Final Report

For

ToDone

Prepared by

Group Members:

Lama Alabdulkarim	442200432	442200432@student.ksu.edu.sa
Dalal Alsadoun	442201417	442201417@student.ksu.edu.sa
Majd Alqahtani	442202282	442202282@student.ksu.edu.sa

Instructor:	Dr. Sara Alotaibi – Ms. Abeer Alessa
Course:	CSC343 – Systems Analysis and Design
Tutorial Section:	75858
Date:	4 June 2023

Table of Contents

1. Customer Statement of Requirements (CSR)

1.1 Problem Statement

In today's fast-paced world, managing time and tasks is an essential skill that everyone must master. Unfortunately, many people are unable to manage their time efficiently, which can lead to stress, anxiety, and missed deadlines. Due to the many demands on our time, such as work, family, social obligations, and personal goals, it is easy to become overwhelmed and lose track of what needs to be accomplished. Setting priorities is one of the most difficult aspects of managing tasks. Many things are competing for our attention, making it difficult to determine what should be done first. We often procrastinate, delaying important tasks until the last minute, causing unnecessary stress and anxiety. In today's digital age, we receive constant notifications, emails, and social media updates, making it difficult to focus on the task at hand. As a result, many people are constantly checking their phones, resulting in a lack of productivity and a waste of time. It is also possible to miss deadlines if you are not proficient in time management, which can have serious consequences, especially at work. As a result, your reputation may be damaged, and you may lose out on opportunities. Moreover, poor time management can negatively impact your personal life, causing stress and strain on your relationships.

We interviewed a sample of people who live in Saudi Arabia from different age groups. As a result of our interview with them, we were able to determine the problems they face with managing their time. Therefore, we developed our application to solve their problems and provide them with useful tools.

1.1.1 Forgetting Creating Task List

It is common for people to forget to create task lists, which can result in disorganization, forgetfulness, and missed deadlines. The process of keeping track of everything that needs to be accomplished can be challenging without a task list. Consequently, productivity can be lost and stress levels can increase. To address this problem, we might be able to provide a platform for task list creation and management. You may be able to specify the beginning date and deadline, set automated reminders, and integrate calendars into the process. The customer believes that by using such a system they will be able to stay organized and ultimately achieve enhanced success in their professional and personal lives.

1.1.2 Difficulties in Prioritizing Tasks

Task lists may seem simple, but they can be challenging for many people. Sometimes, people have difficulty prioritizing and determining which tasks are most important or urgent. As well as this, some individuals may avoid creating task lists due to a fear of failing. Therefore, we create an application that provides solutions to such problems. It offers an opportunity to improve their productivity and manage their time effectively by prioritizing their responsibilities and ensuring that they are focusing on urgent tasks.

1.1.3 Poor time management

People tend to have poor time management skills, they always delay their tasks to the last minute, resulting in unsatisfactory results and late submissions. In order to solve this problem, we provide a reminder service that sends a notification to the user before deadlines are reached.

1.1.4 Limitation of Categorizing

One of the biggest problems that customers face is the limited number of categories available to categorize their tasks. As a result, it can be difficult for customers to organize their tasks in a meaningful way. Limited categories can also make it difficult to filter and search for specific tasks. This problem may be solved by providing more customizable categories. Customers can then create their own categories or subcategories according to their specific needs. In this way, customers can organize their tasks in the way that makes the most sense to them and best suits their individual priorities.

1.1.5 Difficulties in Dealing with Multi-users

Shared to-do lists and concurrent work on them across different devices and locations have been a challenge for many people, creating conflicts with their work together, as well as difficulties in sharing them. As a solution to this problem, we provide services that support multiple users working on the same list, with each user having their own privileges so that their work cannot conflict.

1.2 Glossary of Terms

TERM	DESCRIPTION
App icons	graphic symbols and computer icons help users quickly and easily identify what they need or want. The use of icons also provides a more appealing visual representation. [1]
QR code	QR stands for "Quick Response.", While they may look simple, QR codes are capable of storing lots of data. But no matter how much they contain when scanned, the QR code should allow the user to access information instantly – hence why it's called a Quick Response code. [2]
Megabyte	A megabyte (MB) is a data measurement unit applied to a digital computer or media storage. One MB equals one million (106 or 1,000,000) bytes. [3]

2. System Requirements

2.1 Enumerated Functional Requirements

REQ-ID	DESCRIPTION	PW
REQ-1	The user shall be able to create a task after logging in.	5
REQ-2	The user shall be able to schedule their tasks after logging in.	5
REQ-3	The user shall be able to write notes in each task after logging in.	5
REQ-4	The user shall be able to customize the settings of their account after logging in.	4
REQ-5	The user shall be able to send their tasks after logging in.	5
REQ-6	The user shall be able to mark a task as completed after logging in.	5
REQ-7	The user shall be able to create a list after logging in.	5
REQ-8	The user shall be able to set deadlines and prioritize their tasks after logging in.	5
REQ-9	The user can create new categories after logging in.	3
REQ-10	The user shall search among lists in one category if required.	5
REQ-11	The system shall have multiple users work together on the same task list which include allowing the user to classify the roles of each member, scan their QR code to access to their tasks and share their QR code with others.	4
REQ-12	The system shall ask the user to sign up to their account.	5
REQ-13	The system shall ask the user to log in to their account.	5
REQ-14	The system shall send notifications to the user.	5

REQ-15	The system shall save any changes made by the user.	5
REQ-16	The user shall classify themself as a leader or team member.	5
REQ-17	The system shall support different languages such (Arabic, English) after logging in .	5
REQ-18	The system shall provide users with a way to contact support if they encounter issues or have questions.	5
REQ-19	The user should be able to lock a list with a password after logging in.	3
REQ-20	The user as leader shall be able to assign one or more tasks to each user in a team after logging in.	5
REQ-21	The system shall validate the user’s email.	5
REQ-22	The system shall ask the user to scan their membership QR code to access their tasks.	5
REQ-23	The system shall provide a dropdown list of member names in case assigning more tasks to the same member again.	3
REQ-24	The user should be able to share or copy the membership QR code after logging in.	5

2.2 Enumerated Non-Functional Requirements

REQ-ID	**DESCRIPTION**	***PW***
REQ-25	The system shall save any changes made by the user within 10 seconds.	5
REQ-26	The system shall send the to-do list to other users within 15 seconds.	5
REQ-27	The system shall allow the user to log in within 3 seconds.	5
REQ-28	If multiple users are working on the same task list, they shall authenticate themselves using the QR code that the leader sends to them.	5
REQ-29	The system shall have a response time within one second for all user inputs and commands, such as creating and deleting a task.	5
REQ-30	The system shall provide password-protected access to the account.	5

3. Functional Requirements Specification

3.1 Stakeholders

1. **Users** - They are the primary stakeholders who use the app to manage their tasks and organize their work.

2. **Developers** - They have interest in designing, developing, and maintaining the system, ensuring that the app is functional, user-friendly, bug-free, and meets the user's requirements and expectations.

3. **Advertisers** - The individuals or organizations who may use the app to advertise their products or services. They are interested in the app's ability to attract users, as this will impact the effectiveness of their advertising campaigns.

3.2 Actors and Goals

Actor	Type	Goal
Client	Initiating	The customer who uses the application and needs to manage his or her tasks.
System Administrator	Participating	It is responsible for providing features that are required by the user.

