Kingdom of Saudi Arabia
Ministry of Education
Imam Abdulrahman bin Faisal University
Computer Science department
College of Science and Humanities

جامعة الإمام عبدالرحمن بن فيصل
IMAM ABDULRAHMAN BIN FAISAL UNIVERSITY

CS310: Data Structure Project Report

GoCheck: Task Management System Implemented Using Stack Data Structure

Students Name	ID
Wajood Khalid Al Jearah	2220001292
Nada Alrashidi	2220000552
Norah Alanzi	2220000572
Lojain Alibrahim	2220001292

Instructors:

Dr. Enas Elsharawy – Dr. Haya Aldossary

Academic year 1445 – 2023/2024

Table of Contents

1. Introduction

GoCheck is a Java application designed to help users organize their main goals and achieve them by breaking them down into smaller daily tasks. Users have the flexibility to choose predefined templates or create custom task lists based on specific aspects of their lives, including professional life, spirituality, physical health, personal growth, entertainment, relationships, and financial matters. The application provides users with list management capabilities, allowing them to add, update, check, print, sort, and search their tasks. Additionally, there is an administrative section where administrators can add new templates to the program and perform administrative tasks such as deletion and modification. To ensure efficiency, the program implements a stack data structure, which follows the Last-In-First-Out (LIFO) principle. The stack is implemented using a linked list structure, improving memory usage and user-friendliness.

2. Challenges

1. One of the challenges is using a single class stack to carry out several actions.

2. Creating a new object without the use of an array structure for every unique goal or task.

3. Formatting and printing an extended message in accordance with particular layout specifications.

4. Using a new built-in LocalDate class's features and methods.

5. Handling technical issues and bugs that arise after the project is completed.

6. Managing team members time to enable concurrent work sessions.

7. Handling strict time limitations and deadlines.

8. Differences in coding styles between the team members.

3. Solutions

1. To ensure clarity and modularity, separate the Goal and Task stacks into different classes.

2. Resolving the issue by implementing a line Object obj = new Object(); within a do-while loop construct.

3. Imports an external text file to ensure proper printing of the message.

4. Looks up Java built-in classes on the internet.

5. Identifying and resolving remaining issues through code inspections and testing iterations.

6. Strategically assigning work to team members in order to maximize output and reduce schedule issues.

7. Using efficient time management strategies to fulfill project deliverables and milestones.

8. Using the simplest method or embracing a single coding style to encourage teamwork and innovation.

4. Methods & Explanation

4.1.GoalsStack Class:

- **isEmpty():** checks if the stack is empty.
- **size():** returns the number of elements in the stack.
- **push(Goal goal):** for adding a new goal to the stack.
- **Goal checkGoal():** removes and returns the top goal from the stack.
- **Goal checkAtPos(int position):** removes and returns the goal at a specified position in the stack.
- **Goal checkAll():** removes all goals from the stack.
- **displayGoal():** displays all goals in the stack.
- **find(String val):** searches for a goal with a specified value in the stack.
- **updateGoal(String old, String newGoal):** updates the text of a goal with a new value in the stack.
- **timeLeft(Goal goal):** calculates the time left for a goal's due date.
- **sortGoalsACS():** sorts the goals based on ascending order of due date (from less time left to long time left).
- **sortGoalsDECS():** sorts the goals based on descending order of due date (from long time left to less time left).
- **sortByPriorityHigh():** sorts the goals based on high priority.
- **sortByPriorityLow():** sorts the goals based on low priority

4.2.AdminGoalsStack Class:

same as GoalsStack class, but it is for Admins only.

