**Yaşar University**

**Spring, 2022-2023**

**SE2224 - Software System Analysis**

**Final Project Report**

|  |  |
| --- | --- |
| **Student Name:** | **Arda Harman** |
| **Student No:** | **21070006054** |
| **Department Name:** | **Software Engineering** |
| **Course Section No:** | **1** |

**Table of Contents (Do not change the Section Names!)**

[**1** **Introduction** 3](#_Toc101958757)

[**2** **Requirements Definition** 3](#_Toc101958758)

[**2.1** **Functional Requirements** 3](#_Toc101958759)

[**2.2** **Nonfunctional Requirements** 3](#_Toc101958760)

[**3** **Use Case Analysis** 3](#_Toc101958761)

[**3.1** **Actors** 3](#_Toc101958762)

[**3.2** **Scenarios** 3](#_Toc101958763)

[**3.3** **Use Cases** 3](#_Toc101958764)

[**3.4** **Relationships among Actors and Use Cases** 3](#_Toc101958765)

[**3.5** **Use Case Diagram** 3](#_Toc101958766)

[**4** **Behavioral Models** 3](#_Toc101958767)

[**4.1** **Sequence Diagram** 3](#_Toc101958768)

[**5** **Structural Models** 3](#_Toc101958769)

[**5.1** **Class Diagram** 3](#_Toc101958770)

[**6** **Process Modeling** 4](#_Toc101958771)

[**6.1** **Data Flow Diagram (DFD)** 4](#_Toc101958772)

[**7** **Final Version(s) of the Graphical User Interface(s)** 4](#_Toc101958773)

[**8** **Conclusion and Future Work** 4](#_Toc101958774)

# **Introduction**

My project “Task Planner” is an application which is designed for users who wants to keep track and manage their tasks.

Mainly, the app has 2 parts:

1. Login Screen (LoginFrame)

System has a file called “accounts.txt” which contains all the accounts. When user enters his/her username and password, system checks whether this info matches with any accounts at the “accounts.txt” file or not

If matches, user is redirected to the Main Screen, if not user will still stay at the login screen and won’t be able to go to the Main Screen

* At the file I have added an account which has a “username: a, password: b”. You can log in by entering these values

1. Main Screen (MainFrame)

At the main screen, all the tasks are displayed at the screen (in a JTable). There is a database where system fetches the datas from (when main screen is first initialized, it established a connection to database and retrieves datas from there, and disconnects to database when user quits from the app), to display them as “tasks” (a row at the JTable). When user first enters this screen, system notifies the tasks that have less then 1 day prior to their deadline. From there, user can do some operations Here are the operations that user can do at this screen:

* 1. Insert task
     1. User enters the info about the new inserted task
     2. System checks whether there is a mistake at the entered data or not (for example, “deadline” part expects a data which is a “Timestamp” datatype. So, user can’t enter a string at there, system will give error)
     3. System both updates the displayed JTable and the database (note that ID of the task is generated by the system, not the user and also the entry date of that task is automatically done by the system)
  2. Delete task
     1. User selects the task that they want to delete
     2. System checks whether there is a mistake or not (such as not selecting any tasks to delete etc.) then system can delete the task from Jtable and from database
     3. System both updates the displayed JTable and the database
  3. Edit and Update task
     1. User selects the task they want to update
     2. System checks whether there is a mistake at the entered data or not (for example, “deadline” part expects a data which is a “Timestamp” datatype. So, user can’t enter a string at there, system will give error)
     3. System both updates the displayed JTable and the database
  4. Show All tasks
     1. System displays all the tasks without any filters
  5. Order Tasks Which Has Same Deadline (sortByPriority button)
     1. User enters the deadline date (as a Timestamp datatype)
     2. System shows the matching results (note that this operation is only done on JTable and database is not changed)
  6. Display Tasks Within Given Dates
     1. User enters a starting deadline and an ending deadline
     2. System shows matching results (note that this operation is only done on JTable and database is not changed)
  7. Display Task’s Image
     1. User selects the task
     2. If the task has an image system displays that image. If task does not have any image, then system says that “no image is found for task”
     + System takes the ID of the image. If the “ID of task” matches with the “task(no).jpg”, that image is displayed

# **Requirements Definition**

## **Functional Requirements**

* User should be able to login to the system with true account info
* User should be able to see a table which contains the tasks
* User should be able to see “date of today” under the table
* User should be notified when user first enters to the system
* System should be able to establish connection to the database
* Changes made by user, should be made both at the JTable and at the database
* User should be able to insert a task
* User should be able to delete a task
* User should be able to edit and update a task
* User should be able to show all tasks
* User should be able to order the tasks that have the same deadlines with respect to their priority value
* User should be able to display the tasks within the given dates
* User should be able to view the reminder image of a task if that task has a reminder image

## **Nonfunctional Requirements**

* (After user entered their account info to login) System should complete the checking account info less than 5 seconds
* The datas at the database must be encrypted with SSL/TLS encryption
* System should be usable from almost all computers
* The system must perform without failure in 95 percent of the time
* The system should have a clear and intuitive user interface.

# **Use Case Analysis**

## **Actors**

**User:** People who interact with “Task Planner” application by making some operations on the task

**Database:** Information about tasks are stored and retrieved from there. System will fetch the data from there to display the tasks to user

## **Scenarios**

(2 sample scenarios)

**# Sample Scenario 1 – Logging in**

User enters his username and password. Then, he/she presses log in button. After that, system checks the entered account credentials. If entered account info by user is an existing account, then user successfully logs in. Finally user enters to the main screen and system makes a notification about tasks which has 1 days or less to it’s deadline

**# Sample Scenario 2 - Inserting a task**

User successfully logs in. After that, he/she presses “insertTaskButton” button and then he/she enters the info of the inserted task. Then, system validates the infos of the inserted task. If entered values are valid, then new row is inserted to the table and new task is added to the table. Both the system and database are updated.

## **Use Cases**

(3 **complete** use case forms)

**# Complete use case form 1**

Use Case Name: Delete a task

Actor: User, Database

Description: This use case describes a user who wants to delete a task

Trigger: User selects a task to delete and presses “deleteTaskButton”

Preconditions:

1. User is logged in

2. System is able to establish a connection with database

Normal course:

1. User selects a task

2. User presses “deleteTaskButton”

3. User is notified that the operation is successful

4. Task is deleted from the displayed table

5. Task is deleted from database

Postconditions:

1. Database is updated

2. Table is updated

Exceptions:

E1. User selects more than one task (occurs at step 1)

1. System displays message that not only 1 row is selected

2. System does not allow user to do the delete operation

E2. User not selects any task (occurs at step 1)

1. System displays message that not only 1 row is selected

2. System does not allow user to do the delete operation

**# Complete use case form 2**

Use Case Name: Order Tasks Which Has the Same Deadline Respect to Their Priority

Actor: User, Database

Description: This use case describes a user who wants to display all tasks without any filters

Trigger: User selects a task to delete and presses “sortByPriority”

Preconditions:

1. User is logged in

2. System is able to establish a connection with database

Normal course:

1. System displays a screen to user to enter “deadline”

2. User enters a deadline to make system filter which has that deadline

3. System displays the filtered table sorted by the priority value

Postconditions:

1. Data is updated

Alternative courses:

2.1 User enters nothing

1. (Instead of throwing error, system is designed to not throw error) System displays all tasks (without filtering any) but just displays in a sorted order

2.2 User enters an invalid character (anything other than a number)

1. System filters tasks and that does not match with any deadlines, no task is displayed

**# Complete use case form 3**

Use Case Name: Edit and Update Task

Actor: User, Database

Description: This use case describes a user who wants to edit and update a task

Trigger: User selects a task to delete and presses “edit and update task”

Preconditions:

1. User is logged in

2. System is able to establish a connection with database

Normal course:

1. User selects a task

2. User presses “editAndUpdateTaskButton”

3. System displays a screen to user to enter the edited info of the task

4. User changes and updates some info about the selected task

5. User is notified that the operation is successful

6. Table is updated and table now displays the updated version of the task

7. Database is updated and now has the updated version of the task

Postconditions:

1. Database is updated

2. Table is updated

Exceptions:

E1. User selects more than one task (occurs at step 1)

1. System displays message that not only 1 row is selected

2. System does not allow user to do the delete operation

E2. User not selects any task (occurs at step 1)

1. System displays message that not only 1 row is selected

2. System does not allow user to do the delete operation

**(Basic use case forms)**

**# Basic use case form – Show All Tasks**

Use Case Name: Show All Tasks

Actor: User, Database

Description: This use case describes a user who wants to view all tasks

Trigger: User presses “showAllTasksButton”

Preconditions:

1. User is logged in

2. System is able to establish a connection with database

**# Basic use case form – Filter Tasks**

Use Case Name: Filter Tasks

Actor: User, Database

Description: This use case describes a user who wants to tasks which is between the entered period

Trigger: User presses “filterTasksButton”

Preconditions:

1. User is logged in

2. System is able to establish a connection with database

**# Basic use case form – View Reminder Image**

Use Case Name: View Reminder Image

Actor: User, Database

Description: This use case describes a user who wants to view reminder image of the tasks

Trigger: User presses “viewImage” button

Preconditions:

1. User is logged in

2. System is able to establish a connection with database

**# Basic use case form – Connect to Database**

Use Case Name: Connect to database

Actor: Database

Description: This use case describes system establishes a connection to a database

Trigger: User logs in to the system

Preconditions:

1. Username and password which system uses to connect to database must be true

**# Basic use case form – Disconnect from Database**

Use Case Name: Disconnect from database

Actor: Database

Description: This use case describes system disconnects from database

Trigger: User exits from the system

Preconditions:

1. System was already connected before to the database

## **Relationships among Actors and Use Cases**

***User – Log in:*** User is at the login screen and logs in via entering username and password

***User – Insert Task – Database***: User inserts a task and that task is both inserted to the database and to the table

***User – Delete Task – Database***: User selects a task and then deletes that task and that task is both deleted from the database and the screen

***User – Edit and Update Task – Database:*** User selects a task and then updates that task and that task is both updated from the database and the screen

***User – Show All Tasks – Database***: User presses “showAllTasks” button and then system fetches the table from the database and finally system displays the data of the table

***User – View a Task’s Image:*** User selects a task and then system searches whether that task has a reminder image or not and (if that task has a reminder image) system displays reminder image

***User – Order the tasks that have the same deadlines with respect to their priority value:*** User enters a deadline and tasks which has the same deadline and then sorts them by their priority

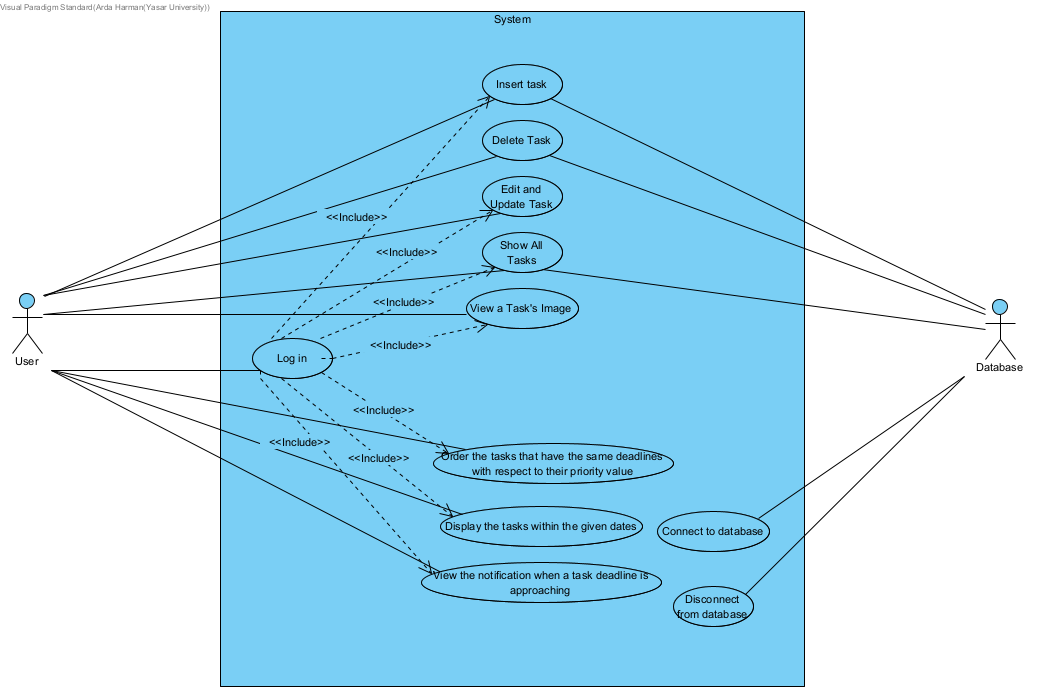
***User – Display the tasks within the given dates:*** User enters a starting and an ending date and system filters tasks

***User – View the notification when a task deadline is approaching:*** When user first enters, system notifies the user by showing tasks which have less than 1 day to expire

***Database – Connect to database:*** System makes a connection with the database by giving right url, username and password

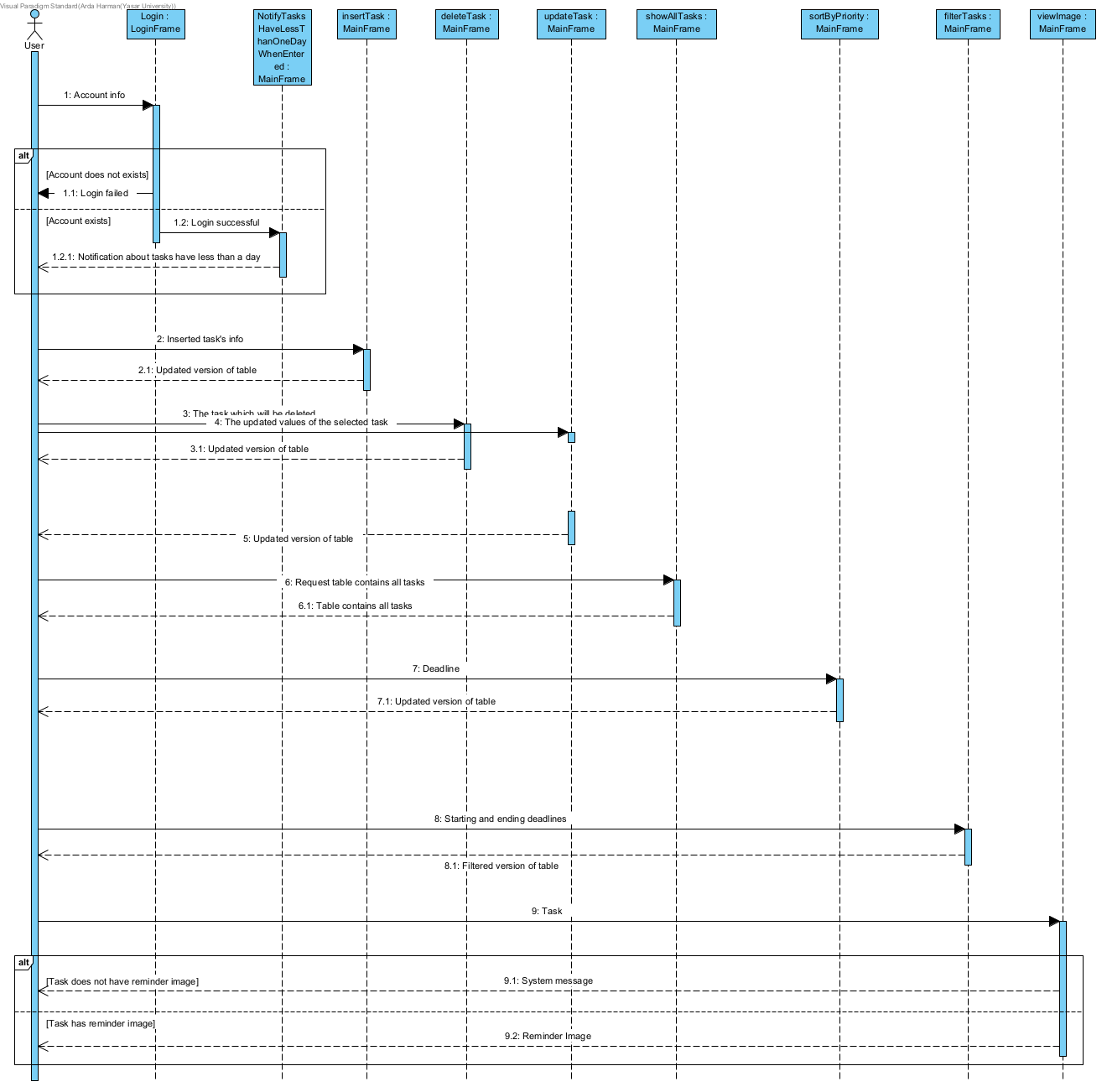
***Database – Disconnect from database:*** System disconnects from the database