3.3 Use Cases

3.3.1 Use Case Casual Description

Use Case ID	Name	Short description	Corresponding REQ-ID
UC1	Create a task/list	Allow the users to create their own list and tasks inside a list after logging in.	REQ-1, REQ-7
UC2	Schedule task	Allow the users to schedule their own tasks, set deadlines and prioritize them after logging in.	REQ-2, REQ-8
UC3	Customize task	Allow the users to add notes after logging in.	REQ-3
UC4	Customize settings	Allow the user to customize their account setting, such as choosing their preferred language (English and Arabic) and locking their list with a password after logging in.	REQ-4, REQ-17, REQ19
UC5	Send task	Allow the users to share their tasks with others after logging in.	REQ-5
UC6	Status of task	Allow the user to check the task when he/she is finished after logging in.	REQ-6
UC7	Create categories	Allow users to create their own categories based on their interests after logging in.	REQ-9
UC8	Search lists	Allow the user to search among lists in categories if required.	REQ-10
UC9	Manage teamwork	The system allows users to work together on the same task list such as classify users, share QR code, scan membership code.	REQ-11, REQ-16, REQ-22 , REQ-24
UC10	Sign up	Allow the user to sign up to the system	REQ-12
UC11	Log in	Allow the user to log in to the system	REQ-13
UC12	Send notification	Allow the system to send notifications to remind the user about a task	REQ-14
UC13	Save changes	Allow the system to save any change made by the user.	REQ-15

UC14	Classify user	Allow the system to assign each user either as a leader or team member.	REQ-16
UC15	Provide technical support	Allow the system to provide a way to contact support.	REQ-18
UC16	Assign tasks	Allow the leader to assign a single task or more to a new team member. A drop-down list of names will be provided by the system in case the member was assigned to a task before, after logging in.	REQ-20, REQ-23
UC17	Provide Authentication	Allow the system to validate the user's email.	REQ-21
UC18	Scan the membership QR code	Allow the user to scan the QR code to access their tasks.	REQ-22
UC19	Share/ copy the QR code	Allow the user to share/copy the membership QR code after logging in.	REQ-24

3.3.2 Use Case Diagram:

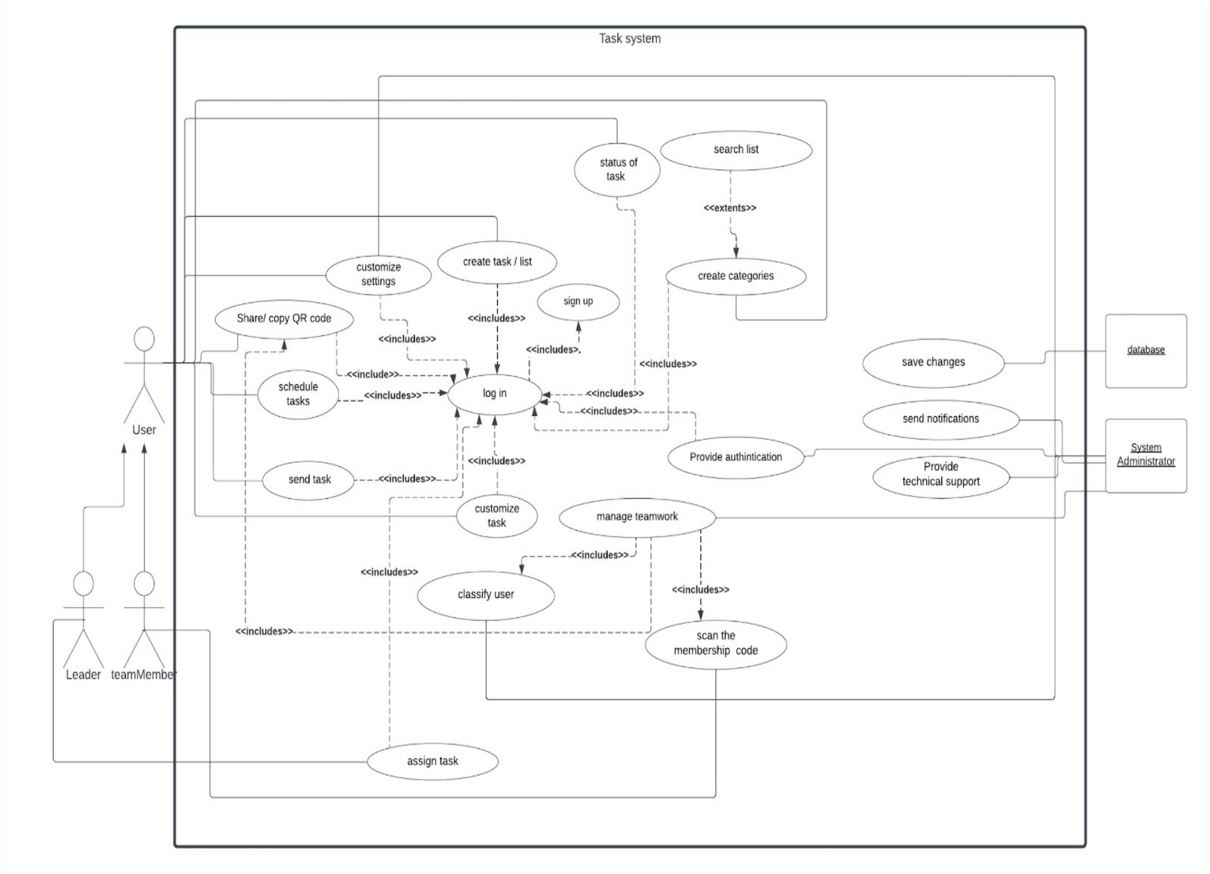


Figure 1. Use Case Diagram

3.3.3 Fully-Dressed Description

UC8 <Search lists>

Initiating Actor: Client.

Participating Actor: System Administrator.

Actor's Goal: To search among the lists in categories.

Pre-condition: The user creates categories based on their needs.

Post-condition: The user searches among categories to access information quickly.

Normal flow:

1. The user creates categories.
2. The user searches among lists in categories.
3. The user selects a task list within the category that was searched for.

Alternative Flows (Extensions):

2.1 The user enters an incorrect category name.

2.2 The system will provide a message with no match result.

UC11 <Log in>

Initiating Actor: Client.

Participating Actor: System Administrator.

Actor's Goal: To log into the system.

Pre-condition: The user must have a valid and registered account.

Post-condition: The system displays the interface and allows the user to access all the features and functionality.

Normal flow:

1. The client wants to log in to the system.
2. The client will enter his/her information.
3. The system will check if the information entered is right and if the client already has an account.
4. The user can interact with the system and use all its functionality.

Alternative Flows (Extensions):

3.1 The user doesn't have an account.

3.2 The system will reject the user's request.

UC12 <Send notifications>

Initiating Actor: Client.

Participating Actor: System Administrator.

Actor's Goal: Select the desired task to request for notifications reminder before the deadline.

Pre-condition: The client must be logged in and created a task.

Post-condition: The system will send reminders for a specific task.

Normal Flow:

1. The client chooses the desired tasks.
2. System will request access to the user's notification center.
3. The user will accept the system's request.
4. The reminder will be sent to the user before the deadline.

Alternative Flow:

3.1 The user will reject the system's request to access the notification center.

UC16 < Assign tasks>

Initiating Actor: The client as a leader.

Participating Actor: System Administrator.

Actor's Goal: Assign a single task or more to a new team member. A drop-down list of names will be provided by the system in case the member was assigned to a task before.

Pre-condition: The client must be logged in as a leader.

Post-condition: Assigning tasks to all team members.

Normal flow:

1. The admin selects a task.
2. The screen will retrieve the details.
3. The details will pop out on the screen.
4. The leader will assign tasks to a new member.
5. The leader will enter the member's information and assign a task to them by sharing a QR code.

Alternative Flows (Extensions):

1.1 No tasks to be selected.

4.1 The leader will choose a member that was assigned to a task before from the drop-down list.

5.1 The user enters incorrect information about a member.

UC17 <Scan the membership QR code>

Initiating Actor: Client.

Participating Actor: System Administrator.

Actor's Goal: to give the team members access to their tasks according to their membership QR code.

Pre-condition: The system must classify the user as a team member or leader and give each user their own access.

Post-condition: Each user has access to their tasks list based on their membership code, so they can change the list whenever they want such as marking the task as completed.

Normal flow:

1. The system classifies the user as a leader or team member by giving them a membership code.
2. The users scan the given membership code to be a part of the team.
3. The system gives access to the list for each team member based on their membership code.
4. The user can interact with the tasks list and use all its functionality.