4.3.TaskStack Class:

- **isEmpty():** checks if the stack is empty.
- **size():**returns the number of elements in the stack.
- **push(Task task):** adding a new task to the stack by creating a new node and setting it as the new top of the stack.
- **Task checkTask():** removes and returns the top task from the stack (similar to the pop() operation in a stack).
- **Task checkAtPos(int position):** removes and returns the task at a specified position in the stack.
- **Task checkAll():**removes all tasks from the stack.
- **displayTask():**displays all tasks in the stack.
- **find(String val):** searches for a task with a specified value in the stack.
- **updateText(String old,String newTask):** updates the text of a task with a new value in the stack.
- **updatePriority(String old,String newTask):** updates the priority of a task with a new value in the stack.
- **updateAspact(String old,String newTask):** updates the aspect of a task with a new value in the stack.
- **updateDueDate(String old, LocalDateTime newDate):** updates the due date of a task with a new value in the stack.
- **timeLeft(Task task):** calculates the time left for a task based on its due date.
- **sortTasksACS():** sorts the tasks in ascending order based on their due dates (from less time left to long time left).
- **sortTasksDECS():** sorts the tasks in descending order based on their due dates (from long time left to less time left).
- **sortByPriorityHigh():**sorts the tasks based on high priority.
- **sortByPriorityLow():**sorts the tasks based on low priority.

4.4.AdminTaskStack Class:

same as TaskStack class, but it is for Admins only.

4.5.Account Class

- **AccountGC()** : Constructor for the AccountGC class.
- **getSize():** Returns the number of nodes (accounts) in the linked list.
- **isEmpty():**Checks if the linked list is empty.
- **addUser(Node newNode):** Adds a new user account to the linked list.
- **find(String username):** Searches for a user account with the specified username.
- **dispalyAccounts():** Displays the information of all user accounts in the linked list.
- **createAccount():** Collects user information and creates a new account.
- **isEmailUnique(String email):** Checks if the given email is unique among existing accounts.
- **isPhoneNumberUnique(String phoneNumber):** Checks if the given phone number is unique among existing accounts.
- **isUsernameUnique(String username):** Checks if the given username is unique among existing accounts.
- **checkEmail(String email):** Validates the format of an email address.
- **checkPassword(String password):** Validates the format of a password based on specific criteria.
- **validateUser(String username, String password):** Validates if the provided username and password match an existing account.
- **checkPNumber(String pNumber):** Validates the format of a phone number.
- **adminAccounts():** Adds predefined admin accounts to the linked list.

4.6. Admin Class

- **welcome():** Displays a welcome message.
- **createTemplate(TaskStack, GoalsStack):** Allows the admin to create a suggestions template.
- **manage():** Manages tasks by providing options for adding, updating, or displaying tasks.
- **printReport(String filename):** Prints a report to a file, including and the number of operations performed by the Admin.
- **addTask():** Creates a new instance of the GoCheck class and calls the showUsermenu() method.
- **updateTask():** Updates either the goal text or task based on user input.
- **display():** Displays either the goal list or task list based on user input.
- **CheckPrioritValue(String answer):** Validates and returns a valid priority value.
- **validGoalDueDate():** Validates and returns a valid goal due date.
- **validTaskDueDate():** Validates and returns a valid task due date.

4.7. User Class

- **suggestions(TaskStack TaskStack, GoalsStack GoalStack):** This method provides goal suggestions to the user based on their chosen goal aspect . It allows the user to select a priority for the goal and generates corresponding tasks associated with the goal.
- **CheckGoalnumber(int):** Validates the user input for the goal number.
- **CheckPrioritValue(String):** Validates the user input for the priority value.
- **FileRead(String):** Reads the contents of a file.
- **calculateDailyPagesOrsavingFor1Month(int):** Calculates the number of pages to read daily to finish a book in one month or calculates the amount of saving daily to save a specific amount of money in one month.
- **YourList(TaskStack TaskStack, GoalsStack GoalStack):** allows the user to add goals and tasks to their respective stacks based on different goal aspects.
- **addTask():**Creates a new instance of the GoCheck class and calls the **showUsermenu()** method.
- **updateTask() :** Updates either the goal text or task based on user input.

