the core functionality of our project management system.

Class Diagram



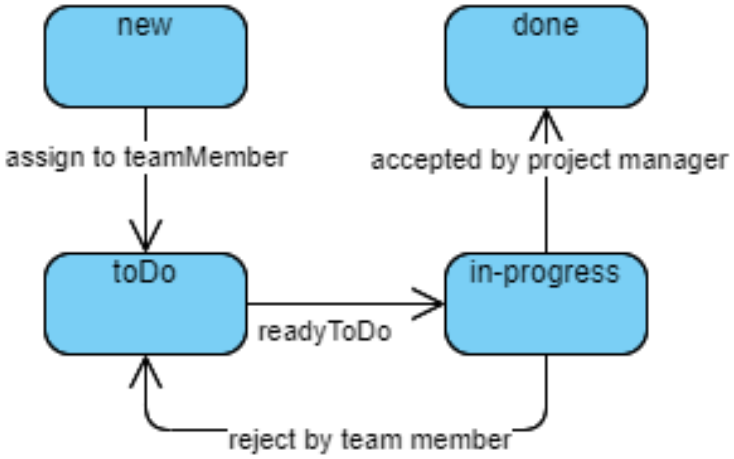
Sequence Diagram

This sequence diagram is for assigning a task by a team leader.

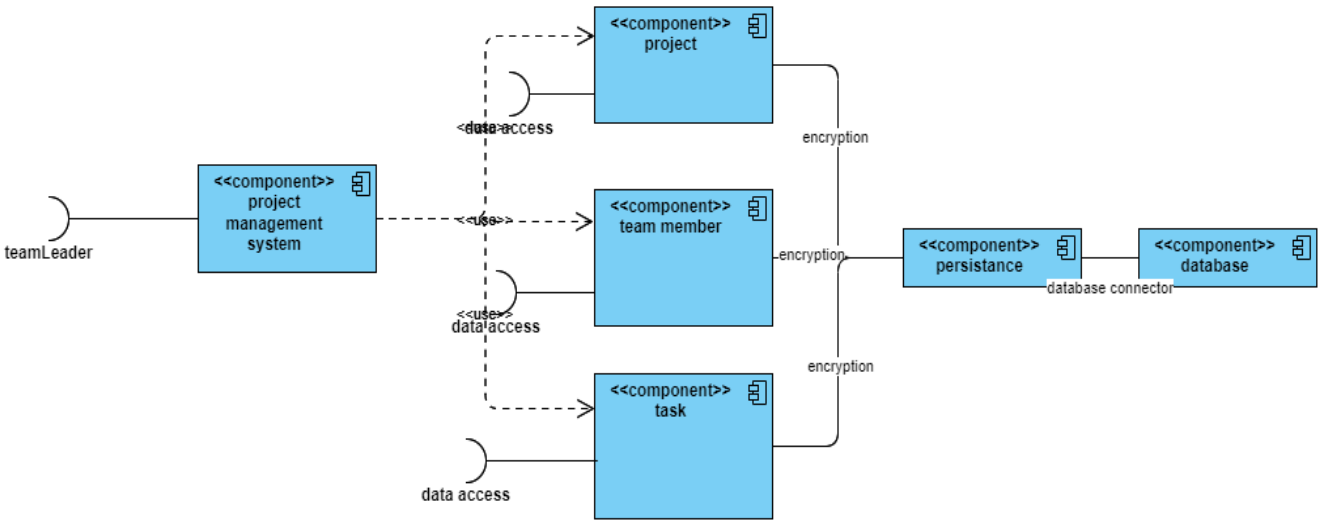


State Diagram

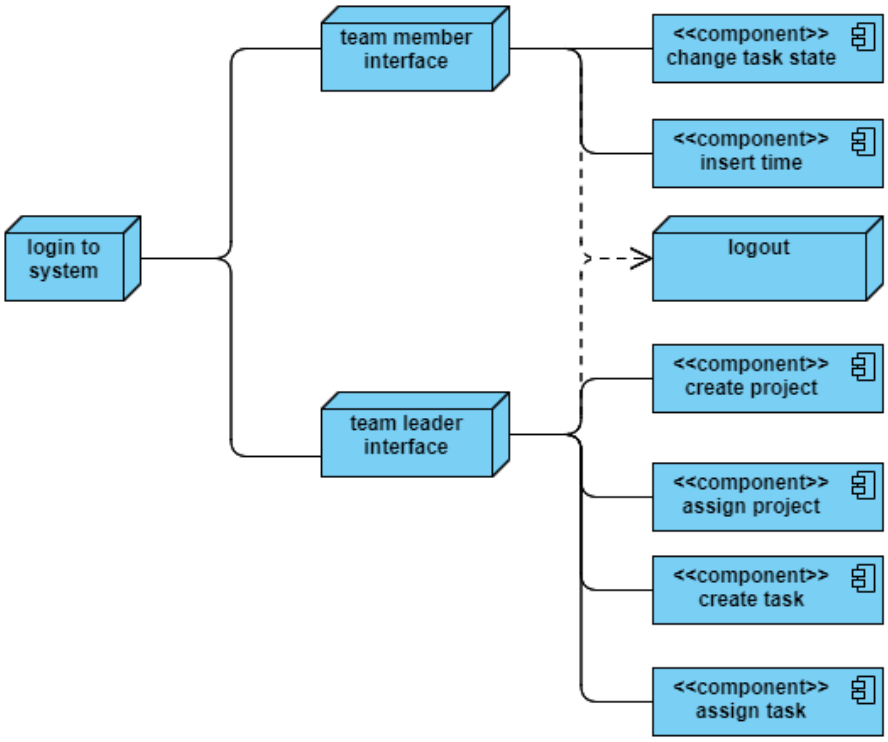
This state diagram is for task management class.