Alternative Flows (Extensions):

3.1 The users attempt to access tasks that are not accessible to them due to their privileges.

3.2 The system will reject the user's request.

4. Interaction Diagrams

4.1 System Sequence Diagram

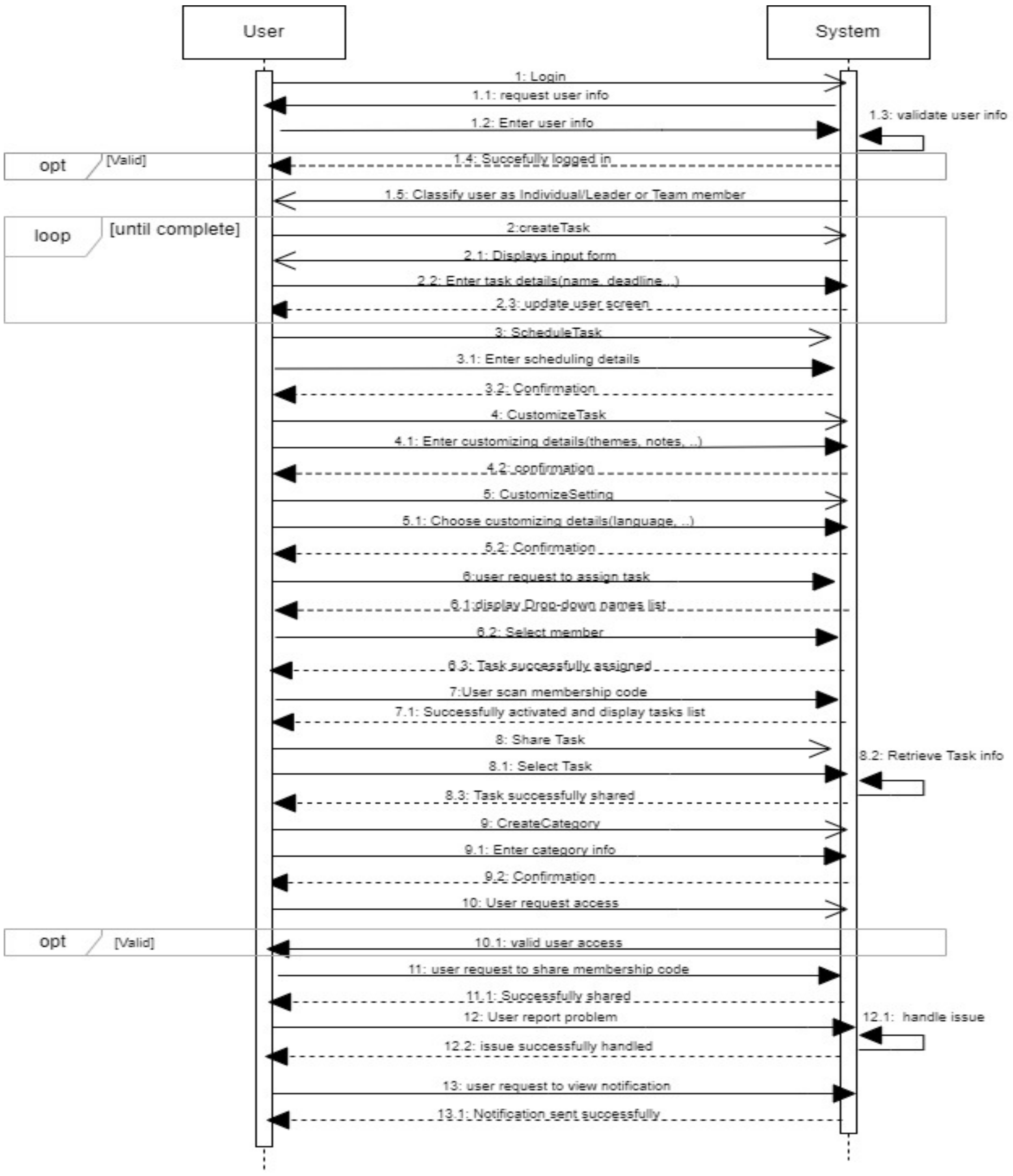

User
System
1: Login
1.1: request user info
1.2: Enter user info
1.3: validate user info
opt
[Valid]
1.4: Succefully logged in
1.5: Classify user as Individual/Leader or Team member
loop
[until complete]
2:createTask
2.1: Displays input form
2.2: Enter task details(name, deadline...)
2.3: update user screen
3: ScheduleTask
3.1: Enter scheduling details
3.2: Confirmation
4: CustomizeTask
4.1: Enter customizing details(themes, notes, ..)
4.2: confirmation
5: CustomizeSetting
5.1: Choose customizing details(language, ..)
5.2: Confirmation
6:user request to assign task
6.1:display Drop-down names list
6.2: Select member
6.3: Task successfully assigned
7:User scan membership code
7.1: Successfully activated and display tasks list
8: Share Task
8.1: Select Task
8.2: Retrieve Task info
8.3: Task successfully shared
9: CreateCategory
9.1: Enter category info
9.2: Confirmation
10: User request access
opt
[Valid]
10.1: valid user access
11: user request to share membership code
11.1: Successfully shared
12: User report problem
12.1: handle issue
12.2: issue successfully handled
13: user request to view notification
13.1: Notification sent successfully

Figure 2. System Sequence Diagram

4.2 Sequence Diagrams

UC11 <Log in>:

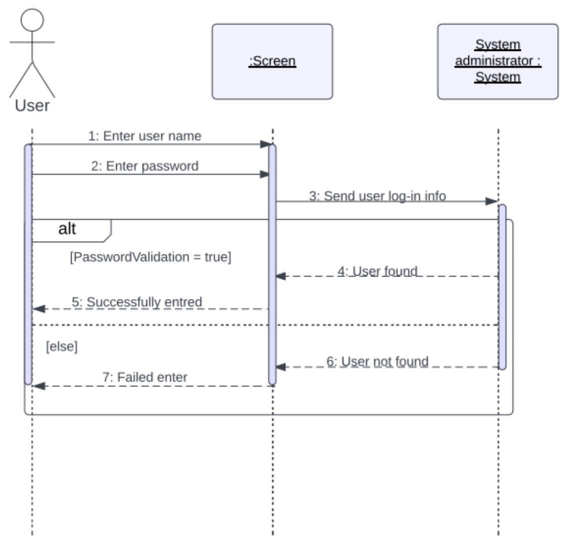


Figure 3. Sequence Diagram for <Log in>

UC12 < Send notification>:

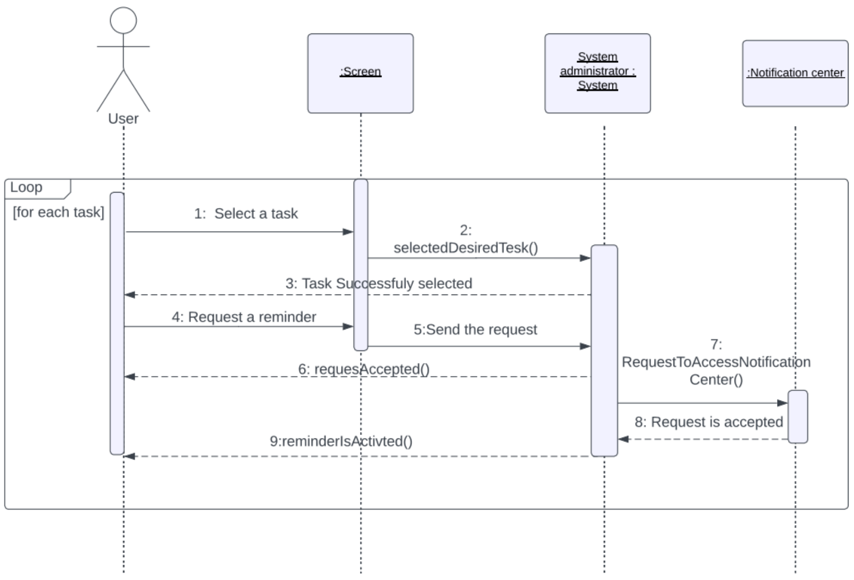
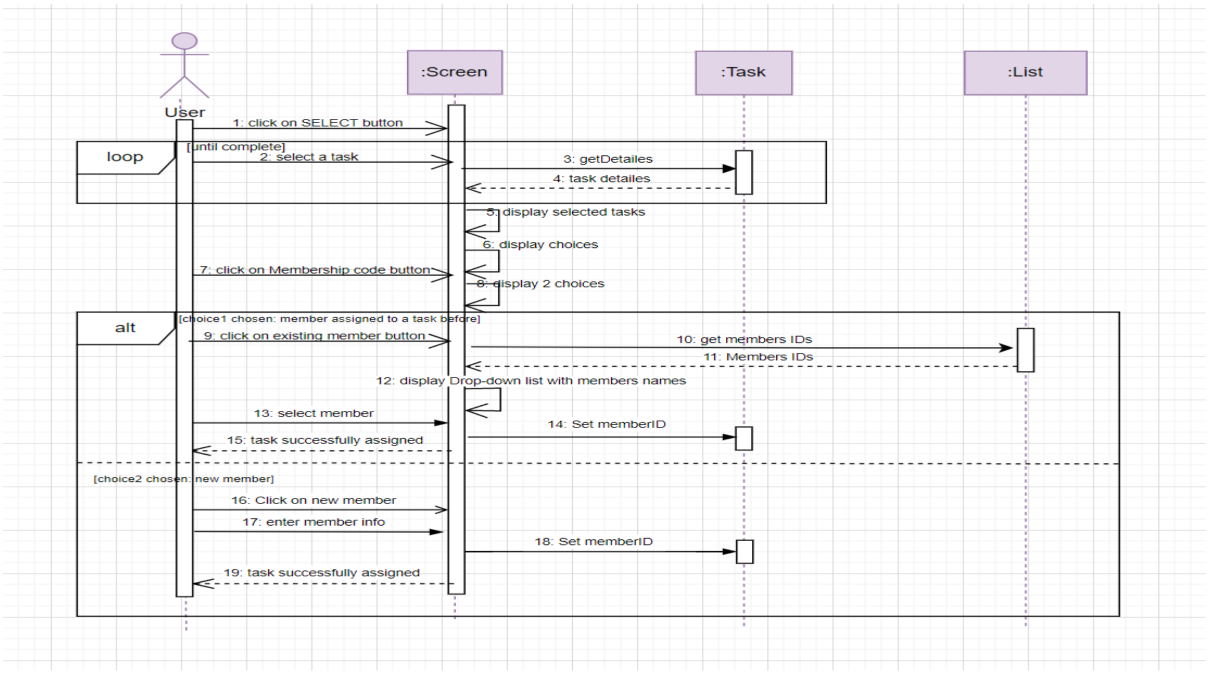