- **checkTask()** :Allows the user to check tasks by selecting various options.
- **checkGoal() :**Allows the user to check goals by selecting various options.
- **display() :**Displays either the goal list or task list based on user input.
- **searchFor() :**Searches for tasks based on a specific due date.
- **validGoalDueDate() :**Validates and returns a valid goal due date.
- **validTaskDueDate() :**Validates and returns a valid task due date.

4.8. Task Class:

- **Task(String task_text, Priority task_priority, Aspact aspact):** Constructor that initializes a task with text, priority, and aspect.
- **Task(String task_text, Priority task_priority, LocalDateTime task_dueDate, Aspact aspact):** Constructor that initializes a task with text, priority, due date, and aspect.
- **getTask_text():**Returns the text of the task.
- **setTask_text(String task_text):** Sets the text of the task.
- **getTask_priority():** Returns the priority of the task.
- **setTask_priority(Priority task_priority):** Sets the priority of the task.
- **getTask_dueDate():** Returns the due date of the task.
- **toString_dueDate():** Converts the due date of the task to a custom string format.
- **setTask_dueDate(LocalDateTime task_dueDate):** Sets the due date of the task, validating that it is not before the current date and time.
- **getAspact():** Returns the aspect of the task.
- **setAspact(Aspact aspact):** Sets the aspect of the task.
- **toString():** Returns a string representation of the task, including its text, priority, and due date.

4.9. Goal Class

- **Goal ():** Default constructor for the Goal class.
- **Goal (String task_text, Priority task_priority, LocalDateTime task_dueDate, Aspact goal_aspact):** Constructor that initializes a goal with text, priority, due date, and aspect.
- **Goal (String goal_text, Priority goal_priority, Aspact goal_aspact)**: Constructor that initializes a goal with text, priority, and aspect.
- **getGoal_text():** Returns the text of the goal.
- **setGoal_text(String goal_text):** Sets the text of the goal.
- **getGoal_priority():** Returns the priority of the goal.
- **setGoal_priority(Priority goal_priority):** Sets the priority of the goal.
- **getGoal_dueDate():** Returns the due date of the goal.
- **toString_dueDate():** Converts the due date of the goal to a custom string format.
- **setGoal_dueDate(LocalDateTime goal_dueDate):** Sets the due date of the goal, validating that it is not before the current date and time.
- **getGoal_aspact():** Returns the aspect of the goal.
- **setGoal_aspact(Aspact goal_aspact) :** Sets the aspect of the goal.
- **toString():** Returns a string representation of the goal, including its text, aspect, priority, and due date.

4.10.GoCheck Class

- **main(String[] args) :** The main method that serves as the entry point for the program.
- **showUsermenu():** Displays the menu options for the user and handles user interactions.
- **showAdminMenu():** Displays the menu options for the admin and handles admin interactions.
- **inputValidation(String answer):** Validates user input for "yes" or "no" responses

5. Results & Output Screens

First, **GoCheck** starts by asking to choose a role either a "user" or an "admin". Each role has its own privileges to make the experience awesome and help the system users achieve big goals.

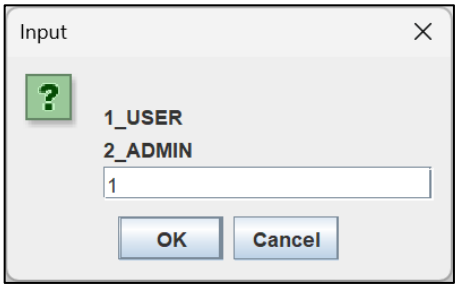

Figure 1. Choose A Role Dialog

5.1. Admin Role

GoCheck has a set number of admins pre-defined in the system, so they log in using a username and password to get started. These special admins have the power to create helpful list suggestions. First, they pick an area of focus from the seven goal aspects. Then, they choose the specific goal and break it down into smaller, more manageable tasks. Finally, they can set suggested due dates for both the overall goal and each individual task.