## **Use Case Diagram**



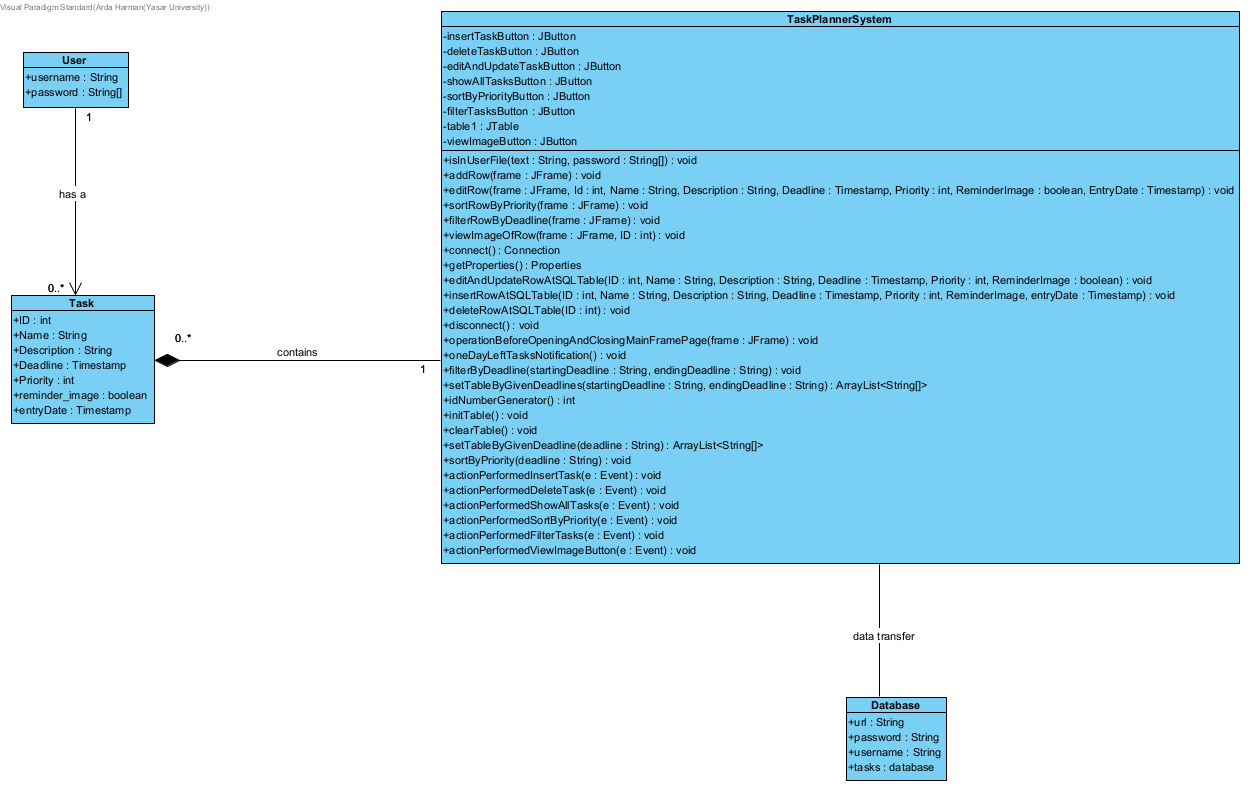
# Behavioral Models

## **Sequence Diagram**



# **Structural Models**

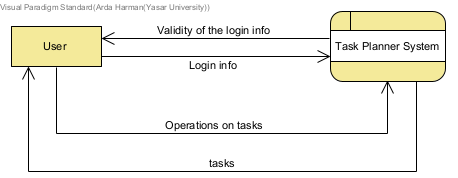
## **Class Diagram**



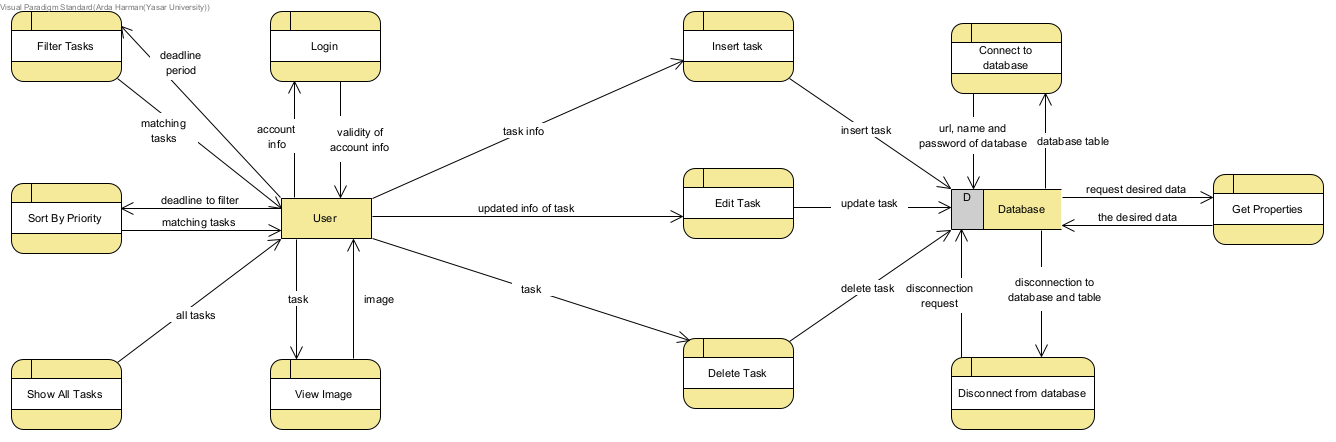
# **Process Modeling**

## **Data Flow Diagram (DFD)**

Context Diagram