Figure 4. Sequence Diagram for <Send notification>

UC16 <Assign tasks>

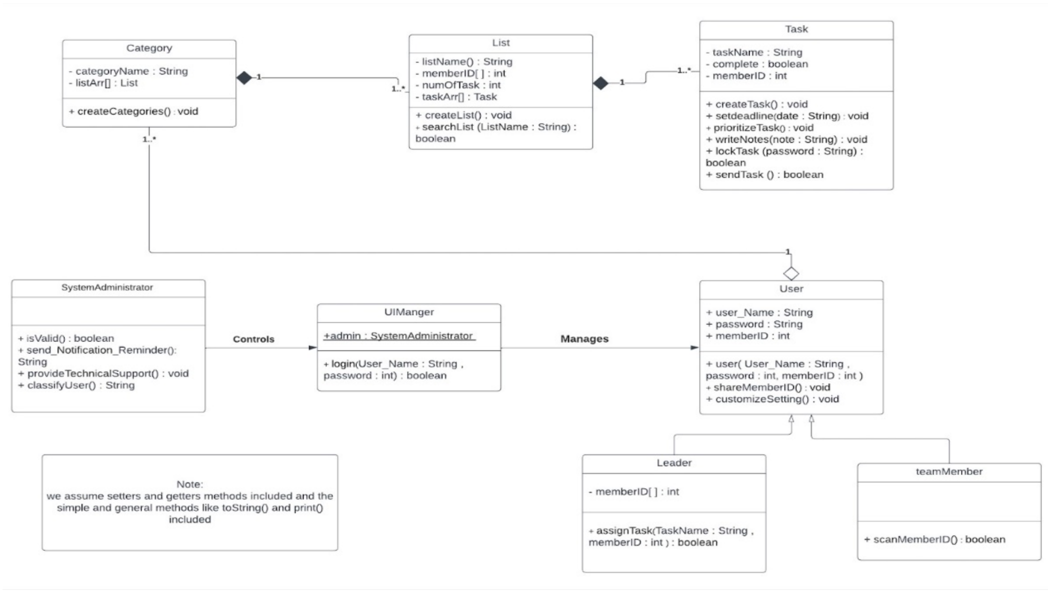


User
:Screen
:Task
:List
1: click on SELECT button
loop
[until complete]
2: select a task
3: getDetailes
4: task detailes
5: display selected tasks
6: display choices
7: click on Membership code button
8: display 2 choices
alt
[choice1 chosen: member assigned to a task before]
9: click on existing member button
10: get members IDs
11: Members IDs
12: display Drop-down list with members names
13: select member
14: Set memberID
15: task successfully assigned
[choice2 chosen: new member]
16: Click on new member
17: enter member info
18: Set memberID
19: task successfully assigned

Figure 5. Sequence Diagram for <Assign tasks>

5. System Architecture and System Design

5.1 System structural diagram



Category
- categoryName : String
- listArr[] : List
+ createCategories() : void
1
1..*
List
- listName() : String
- memberID[ ] : int
- numOfTask : int
- taskArr[] : Task
+ createList() : void
+ searchList (ListName : String) : boolean
1
1..*
Task
- taskName : String
- complete : boolean
- memberID : int
+ createTask() : void
+ setdeadline(date : String) : void
+ prioritizeTask() : void
+ writeNotes(note : String) : void
+ lockTask (password : String) : boolean
+ sendTask () : boolean
1..*
1
SystemAdministrator
+ isValid() : boolean
+ send_Notification_Reminder(): String
+ provideTechnicalSupport() : void
+ classifyUser() : String
Controls
UIManger
+admin : SystemAdministrator
+ login(User_Name : String , password : int) : boolean
Manages
User
+ user_Name : String
+ password : String
+ memberID : int
+ user(User_Name : String , password : int, memberID : int)
+ shareMemberID() : void
+ customizeSetting() : void
Note:
we assume setters and getters methods included and the simple and general methods like toString() and print() included
Leader
- memberID[ ] : int
+ assignTask(TaskName : String , memberID : int) : boolean
teamMember
+ scanMemberID() : boolean

Figure 6. Class Diagram

The Attributes and Methods description for each class

User:

Attributes:

- user_Name : String

Client username.

- password : int

Client password.

- memberID : int

QR membership code.

Methods:

+ user(user_Name : String, password: int, memberID : int)

A constructer to initialize attributes

+ ShareMemberID (): void

This method allows the users to share their own QR membership Code.

+ customizeSetting(): void

This method used for customizing user's account such as preferred language.

A- Leader:

Attributes:

- memberID[ ]: int

Array of all users’ id.

Methods:

+ AssignTask(TaskName : String, memberID : int): boolean

By using this method, the leader assigns tasks to the team members based on their QR codes.

B- teamMember:

Attributes:

The class doesn't have attributes.

Methods:

+ scanMemberID(): boolean

This method used for scanning QR membership Codes to activate it.

SystemAdministrator:

Attributes :

The class doesn't have attributes.

Methods:

+ isValid(): boolean

This method used to validate the QR code by sending an email

+ Send_Notification_Reminder(): String

This method used to send reminders/notifications to the users.

+ ProvideTechnicalSupport(): void

This method used for solving technical issues.

+ classifyUser(): String

This method used for categorizing the user as leader or team member.

Category:

Attributes :

- categoryName: String

The name of category.

- ListArr[]: List

Array of list.

Methods:

+ createCategories(): void

This method used for creating new categories based on the user needs.

Task:

Attributes:

- taskName: String

Name of the task.

- complete: boolean

The status of the task.

- memberID: int

QR membership code.

Methods :

+ create Task(): void

This method used to create new task.
+ setdeadline(date: String): void

This method used to set deadline for a task.

+ prioritizeTask(): void

This method used to prioritize the task based on the user decision.

+ writeNotes(note: String): void

This method used to write notes for the tasks.

+lockTask (password: String): boolean

This method used to lock the tasks list with a password.

+ sendTask (): boolean

This method used to send the task to other.

List:

Attributes:

- listName(): String

Name of task list.
- memberID[ ]: int

Array of all users' id.
- numOfTask : int

Number of tasks in one list.

- taskArr[]: Task

Array of tasks inside one list.

Methods:

+ createList(): void

This method used to create new list of tasks.

+ searchList (ListName : String): boolean

This method is used to search for a specific list in a categorie.

UIManger:

Atrributes:

+admin: SystemAdministrator

This object used for doing different operations and calling other methods.

Methods:

+ login(User_Name : String, password: int):boolean

This method used to log into the app by entering a username and password.