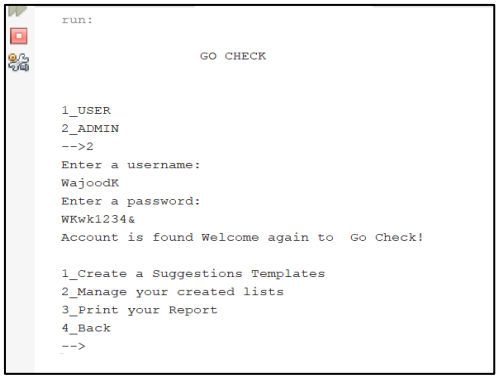

Figure 2. Admin Login

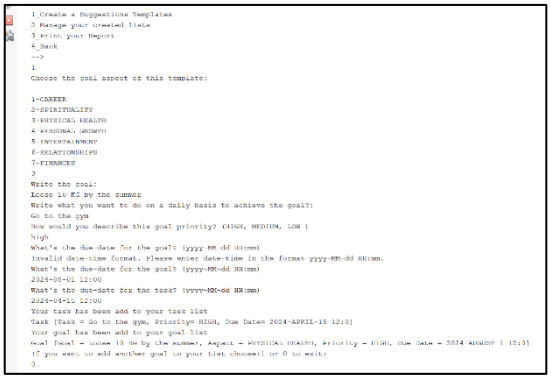

Figure 3. Admin Create Template

Moreover, admins can also manage the lists they create. This includes adding new tasks to existing lists, updating the text of a goal or task itself, and even viewing all the lists they have created in one place. For ultimate control, when an admin updates a task, they will see a menu with a variety of options. This wide range of choices makes managing lists in **GoCheck** smooth and efficient. Furthermore, admins can also see a report that will be printed in a file. This report shows a summary of their activity within the program.

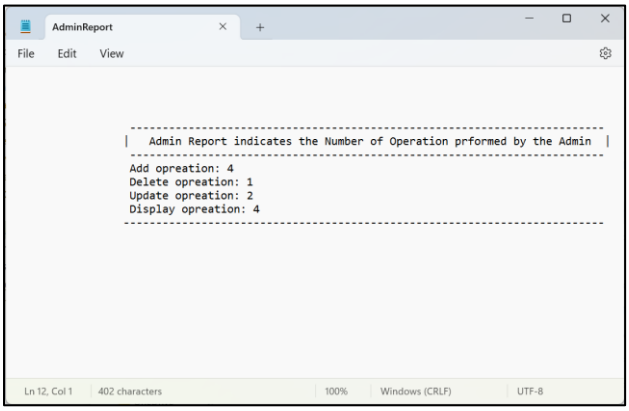

Figure 4. Admin Report

5.2. User Role

Once you log in or sign in to **GoCheck**, a menu will be displayed that allow the user to choose what to perform.

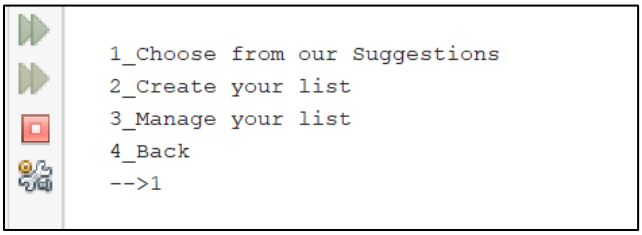

Figure 5. User Menu

Firstly, if users choose the first option, "Choose from our Suggestions," they will see a list that is printed from a file. This list has seven sections (goal aspects) with five suggested goals for each section. Users can pick any goal they like and add it to their own "goals stack." They can also set the importance of this goal (its priority). Then, the program will break the goal down into smaller tasks with suggested due dates to help them achieve it.