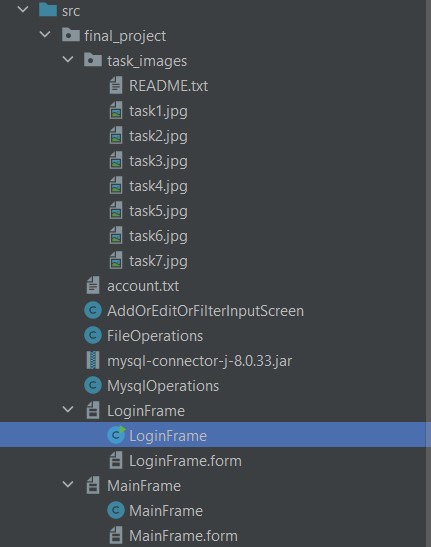


Level 0 DFD

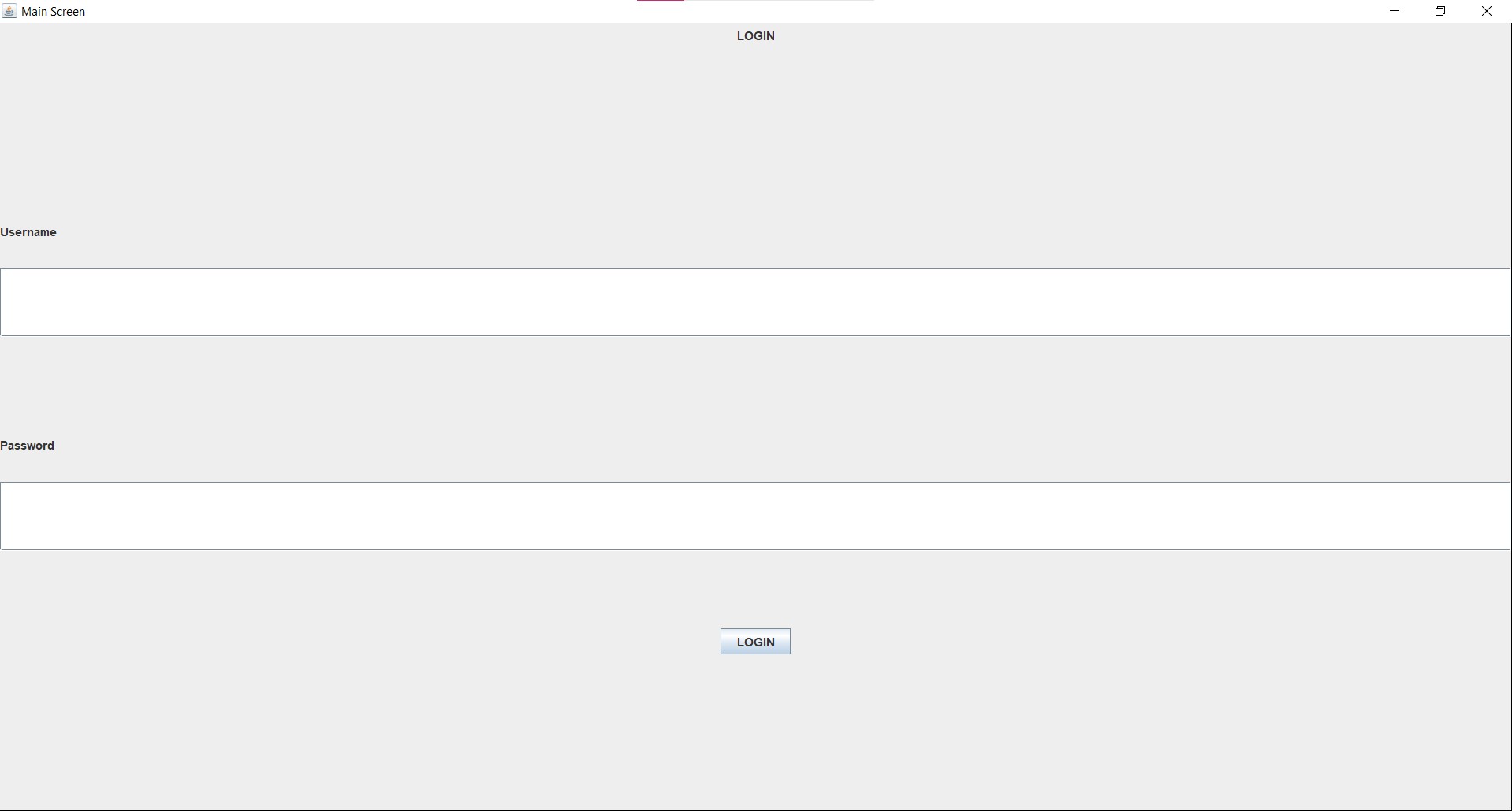


# **Final Version(s) of the Graphical User Interface(s)**

**MY FILES AND CLASSES**: This is just to demonstrate files and classes used at the project



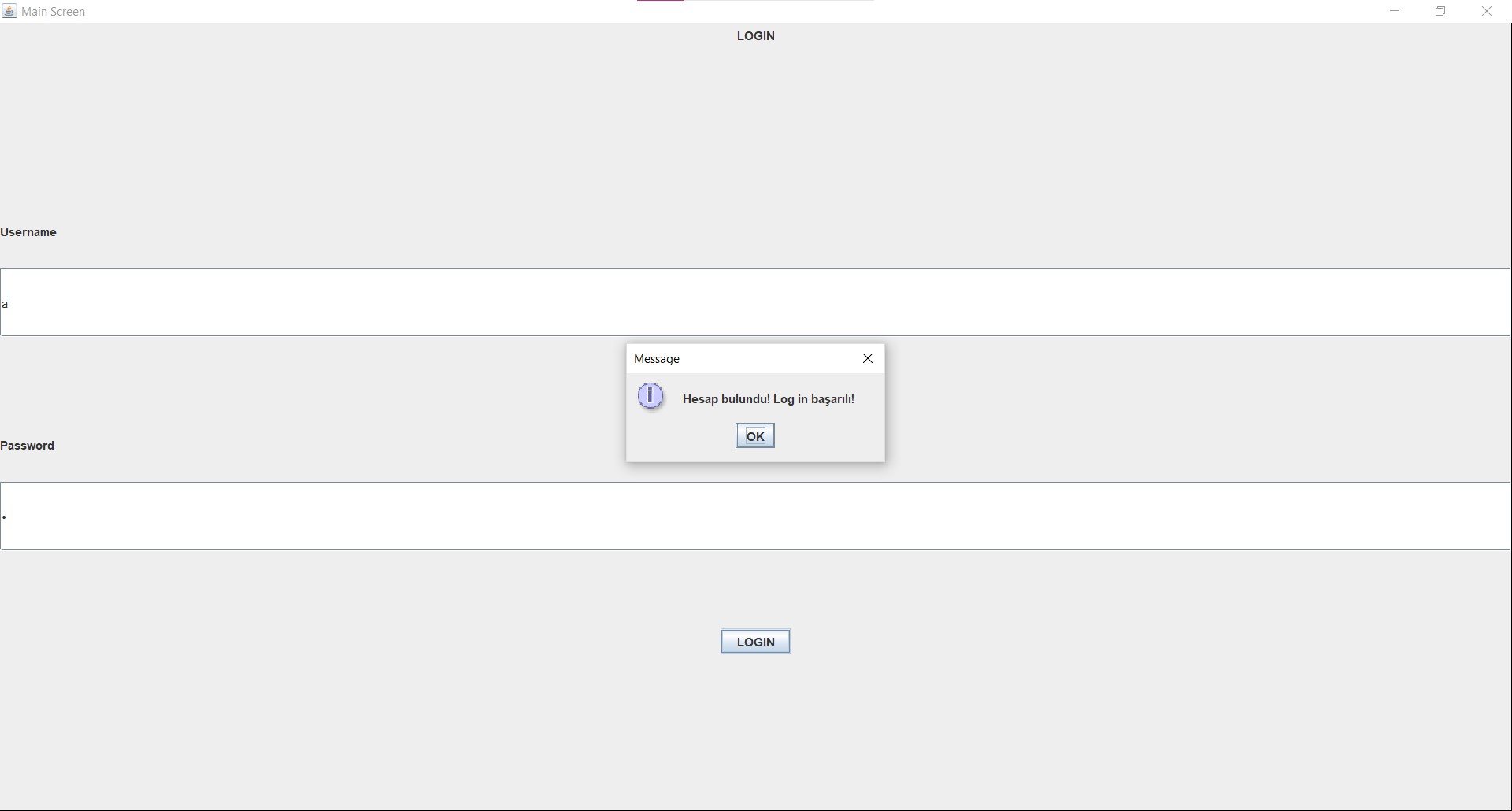
**LOGIN SCREEN**: User logs in at this screen