<object diagram>

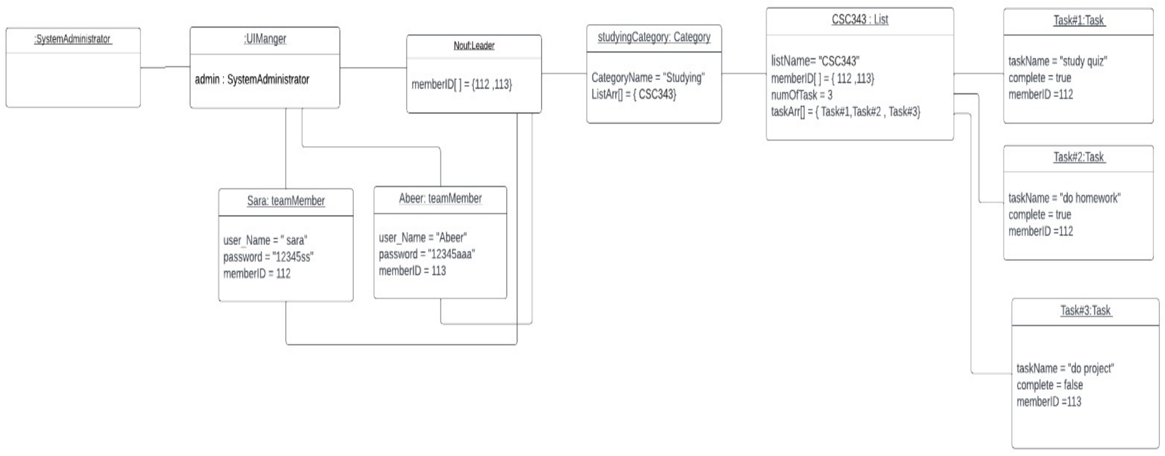


Figure 7. Object Diagram

5.2 Architectural diagram

5.2.1 The System Organization

The client-server architecture is a system in which the majority of the resources and services requested by the client are hosted, delivered, and managed by the system. In this model, all requests and services are delivered through a network. This architecture is beneficial for businesses because it allows for scalability, reliability, and flexibility in the network. Furthermore, it provides better control over the resources and services, allowing for improved security and performance. It is also known as a client-server computer network or the networking computing model. As a result of our decision to design a client-server architecture, the system is divided into client and server components, each of which is responsible for its own functions. The client side is responsible for interfaces and interaction, while the system side is responsible for data storage and processing. By using this architecture, data can be protected more easily. In order to prevent unauthorized access, access controls, authentication, and encryption can be enforced at the server level.

5.2.2 The Modular Decomposition

An object model represents real-world entities, objects and interactions. In our application "ToDone", we have various objects such as "Task," "List," and "User." All of these objects can have their own properties and methods, which allows for encapsulation and reuse. This object-oriented approach allows us to quickly develop our application with a modular design. We can also extend existing objects with updated properties and methods as needed. This makes our application easy to maintain and scale in the future. An object-oriented model allows us to modularize the application by separating different functionalities into different classes or modules. In order to provide the application's overall functionality, these modules must be able to interact with each other through well-defined interfaces.

5.2.3 The Control Model

The event-driven architecture uses events to trigger and communicate between decoupled services. An event is a change in state or update, such as a task being added to a list. By using this method, different services are able to communicate asynchronously without knowing each other's details. The application can be scaled and maintained more easily since services can be added and removed without affecting the entire system. Our system can create categories and lists of tasks. It sends reminders to the user so he can finish the task before the deadline. The leader can distribute tasks between team members which helps the team manage their time and optimize their workflow. Our system allows users to modify their settings and tasks. After all operations, the database saves all changes immediately. In addition, when using event-driven modeling, services operate independently. This helps increase scalability and reduce errors. Finally, the system provides real-time feedback to the user, making it easier to monitor and adjust operations.

Identifying the Subsystem

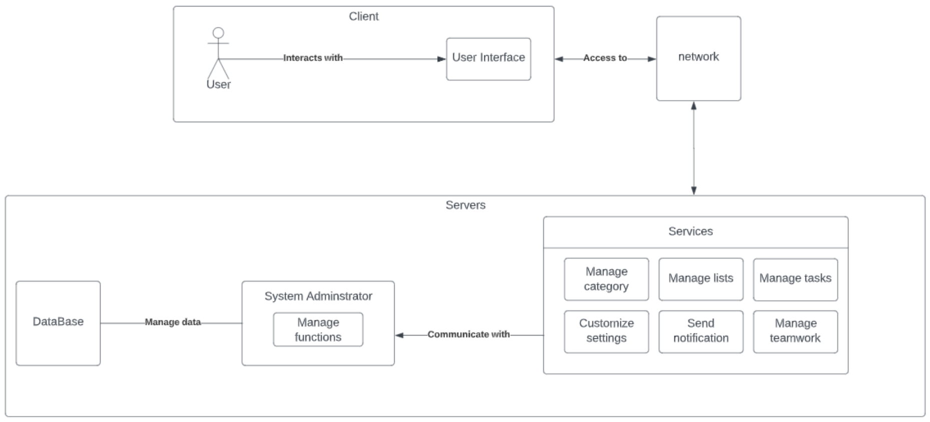


Figure 8. Subsystems block diagram

5.3 System behavioral diagram

State diagram of a "Task" object:

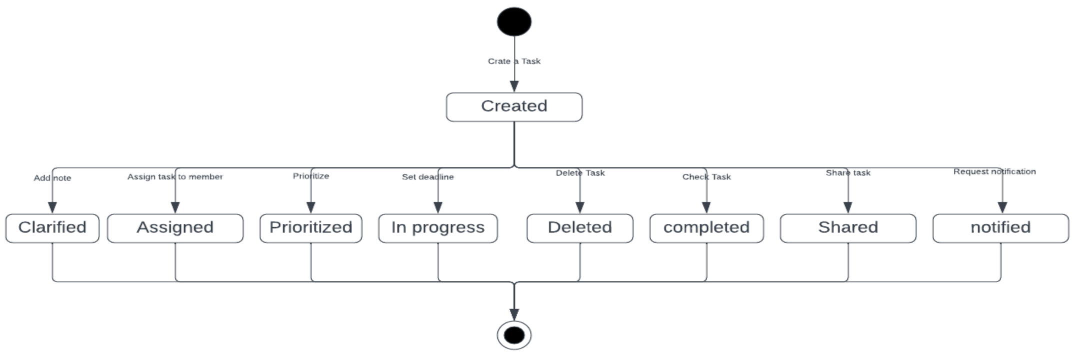
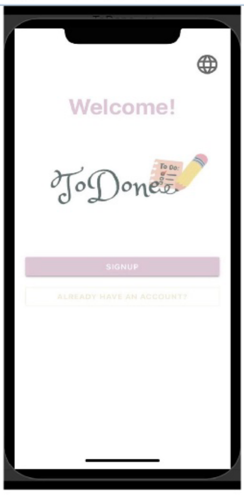


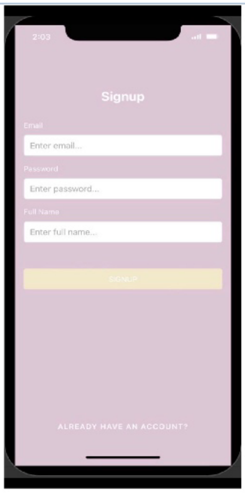
Figure 9. State diagram

6. User Interface Design

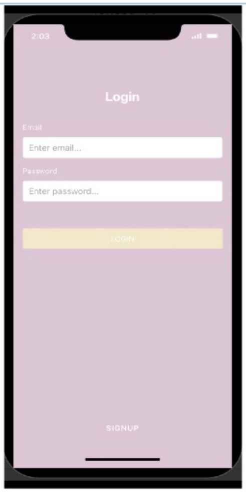
A. The user as leader



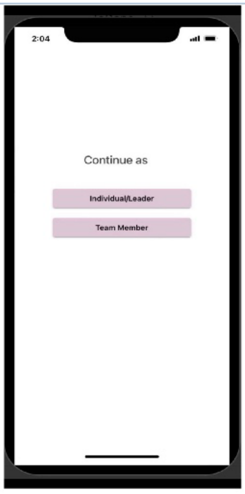
Welcome screen.



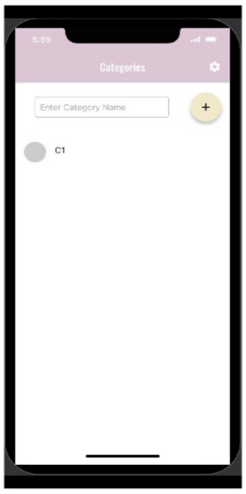
Sign up screen.