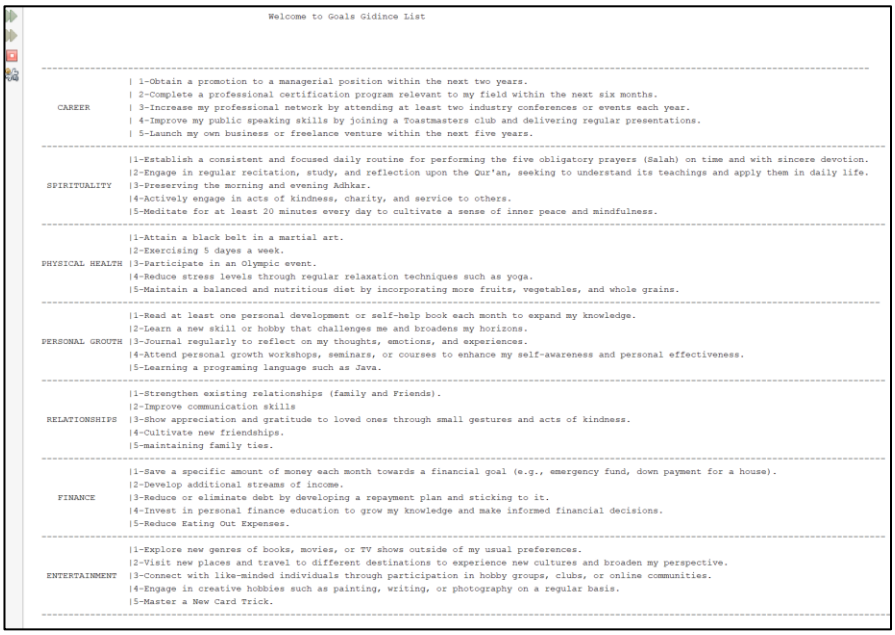

Figure 6. GoalCheck Suggestion List

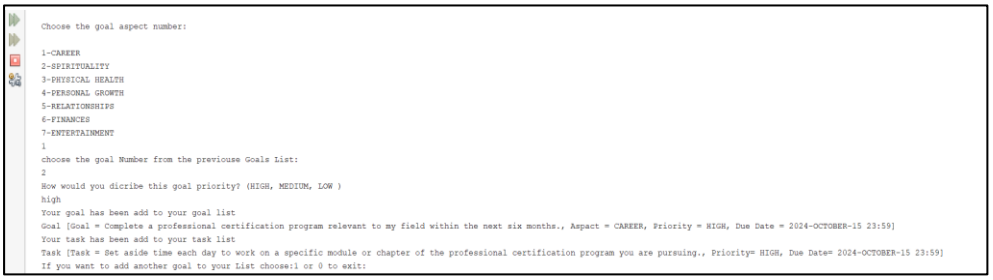

Figure 7. Choose a Goal From List

Secondly, if users pick the second option, "Create your List," they get to build their own custom list. They start by choosing one goal aspect, then write in their specific goal text. The program will then ask them to "Write what you want to do on a daily basis to achieve the goal?" This helps them set the tasks needed to reach their goal. Finally, users can set the priority and due date for both the overall goal and each individual task. Once everything's set, then the goal pushed into their "goal stack" and the tasks to their "task stack."

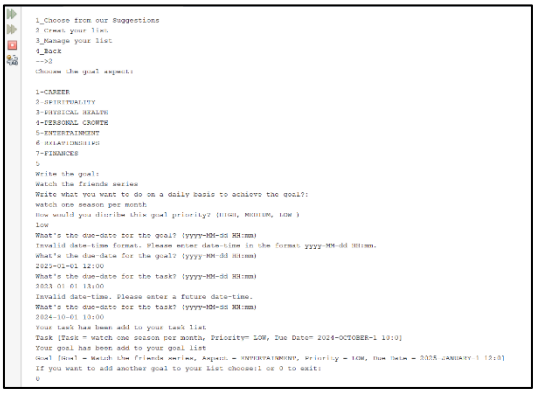

Figure 8. Create Your List

The final option is "Manage your lists." This lets users take control of their existing lists. As shown in Figure 9, there are different actions they can perform to manage their lists effectively. This gives them the flexibility to keep their goals and tasks organized and on track.