* ***INVALID ACCOUNT INFO ENTERED***

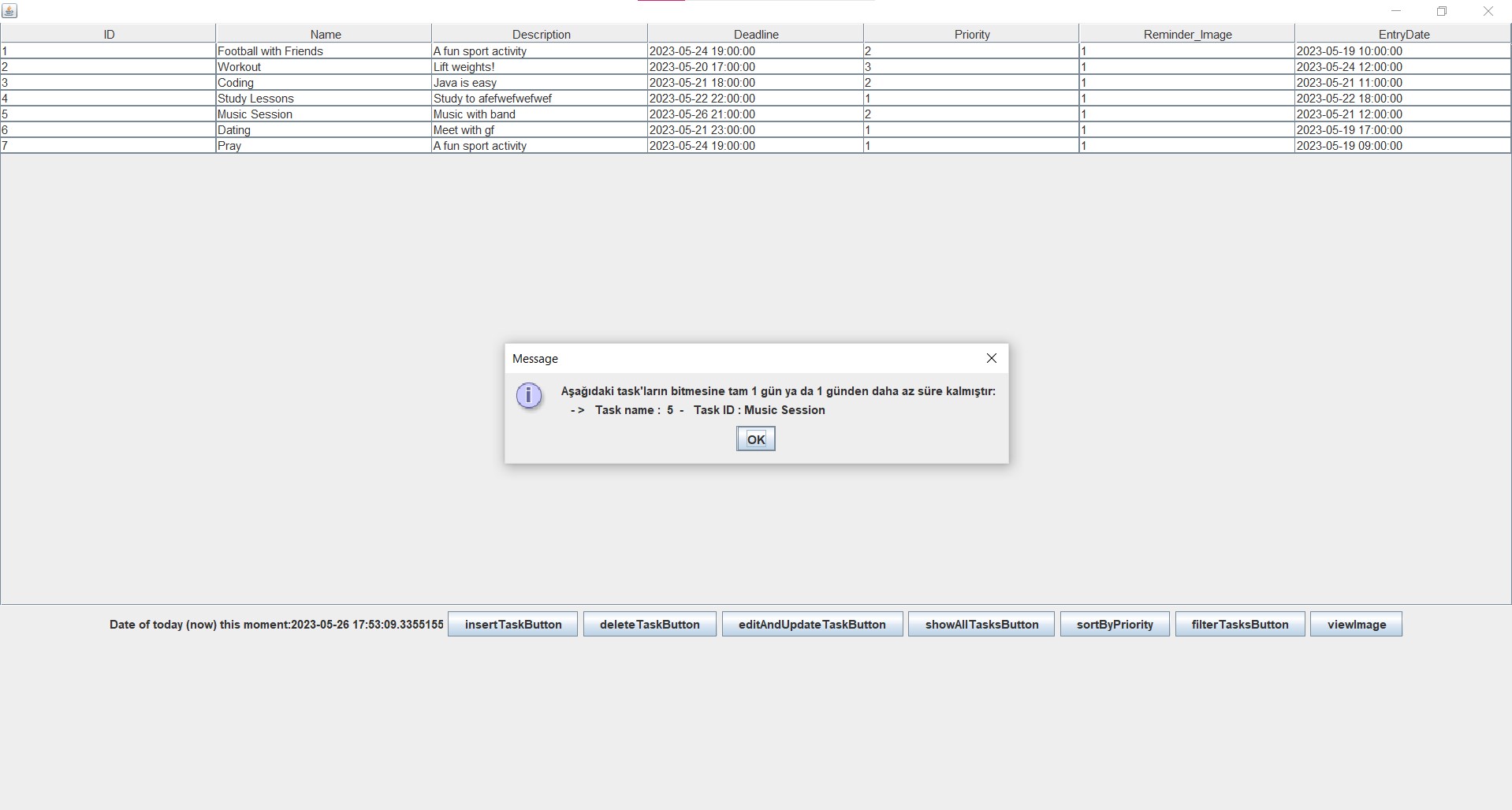


* ***VALID ACCOUNT INFO ENTERED***

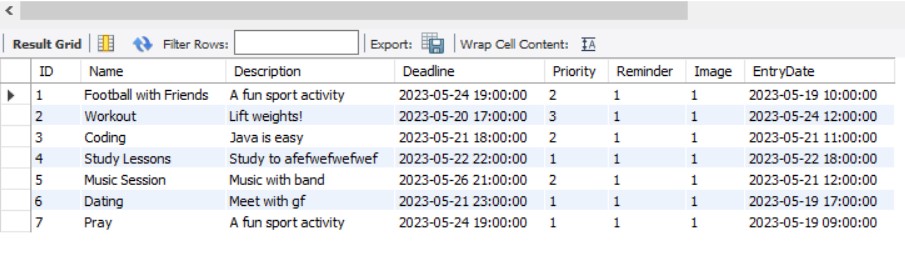


**MAIN SCREEN**: At this screen, tasks are shown to user and from this screen, user can do some operations on tasks

* ***JTABLE DISPLAYED TO THE USER***

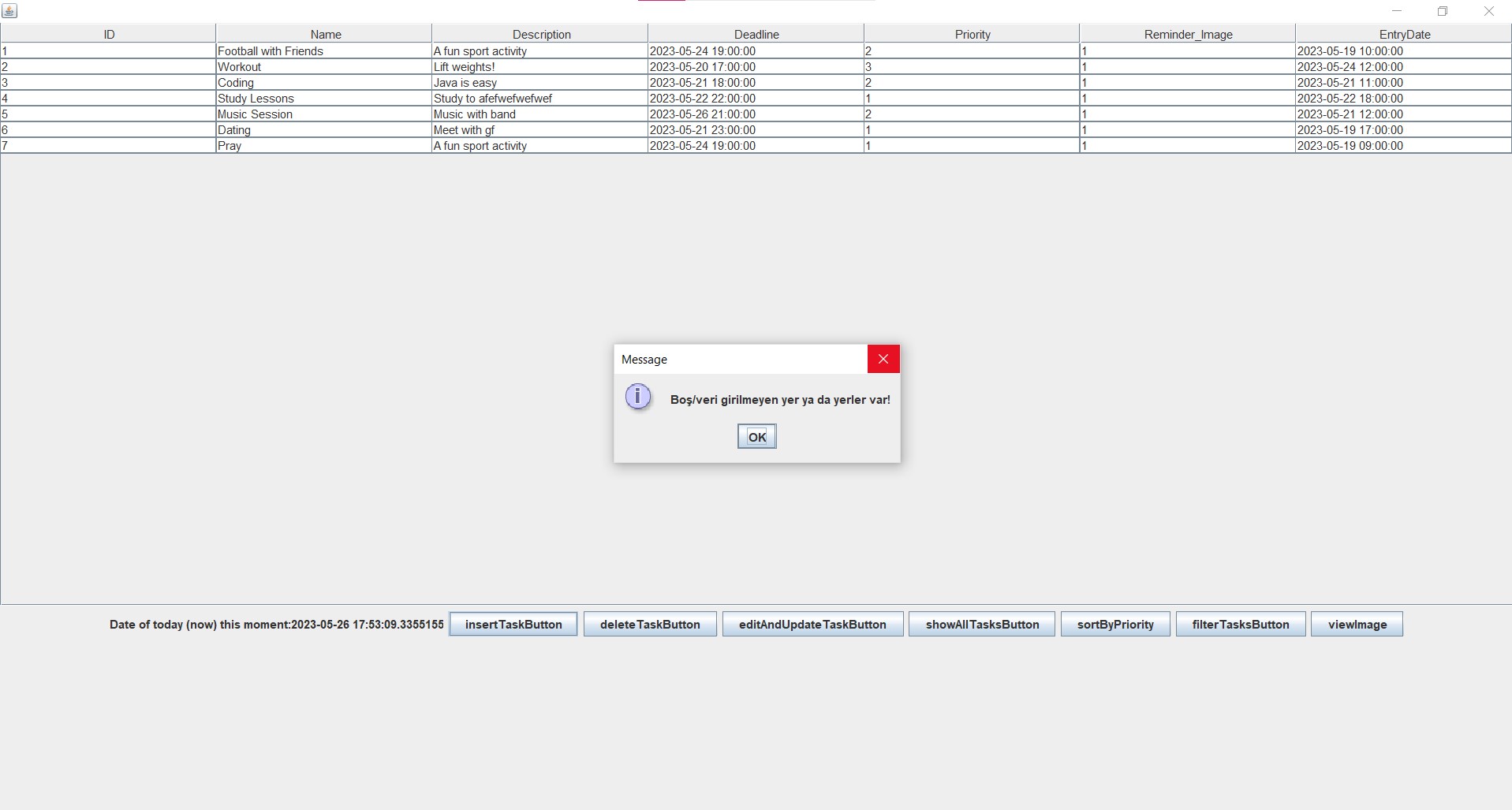
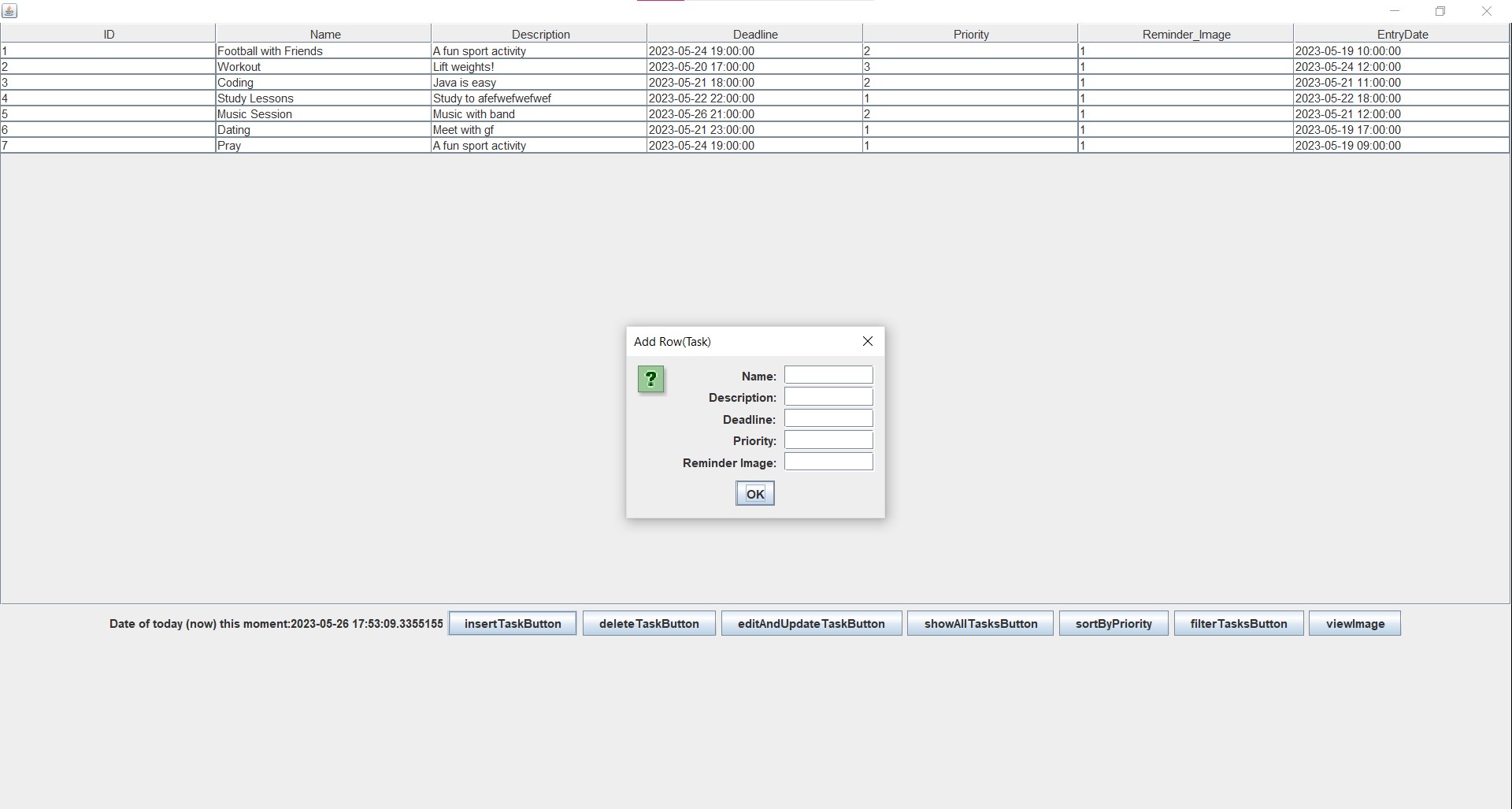