Component Diagram



Deployment Diagram



Design Pattern

Observer Design Pattern:

The Observer pattern enables communication between objects in a loosely coupled manner. It consists of a subject (or observable) and one or more observers:
Subject (Observable): Maintains a list of observers and notifies them of state changes.
Observer: Registers with a subject to receive notifications and reacts to changes independently.
This pattern promotes loose coupling and facilitates maintenance and extensibility.

Event-Driven Architecture (EDA):

EDA emphasizes event generation, detection, and handling within a system:
Event Sources: Entities generate events based on specific conditions.
Event Handlers (Subscribers): Components react to events asynchronously.
Event Bus (Event Broker): Centralized communication channel routes events to handlers.
EDA promotes decoupling and scalability, enabling systems to be more flexible and responsive.

Data Structure and The File Formats

For the implementation of this system, option 1 is chosen. Therefore, the projects and the associated data are stored using containers such as lists and arrays.
On the local hard disk, data is permanently stored in ASCII (text) files. There are multiple classes and different methods. Inside the methods, there are pieces of code for loading data from files and for saving data from memory into the files. The files are CSV (comma-separated values) files.

Algorithms

Create
In the system, users (depending on their role and the situation) can create user accounts, projects, and tasks.

For example, an administrator can create a user member account. When doing this, the administrator should enter the required data for the account, including user name, email, phone number, and password, via the console. Then the program creates a new object of the team member class, passing the entered data to its constructor. Next, in the constructor a new unique user ID is generated for the new team member. Then This information will be written in a CSV format in a text file that is used for storing team member account's information inside the members' directory (folder). However, prior to that, the system checks whether the directory and the text file exist or not and create them if they do not already exist. The same algorithm is used for creating team leader and administrator accounts.

Team leaders and administrators can create projects. When creating a project, the system asks the admin (or the team leader) to enter the project title (name) and checks whether a project with this name already exists or not. If yes, it notifies the user of this problem and asks them to enter another name. If a project with the entered name does not already exist, the system creates the project directory (folder) and moves to the next step. In the next step, the system creates a text file for storing project members' information and another text file for storing project tasks information, in the project directory (folder). Next, the system asks the user to enter project start and end date in the desired

format. If the user enters invalid input or enters the dates in a format other than the desired format, the system notifies them and asks them to re-enter the dates in the desired format. Then the system sets the state of the project (using an enum that indicates different states) as not started. After that, an object of the project class is created, passing the entered data to its constructor and inside the constructor, a new unique project id is assigned to the project. In the final step, a text file named 'description' is created inside the project directory and all the entered information plus the project ID, is written in that text file.

Also the team leaders and administrators can create tasks for projects. When doing this, the user is asked to enter the title of the task. After that, an object of the task class is created, passing the entered title, the project id (as each task belongs to a project, this instantiation takes place in the project class and therefore, we can pass the project ID using 'this.id'), and the state of the task which is initially sets as not started (using an enum that indicates different states). Then inside the constructor of the task class, a new unique task ID is generated for the new task. Then the system checks whether the tasks text file exists in the project directory and creates it if not. In the final step, all the task information is written inside the tasks text file in the format of comma-separated values, in a line.

Update

In the system, some things may get updated; such as the status of the projects and the status of the tasks. In the following lines, the algorithm for updating task status is explained. The algorithms for updating other things such as project status, are similar.

When a task is created, its information is written in a line inside a text file, in the format of CSV, that is each attribute (field) of the task is written and the attributes are separated with commas; and initially the status of the task is written as not started. When the system wants to update the status of a task, it read all lines of the tasks.txt file into an array called lines. Then it Iterates over each line in the array and Splits the line by comma to extract the desired attribute (field), which is the task status in this context. When splitting, it puts each attribute in a cell of another array. Then according to the format of saving task information, we know that the status field is the 4^{th} field in the line. Hence, we change the value of the array's 4^{th} cell (that is the task status) to the new status (in progress or completed) using an enum that indicates different states. Then the system creates a new array, combining the different fields (attributes) of the task and next, it joins the fields with commas between them, in the correct order to form a string (line) and Replaces the current line in the array with the updated line. In the final step, it Writes the modified lines back to the tasks.txt file (actually it overwrites the text file).