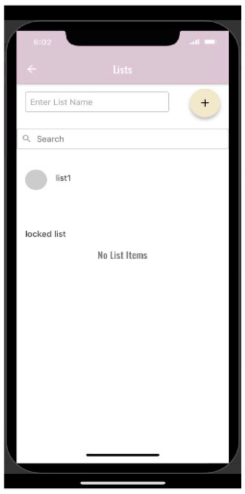
Log in screen.



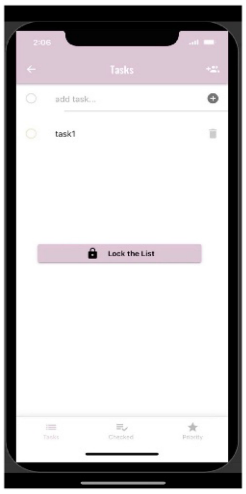
User classification screen.



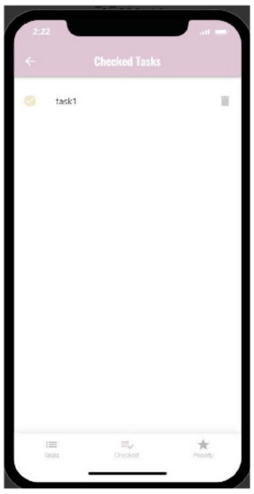
Leader/Individual Home Screen.
Creating new category.



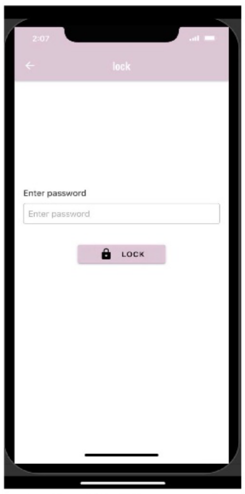
Creating a list inside C1.



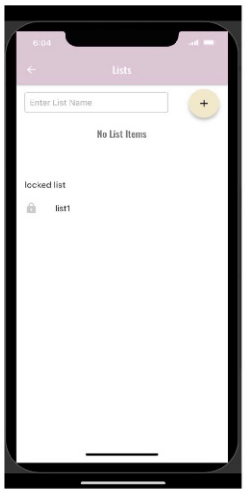
Creating a task inside list1.



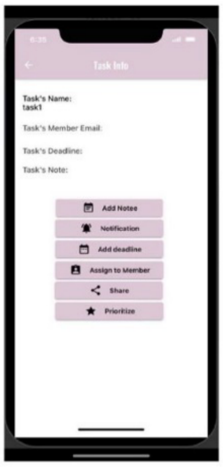
Checked tasks screen task1 was checked.



Locking a list with a password.



list1 was locked with a password.



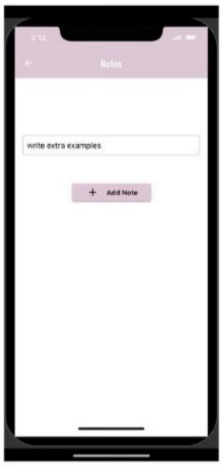
Task's Name:
task1

Task's Member Email:

Task's Deadline:

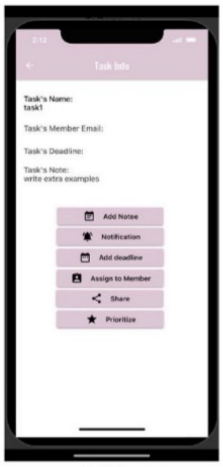
Task's Note:

Tasks info screen
User choose to add notes



write extra examples

Adding note screen
User wrote his note



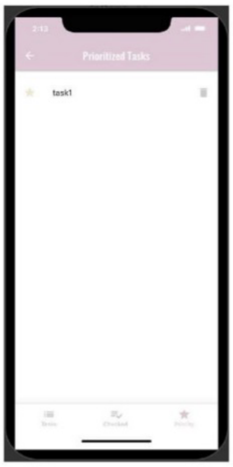
Task's Name:
task1

Task's Member Email:

Task's Deadline:

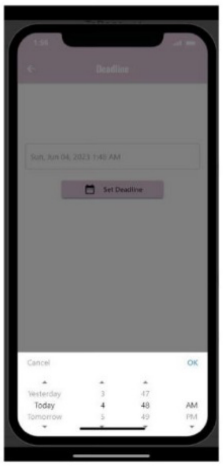
Task's Note:
write extra examples

Task info screen
1-The note was added
2-user choose to prioritize his note
3-user choose to set deadline of his task



task1

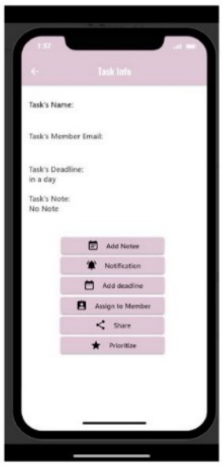
Prioritized tasks screen
User prioritized his note



Sun, Jun 04, 2023 1:40 AM

Yesterday 3 47
Today 4 48 AM
Tomorrow 5 49 PM

Set deadlines screen
User set deadline



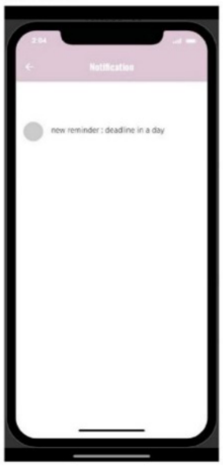
Task's Name:

Task's Member Email:

Task's Deadline:
in a day

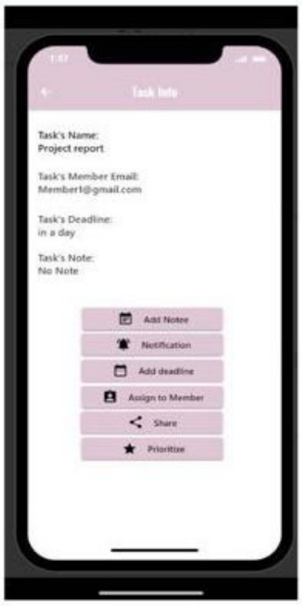
Task's Note:
No Note

Task info screen
Deadline was added



new reminder : deadline in a day

Notifications screen

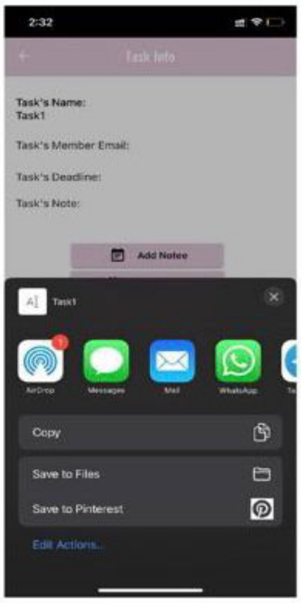


Task's Name:
Project report

Task's Member Email:
Member1@gmail.com

Task's Deadline:
in a day

Task's Note:
No Note



Task's Name:
Task1

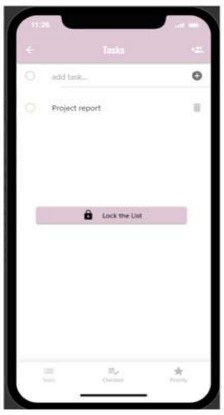
Task's Member Email:

Task's Deadline:

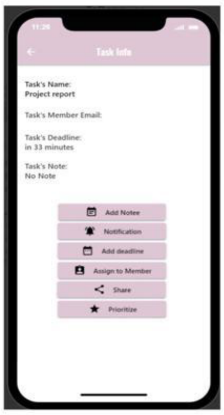
Task's Note:

Task info screen
User choose to share
his task

Selected task was shared



Tasks list for the Leader.



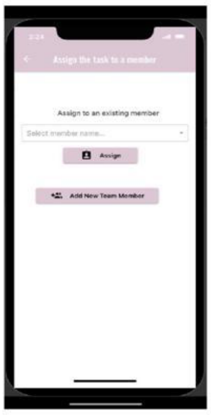
Task's Name:
Project report

Task's Member Email:

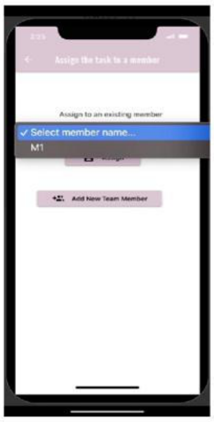
Task's Deadline:
in 33 minutes

Task's Note:
No Note

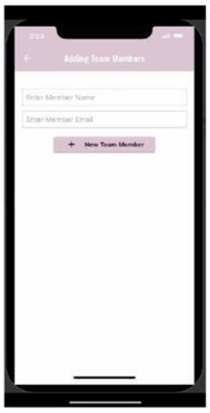
Click a task to view task Info screen.