* ***DATABASE***

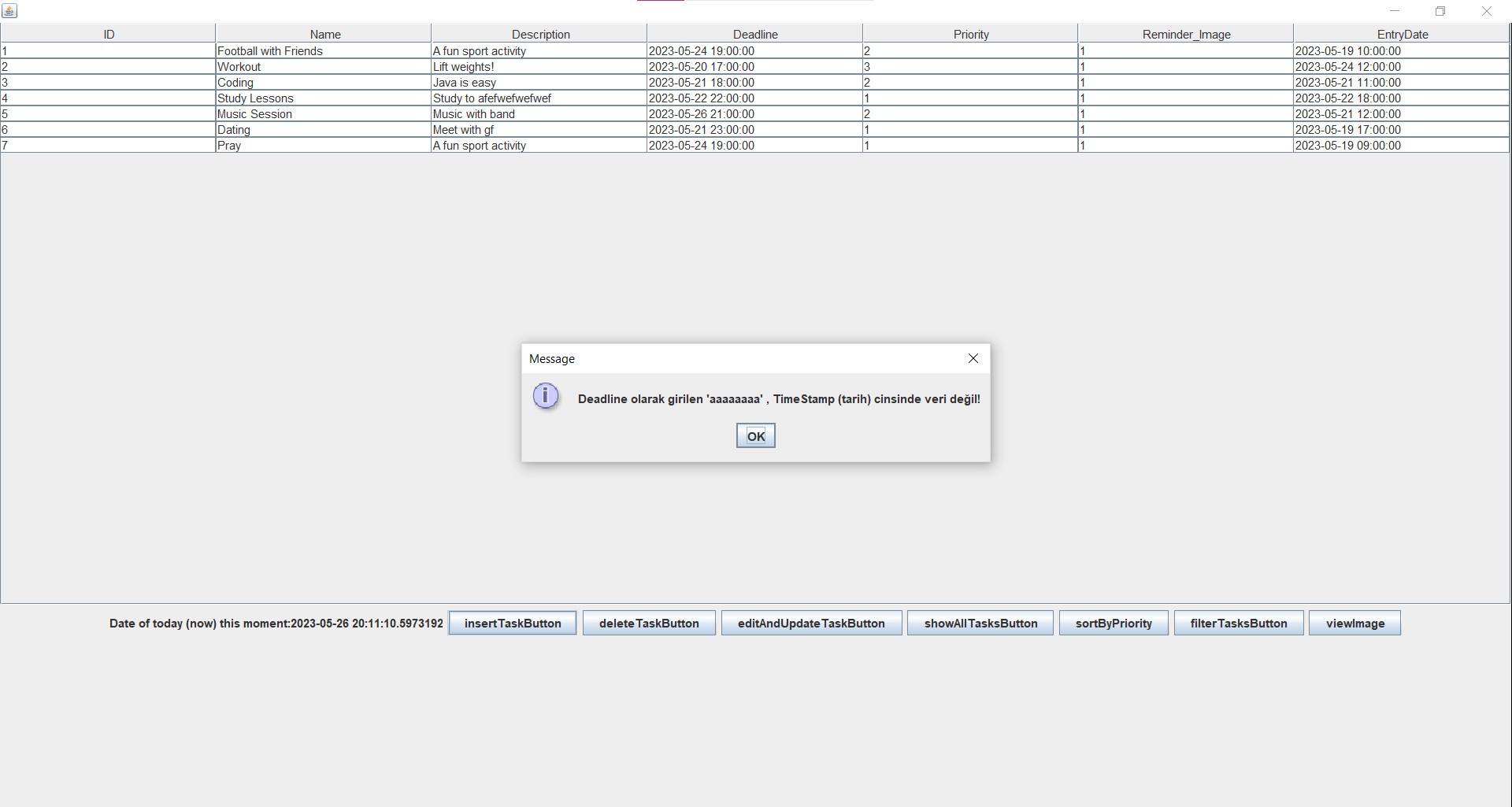
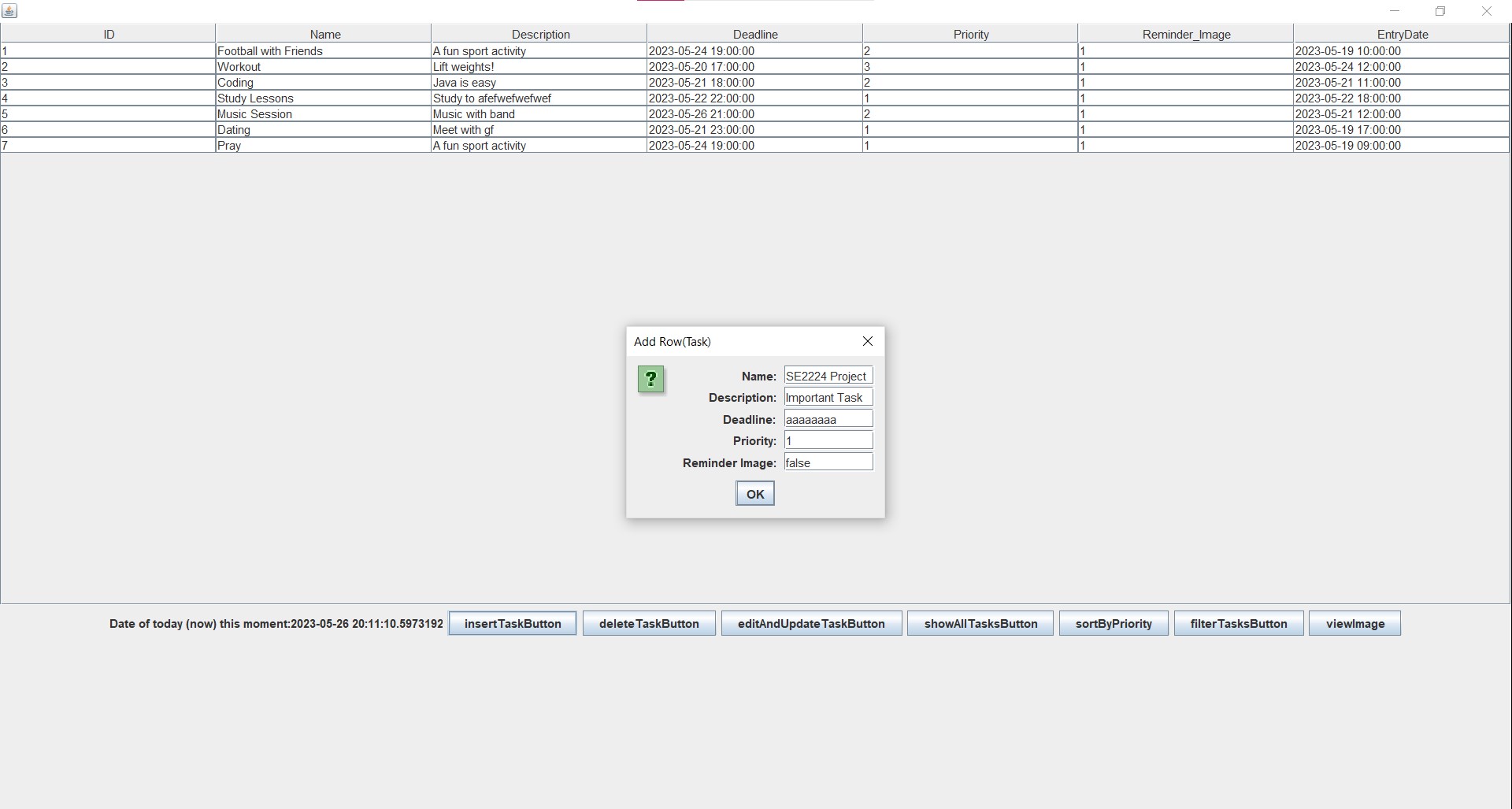