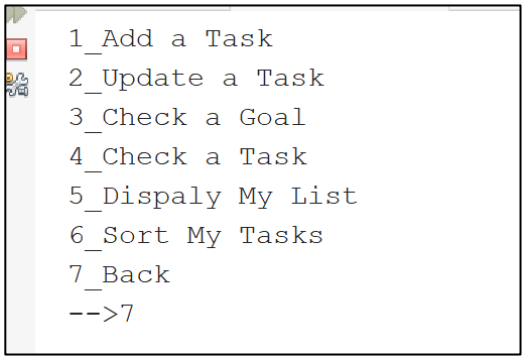

Figure 9. Manage Your List Menu

- **Add Task**

Within "Manage your lists," users can add new tasks. They can either choose from suggested options based on existing templates, or create their own custom tasks. This gives them the flexibility to personalize their approach to achieving their goals.

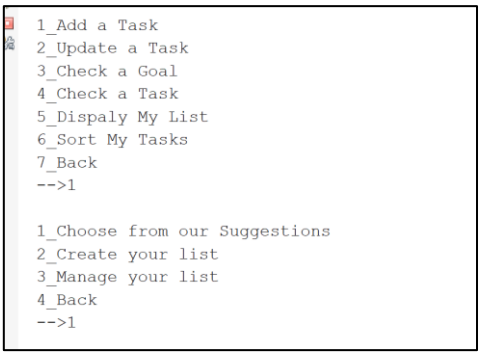

Figure 10. Add Task

- **Update a Task**

One of the options within "Manage your lists" lets users update tasks. They can change the name of the overall goal, or modify the individual tasks themselves. When updating a task, a menu appears (like in the Figure) that gives them control over several parts. They can edit the task description, its priority, the goal aspect it is associated with, or even its due date. This flexibility helps users keep their tasks current and aligned with their evolving goals.

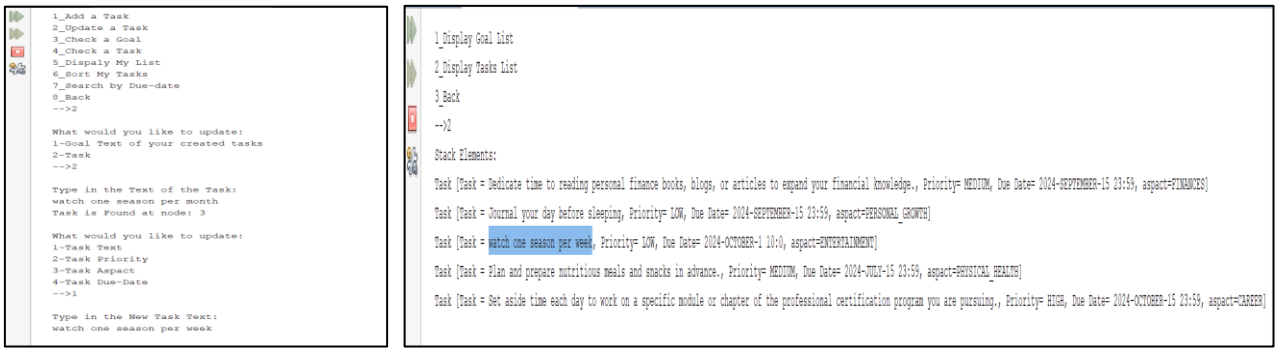

Figure 11. Update a Task

- **Check a Goal and Check a Task**

"Check a goal" lets users mark completed goals from their list. They can choose to check the first goal in their stack (the top element in the stack), check a goal at a specific position in the stack, or even check off all the goals on their stack at once.