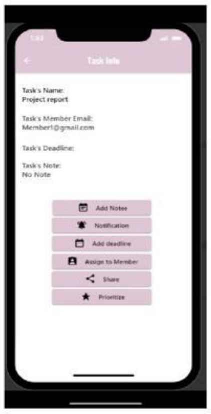
Click on assign to member.



Click on an existing member from the drop-down list.



Click on add new team member button to add a new member.



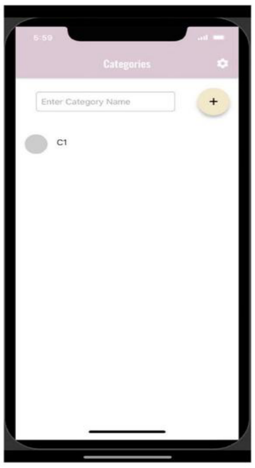
Task's Name:
Project report

Task's Member Email:
Member1@gmail.com

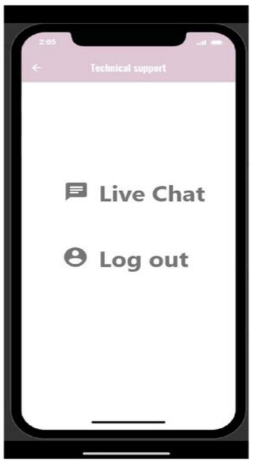
Task's Deadline:

Task's Note:
No Note

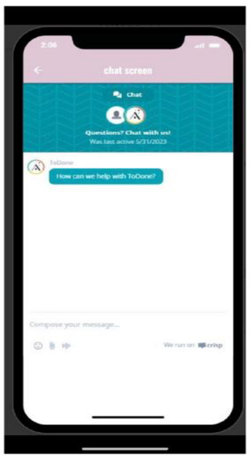
Task was assigned to a member.



Individual/leader home screen
User chose the settings icon



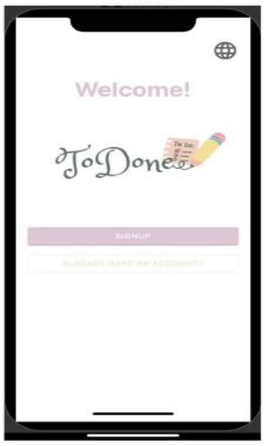
Technical support team
User choose live chat



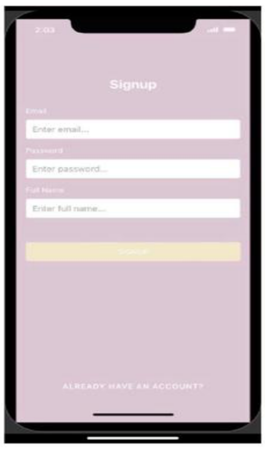
How can we help with ToDone?

Chat screen

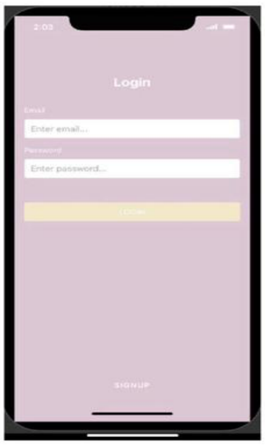
B. The user as team member



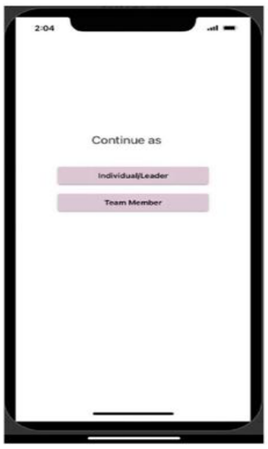
Welcome screen



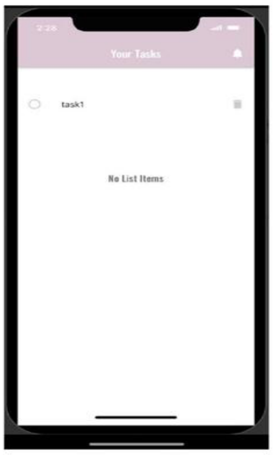
Sign up screen



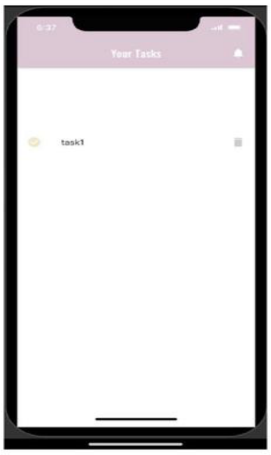
Log in screen



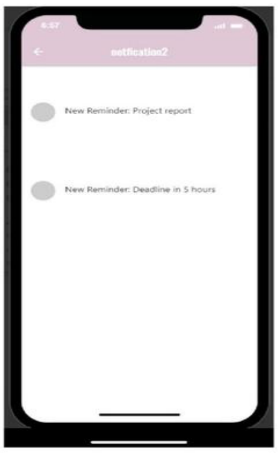
User classification screen
Team member entered to system



Task screen
Task was created



Checked tasks screen


notification2
New Reminder: Project report
New Reminder: Deadline in 5 hours

Notification screen

Figure 9. User Interfac

7. Design of Tests

7.1 Unit Testing:

Test Case Id	Use Case Teste d	Unit name	Description	Test data/input	Expected Output	When it is considered pass/fail	Actual output	Limitation
TC1	UC1, UC2, UC3, UC5 and UC6	**Manage Task**	Tests if the user can create tasks, mark them as completed, scheduling them such as setting deadlines and prioritizing, customizing such as writing notes and sharing.	The user writes the task name and then clicks on the "+" button. Then the user can select the required task and select between "checked" to mark the task as completed or "Priority" or "Add deadline" or "Add Notes" or "share" buttons.	A task is created and marked as completed, scheduled, customized and shared.	Pass if the user can create a task, then able to schedule, customize, share and mark the task as completed. Fail if the user can't create a task, then they will not be able to schedule, customize, share and mark the task as completed.	As expected.	None
TC2	UC4	**Customize settings**	Test if the user can customize settings such as locking lists with a password or changing language.	The user selects the preferable language and chooses the desired list to lock it with a password and clicks the lock button.	The preferred language is selected, and the selected list is locked with a password.	Pass if the user can customize the settings. Fail if the customer can't customize the settings.	As expected .	None
TC3	UC7	**Manage Category**	Tests if the user can create a category.	The user writes the category name and clicks on the "+" button.	The category created.	Pass if the user can create a category. Fail if the user can't create a category .	As expected .	None.
TC4	UC1, UC4, UC8	**Manage list**	1.Tests if the user can create a list	1.The user writes the list name and then	1.The list created successfully.	1.Pass if the user can create a list.	1.As expected .	None.

			and search list. 2.Tests if the user can lock list with password.	clicks on “+” button. 2.The user can search for a specific list by writing its name in the search box. 3.The user selects the list, then clicks on “lock the list” button and then enters the password.	2. The user will find the list if it exists. 3.the list locked successfully.	Fail if the user can’t create a list. 2. Pass if the user can find the list if it exists. Fail if the user cannot find the existing list. 3.Pass if the user can lock a list. Fail if the user can’t lock a list.	2.As expected. 3.As expected.	
TC5	UC12	**Send notification**	Tests if the system can send notifications to users.	The user selects the desired task and then clicks on the “Notification” button then the system notifies the user before the deadlines meet the selected date.	The system sends notifications to users.	Pass if the notification is displayed in the notification center after the system has sent the notification. Fail if the notification was not displayed in the notification center after it was sent by the system.	As expected.	None
TC6	UC9, UC14, UC16, UC18, UC19	**Manage Teamwork**	1. Tests if the user can classify their roles as a leader or team member. 2. Tests if the leader can assign tasks to a team member. 3. Tests if the user can scan the QR code	1. The user clicks on the “individual /leader” button or clicks on “team member” after signing in. 2. The leader selects a task then chooses the “Assign to Member”	1. The users are classified as leaders or team members. 2. Each team member has a task or multiple tasks. 3. The user logs in as a team member then scans their QR	1. Pass if the user can classify their roles as leader /individual or as a team member. Fail if the user can’t classify their roles as a leader or team member. 2. Pass if the leader can	1. As expected. 2. As expected. 3. The user could access their tasks but not through a QR code.	3. because we had problem with the scanning and generating tool, so we find another solution. Since we could not give access to team members through a QR code to

			4. Tests if the user can share or copy their QR code	button then selects the required team member from a drop-down menu or adds a new team member and assigns the task to them. 3. Tests if the users can work together on the same task with the help of the system, such as giving the team Members access to task through scanning a QR code generated by the leader. 4. The user clicks on the "share" button to share their QR code with others.	code to get access to their tasks. 4. The user shared their QR code with others.	assign tasks to team members. Fail if the leader can't assign tasks to team members. 3. Pass if the user can access their tasks. Fail if the user can't access their tasks. 4. Pass if the user can share or copy their QR code with others. Fail if the user can't share or copy their QR code with others.	4. The user could not share their QR code with others.	the leader's list, the leader will give access to members through their registered mail. 4. Since we can't build the QR code so the user can't share their QR code because the scanner is not free.