Search

In some parts of the program, there is a need to search for the desired user, task, or project in the directories and text files. The search may be based on an ID or a name. since the algorithm for searching for any of the items mentioned is similar, in the following we will explain only the process of searching for a team member account.

In different parts of the program we may need searching as mentioned earlier, for instance, when a user wants to log in the system, the program should search to check if we have such a user in our system. For this purpose, first we want the user to enter their role. This is needed so that we know we should search for the user information in which text file, as we store the information of users in different text files and the name of the text files is the name of the user's role. After the user entered their role, we set the file path (in which we should search for the user's information) due to their role. Next, we ask the user to

enter their email as well as their password. The program then reads all lines of the desired text file into an array called lines. Then it Iterates over each line in the array and Splits the line by comma to extract the desired attributes (fields), which are email and password in this context. When splitting, the program puts each attribute in a cell of another array called parts. Then according to the format of saving user information, we know that the email field is the 2nd and the password is the 4th field in the line. Therefore, we compare the value of these cells to the email and password that the user has entered. In fact, first we search for the entered email in each line that we iterate over, if we do not find the email in any of the lines, we notify the user that the entered email is invalid and give them another chance to enter their email and password. But, if the email is found, then we compare the password field of that line with the entered password. If they do not match, we notify the user that the entered password is not correct and we give them another chance to enter their email and password. However, if they match, we let the user log in the system and notify them that their log in was successful.

Delete

We may want to delete a project from the system. The algorithm for doing this is simple. Briefly explaining, we ask the user (which should be an administrator in this case) to enter the name of the project. Then we check if such project directory (folder) exists in the system. If not, we notify the user that such project does not exist. And if yes, we delete that directory and hence, all the data for that project will be deleted from the system.

It should be noted that we have chosen option 1 of difficulty for the implementation of the system.

User Manual and Instruction

This is console based application; Hence, users can interact with it via the console.

There are 3 different roles defined in the system. Administrator, team leader, and team member. Each user can do some specified tasks due to their role. The overview of the affairs each user can do is mentioned in the 'general description' section of this report. In this section, the procedure of each affair is explained in order to make the users familiar with the system's functionality and how they can interact with it.

Log In and Out The System

When program is run, first the system asks the user to log in. So they should enter their role. 4 numbered options are shown to the user. The options are:

1. Administrator
2. Team Member
3. Team Leader
4. Exit

The user should enter the number of the option they want to select. Obviously if they select option 4, they will completely exit the system. However, if they select other options (specify their role), they are then required to enter their email and password to log in the system. After they are logged in successfully, they can take different actions according to their role. Same as the logging in process, numbered options are shown to the user and the user should enter the number of the option they want to select. The last numbered option for any user is to log out the system. By selecting that option, they simply log out the system and are redirected to the initial page with the mentioned 4 numbered options.

See The Status of a Project
Every user can do this, regardless of their role. To do so, the user just should enter the name of the project.

See Progression Messages for a project
For taking this action, team members and team leaders are only able to do so if their name is already added to the list of the members working on that specific project. However, administrators can do it unconditionally.
Same as seeing the status of a project, the user just should enter the name of the project.

Announcing a task is done
Team members and team leaders can announce that the task they are assigned to, is done.
The user should first enter the name of the project. Then, they need to enter the ID of the task.

Add a Comment on a Project
Same as seeing progression messages for a project, For taking this action either, team members and team leaders are only able to do so if their name is already added to the list of the members working on that specific project. However, administrators can do it unconditionally.
The user should first enter the name of the project. Then, they need to enter their comment (it should be in one line).

Create a project
Administrators and team leaders can create a project.
The user should first enter the name of the project. Then, they should enter the project's start date in the format dd/mm/yyyy. Next, they should enter the project's end date in the same format. After this step, the project will be created.

Add a Task to a Project
Administrators and team leaders can add a task to a project.
The user should first enter the name of the project. Then, they should enter the title of the task.

Add a Member to a Project
Administrators and team leaders can add a Member to a Project.
The user should first enter the name of the project. Next, the user should enter select the role of the member they want to add to the project. Hence, 2 numbered options are shown to the user. The options are:
1. Team member
2. Team leader
The user should enter the number of the option they want to select. Finally, the user should enter the email of the member they want to add to the project.