"Check a task" works just like "Check a goal," but for their task stack. Users can check off the top task, a task at a specific spot in the list, or all the tasks at once.

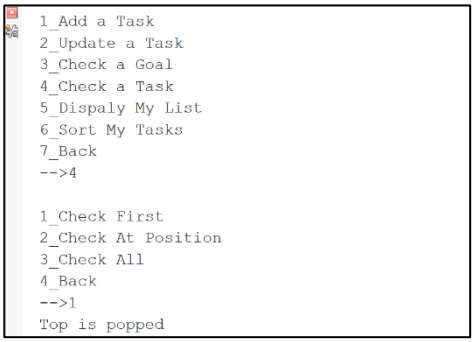

Figure 12. Check a Task

- **Display my List**

Another option in "Manage your lists" lets users view their lists. They can choose to see either their list of goals or their list of tasks, as shown in the figure. This helps the user stay organized and easily access the information they need.

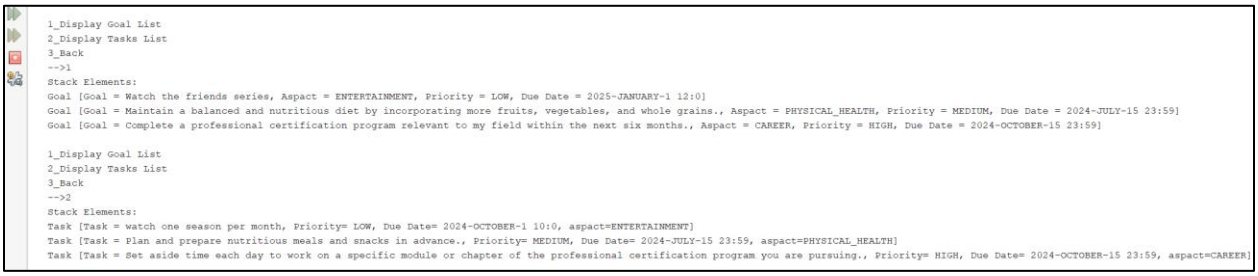

Figure 13. Display Lists

- **Sort my Tasks**

"Manage your lists" also lets users sort their tasks. Users have four different sorting options available, they can sort by the due-date or by the priority.