7.2 Integration Testing

We have adopted the Bottom-up approach which we believed it is the best approach that suits our application because it involves testing the lower level of the components first and then testing the higher-level components, gradually moving up the hierarchy. This approach will ensure that the lower-level components are functionally correct before testing the entire application. This approach also helps to identify any issues or errors early and enables quicker debugging and resolution.

We tested the integration of the "To-Done" app with the database. We verified that data is being successfully stored in the database when new tasks are added and that the data is retrieved correctly when tasks are displayed on the front end. After that, we tested the integration of the "To-Done" app with the APIs. we verified that the API calls are made correctly and that the correct data is being retrieved to display on the front end. Then we tested the frontend components of the "To-Done" app such as the user interface, input forms, and display of tasks. We verified that the front end can successfully display data retrieved from the database and that user input is being successfully sent to the APIs. Then we tested the entire app's functionality, included in task managing, list managing, categories managing, teamwork managing, etc. We tested the features using integration tests to ensure that all features work correctly and do not cause bugs or issues. For example, the user can choose their preferable language. The user as leader/individual first creates a category and then creates multiple lists with searching and locking features. Then, the user can create multiple tasks in a list and then customize, schedule, mark as completed, write notes and share the tasks. If the user classify himself as a leader, he can assign tasks to team members and share membership QR codes with them. If the user classify himself as a team member, he can scan his QR code to access his task then he can mark a task as completed. All users can view notifications of their deadlines or new task assignment in case the user is a team member. The bottom-up approach for integration testing proved to be an efficient testing methodology in ensuring the correct functioning of the to-do list app. By testing the smallest units first and moving upwards to higher-level components, we were able to identify and resolve issues at an early stage, ensuring a seamless integration between the different modules. Using this testing approach, we were able to provide reliable and robust software to our users.

7.3 Acceptance Testing

Alpha Testing:

Our testing team worked primarily on mobile devices running iOS. during this alpha phase, we tested the app in real-world scenarios.

Overall, our alpha testing phase went smoothly with few major issues discovered. One issue we encountered multiple times was the QR code generator and scanner weren’t working as expected. So as a solution, we thought of another way that a Leader can assign a Task to Team members which will be through the registered email of the Team member. Another issue was that notifications didn't always come at the right date which is before the task deadline. And we fixed the notification issue by updating the code associated with the affected buttons.

Beta Testing:

We received feedback from users who tested our application. Their recommendation was to add photos to the notes attached to tasks as a new feature. If a client complains about having difficulty using our app or experiencing difficulties with any feature provided by our application, we will provide a tutorial at the beginning of the first time the client opens the application to demonstrate to the client how to use the application in an easy way and use useful features provided by our application.

Finally, we honestly listen to our client’s input and strive to meet their expectations and demands. Furthermore, we will do all possible to alleviate and manage these difficulties by developing our system and improving algorithms.

7.4 Path Testing

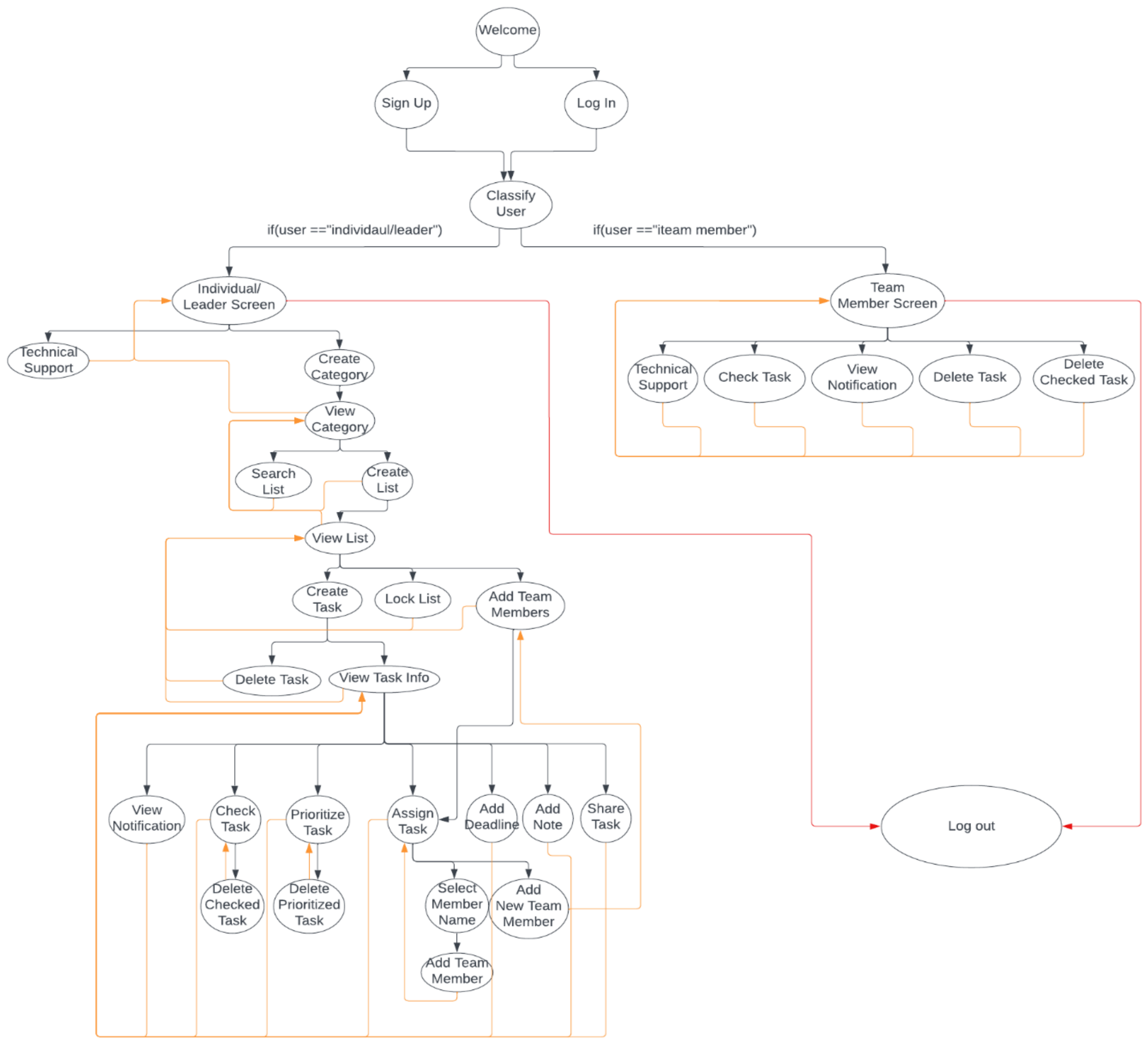


Figure 10. Path Testing Diagram.

Cyclomatic Complexity:

Number of edges – Number of Nodes + 2 = 61 – 35 + 2 = 28

8. References

[1] K. T. Hanna, "icon," WhatIs.com, Jun. 2022, [Online]. Available: https://www.techtarget.com/whatis/definition/icon

[2] "QR Code Security: What are QR codes and are they safe to use?" me-en.kaspersky.com, Mar. 01, 2023. https://meen.kaspersky.com/resource-center/definitions/what-is-a-qr-code-how-to-scan

[3] Techopedia, "What is a Megabyte (MB)? - Definition from Techopedia," Techopedia, Oct. 10, 2016. https://www.techopedia.com/definition/2724/megabyte-m