**INSERT TASK**: User can insert a new task to the table

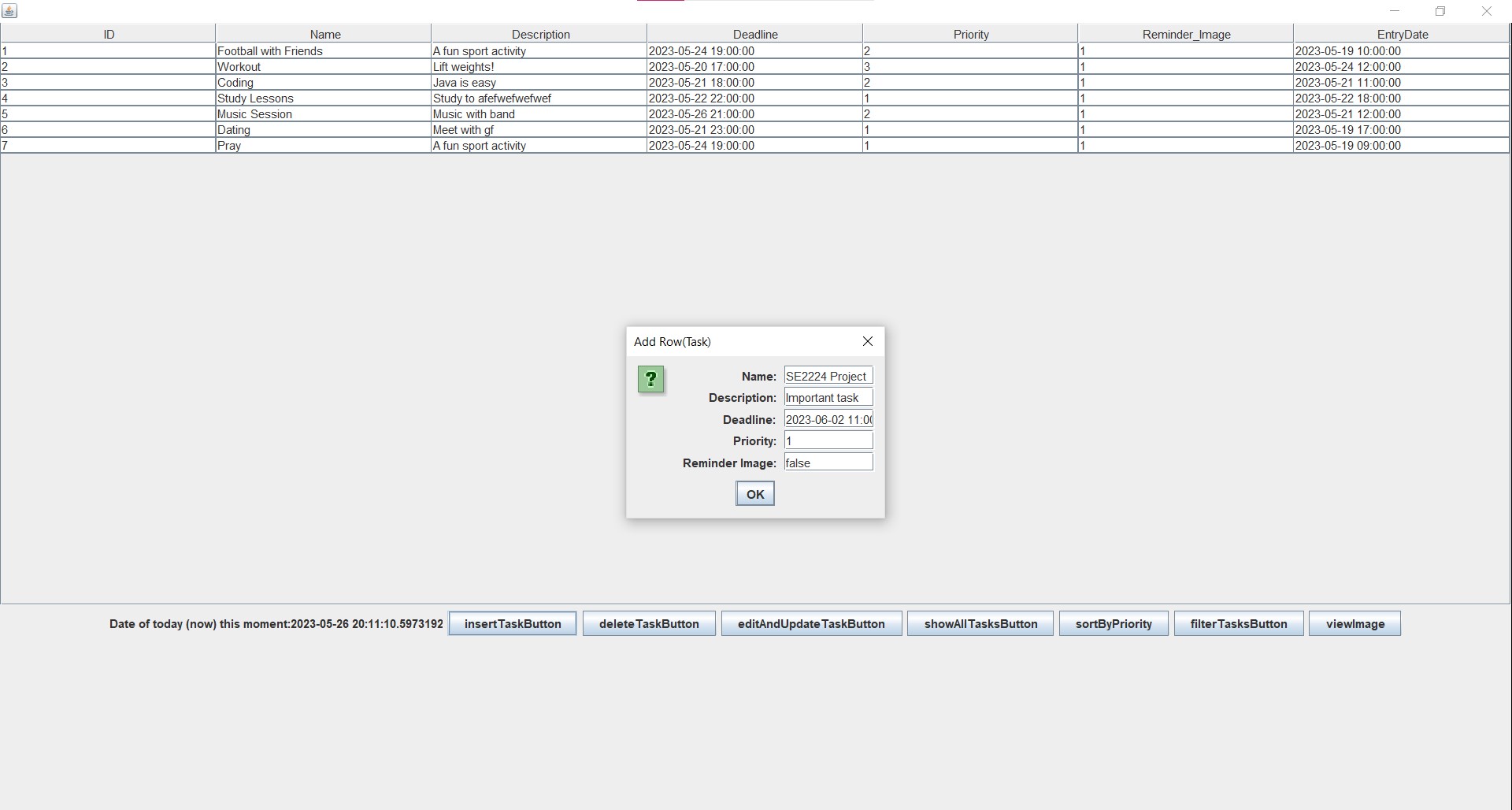
* ***ATTEMPTING TO INSERT TASK WHILE LEAVING SOME AREAS EMPTY***

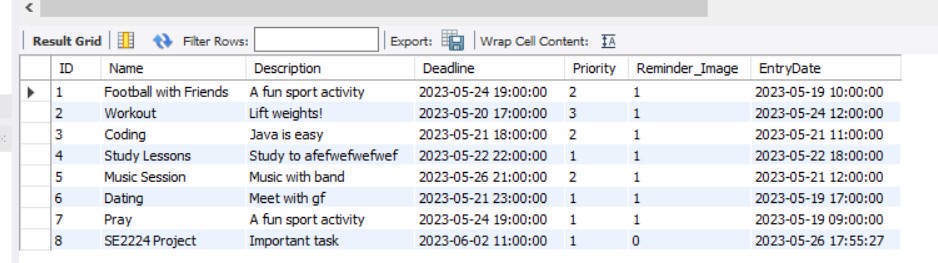


* ***ATTEMPTING TO INSERT A TASK WHILE ENTERING INCORRECT DEADLINE***



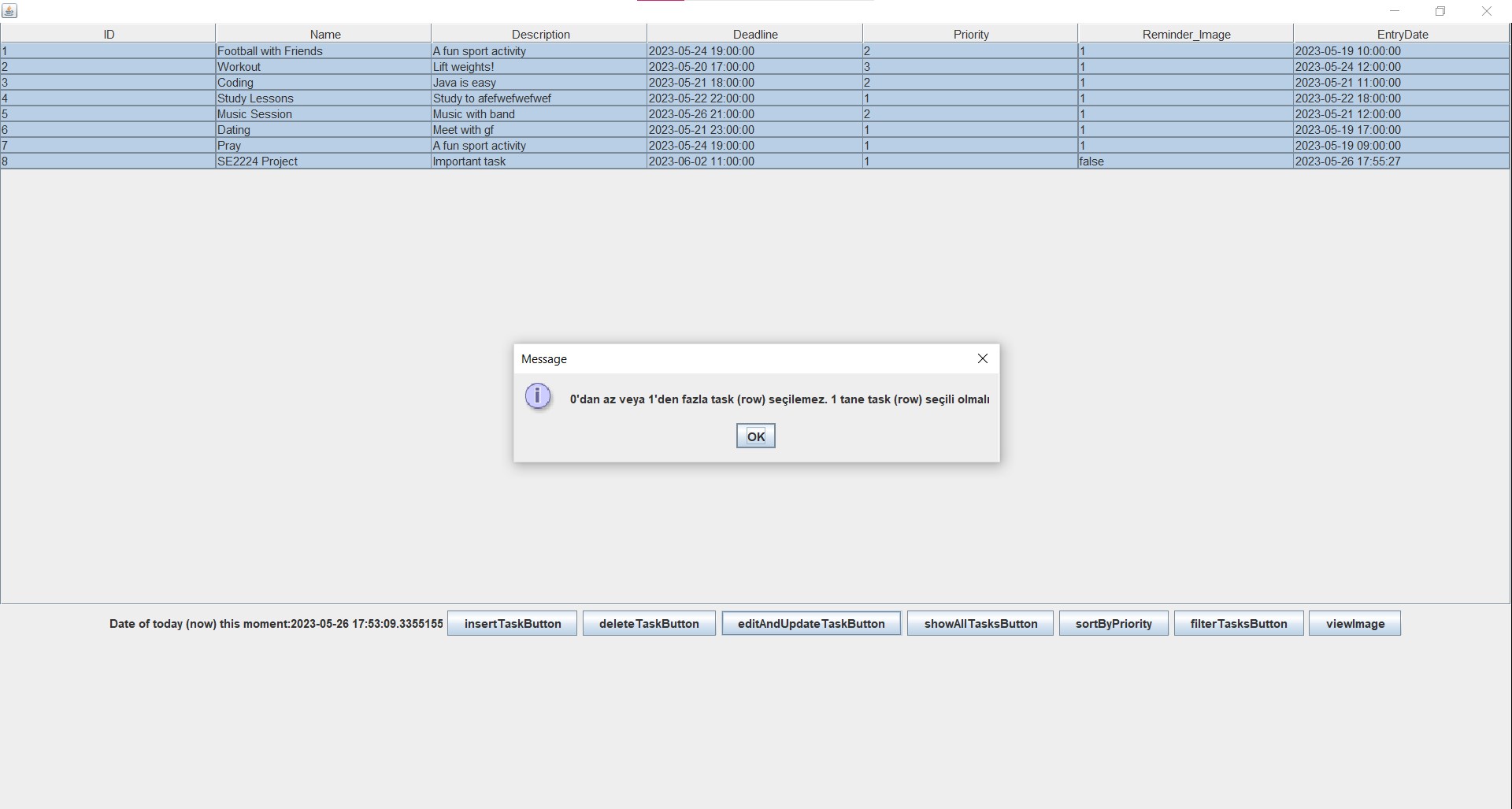