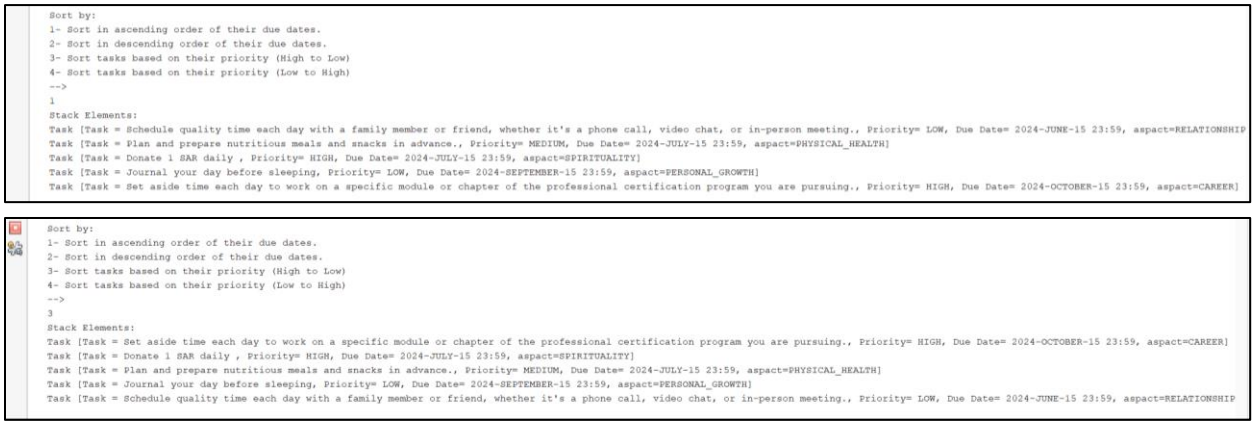

Figure 14. Sort Lists

5.3 Interfaces

In our program, we have implemented clear and simple interfaces to guide the user through the program. For example, there is a welcoming messages dialog or an informative messages dialog to inform the user of adding the goals successfully. Moreover, there is dialogs that has a combo box that allow the user to choose a goal aspect from the menu.

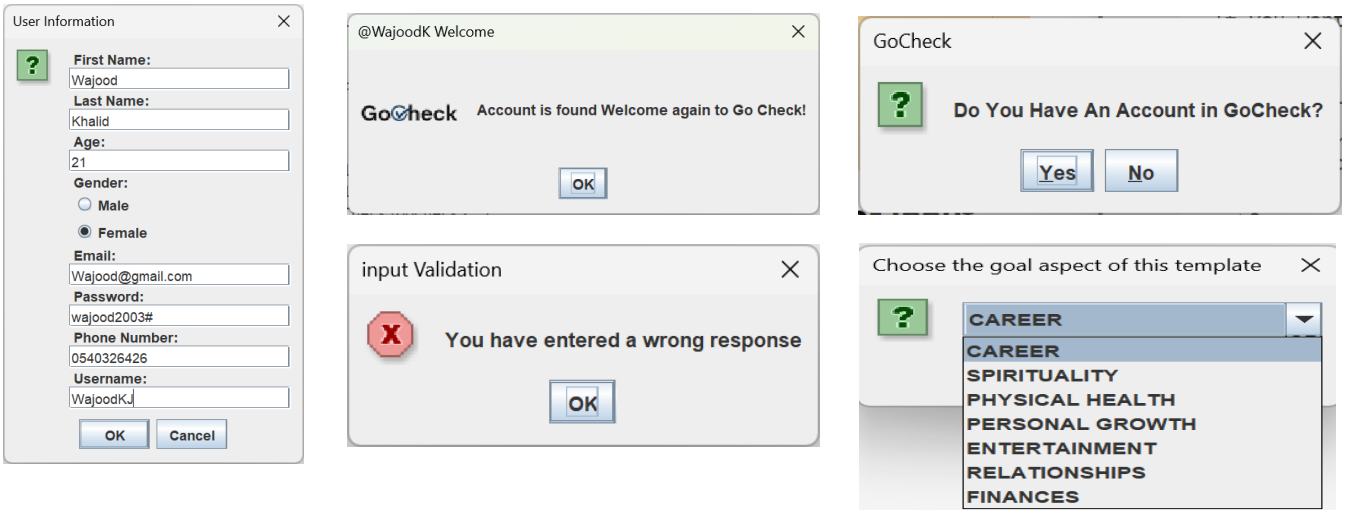

Figure 15. some GoCheck Dialog Messages

6. Conclusion

GoCheck is a program designed to help users organize their goals and tasks in a user-friendly manner. It comes with pre-made templates and user user-made list that allows for easy task management. Additionally, administrators have the ability to create custom template lists and perform various operations on them. With its efficient design, GoCheck assists users in staying on track with their goals and tasks in a highly effective manner.

7. References

- Liang , D. Y. (2023). *Introduction to Java Programming and Data Structures* (12th ed.). Pearson.
- *Class LocalDateTime*. Java Platform SE 8. (n.d.). https://docs.oracle.com/javase/8/docs/api/index.html
- Bruegge, B., & Dutoit, A. H. (2009). Object–oriented software engineering. using uml, patterns, and java. Learning, 5(6), 7.
- Naughton, P., Schildt, H., & Schildt, H. (1999). Java 2: The Complete Reference (p. 1108). Osborne McGraw-Hill.