Assign a Task
Administrators and team leaders can assign a Task.
The user should first enter the name of the project. Then, the user should enter the email of the member to who they want to assign a task. Finally, they should enter the ID of the task (the task should be already added to the project. The members who are assigned to a task will receive an email in which the ID of the task is written. Thus, they normally should have the ID of their task to enter in this stage).

When this affair is done, the member to who the task is assigned, will receive a notifying email, explaining that the task is assigned to them. The details of the task is written in the email.

See Members Working on a Project
Administrators and team leaders can see Members Working on a Project.
The user just should enter the name of the project.

See Tasks of a Project (and Task Assignments)
Administrators and team leaders can see tasks of a project (and task assignments).
The user just should enter the name of the project.

Mark a Project as 'Completed' and Close It
Only administrators can Mark a Project as 'Completed' and Close It.
The user just should enter the name of the project. When this affair is done, the members of the project will receive a notifying email explaining that the project is completed and closed.

Create a Team Member/Team Leader/Administrator Account
Only administrators can do this.
The user needs to enter the name, email, phone number, and password for the account, step by step in order.

Delete a Project
Only administrators can do this.
The user just should enter the name of the project.

Software Test

For this system, 2 test methods are implemented using MSTest framework. The names of the test methods are 'TestMethod1' and 'TestMethod2'. These methods are implemented inside a test class named 'UnitTest1'. They are utilized to test the functionality of the method 'sendEmail' that is inside 'TeamLeader' class. The context in which the function will be called is set up. Then the function is called and after that, we can observe if the function call resulted in the correct behavior using assertion methods. Either of the 2 test methods, test the 'sendEmail' method in different contexts. Actually, there are 2 feasible contexts (scenarios) for the 'sendEmail' method and each of them is tested in a separate test method.

Log of Software Development

```
$ git log
commit f346f46e4aac7d01734177c931448a0eba774e7f (HEAD -> main)
Author: N1237770 <soroush.ariana2023@my.ntu.ac.uk>
Date:   Thu Mar 21 22:13:40 2024 +0000

    back to the last commit

commit 2a8dc41d4bf94d68939a06b88c0ecde6294da381 (origin/main, branch1)
Author: N1237770 <soroush.ariana2023@my.ntu.ac.uk>
Date:   Thu Mar 21 16:10:09 2024 +0000

    The method 'deleteProject' is implemented inside the 'administrator' class. The code is quite finalised.

commit 17a9e277b808c52889d4339308b22491d1d07c16
Author: N1237770 <soroush.ariana2023@my.ntu.ac.uk>
Date:   Mon Mar 18 18:22:35 2024 +0000

    Methods 'seeMembersOfProject' and 'seeTasksOfProject' are implemented inside 'TeamLeader' class.

commit c3d23a743d8536b8089f22dfe1b12be78f5d8879
Author: N1237770 <soroush.ariana2023@my.ntu.ac.uk>
Date:   Mon Mar 18 17:18:17 2024 +0000

    Methods 'showProgressionMessages' and 'showProjectStatus' implemented inside 'User' class.

commit 405199b2f917736447c6c461ae8e9a78fa9475fc
Author: N1237770 <soroush.ariana2023@my.ntu.ac.uk>
Date:   Mon Mar 18 00:40:45 2024 +0000

    Method 'addComment' is implemented inside 'User' class. And some minor changes in 'Administrator' class.

commit bc66c26b22b249f0277ebb661a837a1acfbf1d0a
Author: N1237770 <soroush.ariana2023@my.ntu.ac.uk>
Date:   Sun Mar 17 00:00:30 2024 +0000

    The method 'closeProject' is implemented inside 'Administrator' class.

commit f9a159cac221f61cc2738e28dc1568eff8495a49
Author: N1237770 <soroush.ariana2023@my.ntu.ac.uk>
Date:   Sat Mar 16 18:14:35 2024 +0000

    The method 'taskIsDone' is implemented inside the 'User' class.

commit 53bc00620a17e1510fd0c3ec0c2ac5d4f9144f40
Author: N1237770 <soroush.ariana2023@my.ntu.ac.uk>
Date:   Fri Mar 15 20:06:07 2024 +0000

    The methods 'assignTask' and 'sendEmail' are written inside the 'teamLeader' class.

commit 927959df36dbaa69d96c1f63a1d94daa77c2e8af
Author: N1237770 <soroush.ariana2023@my.ntu.ac.uk>
Date:   Thu Mar 14 20:35:27 2024 +0000

    The method 'addMember' got completed in the class'Project'.

commit f3fb563af68c1e1b1497de56f205c14ff0dccd5a
Author: N1237770 <soroush.ariana2023@my.ntu.ac.uk>
Date:   Wed Mar 13 21:05:26 2024 +0000

    Classes User, TeamMember, TeamLeader, Administrator, Project, Task, IDManager, and Program are created with thie
r private attributes, constructors, setters and getters and some methods. an Enum called CommonEnum is also created.
```