* ***INSERTING A TASK SUCCESSFULLY***



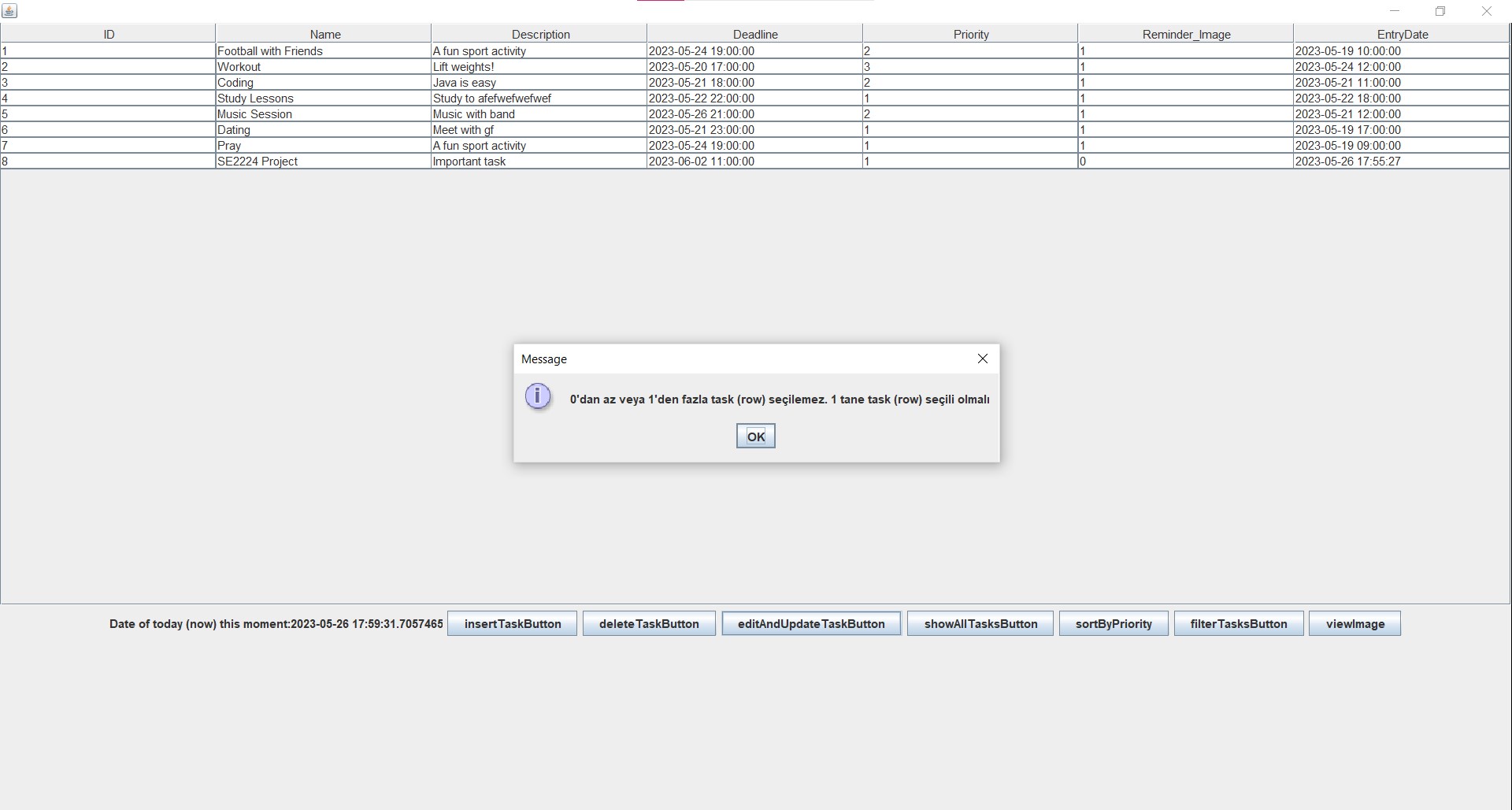


**EDITING AND UPDATING TASK:** User selects and edits and updates a task (user can’t edit the entry date and the ID of the task)

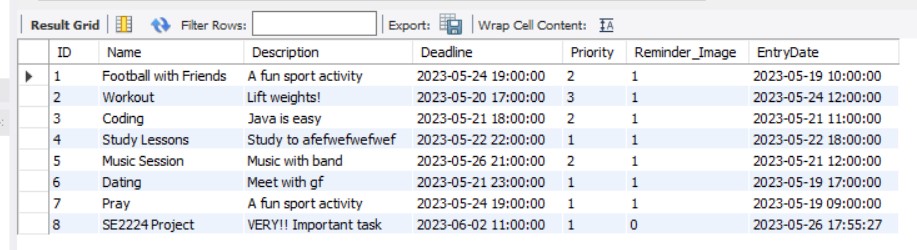
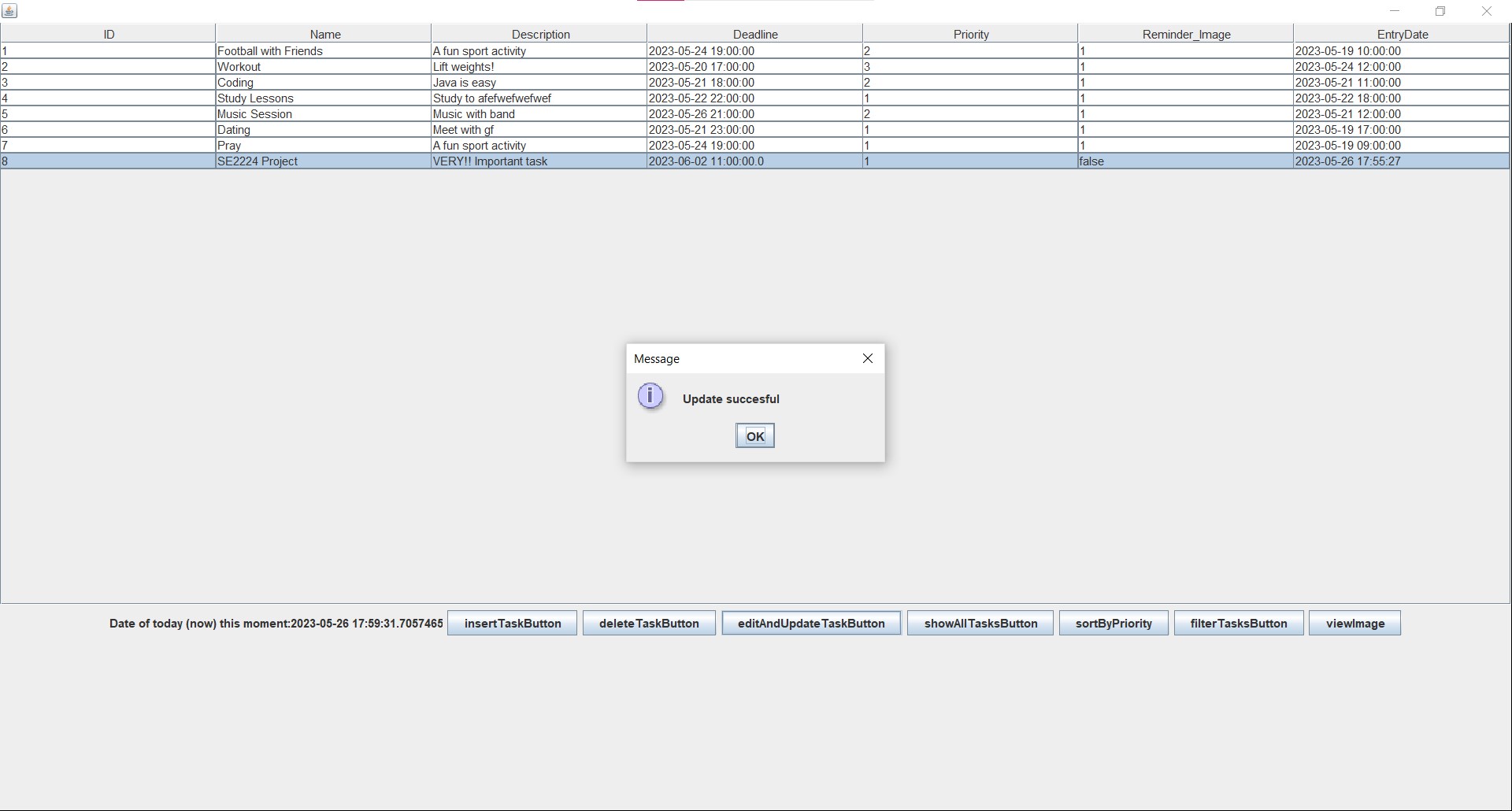
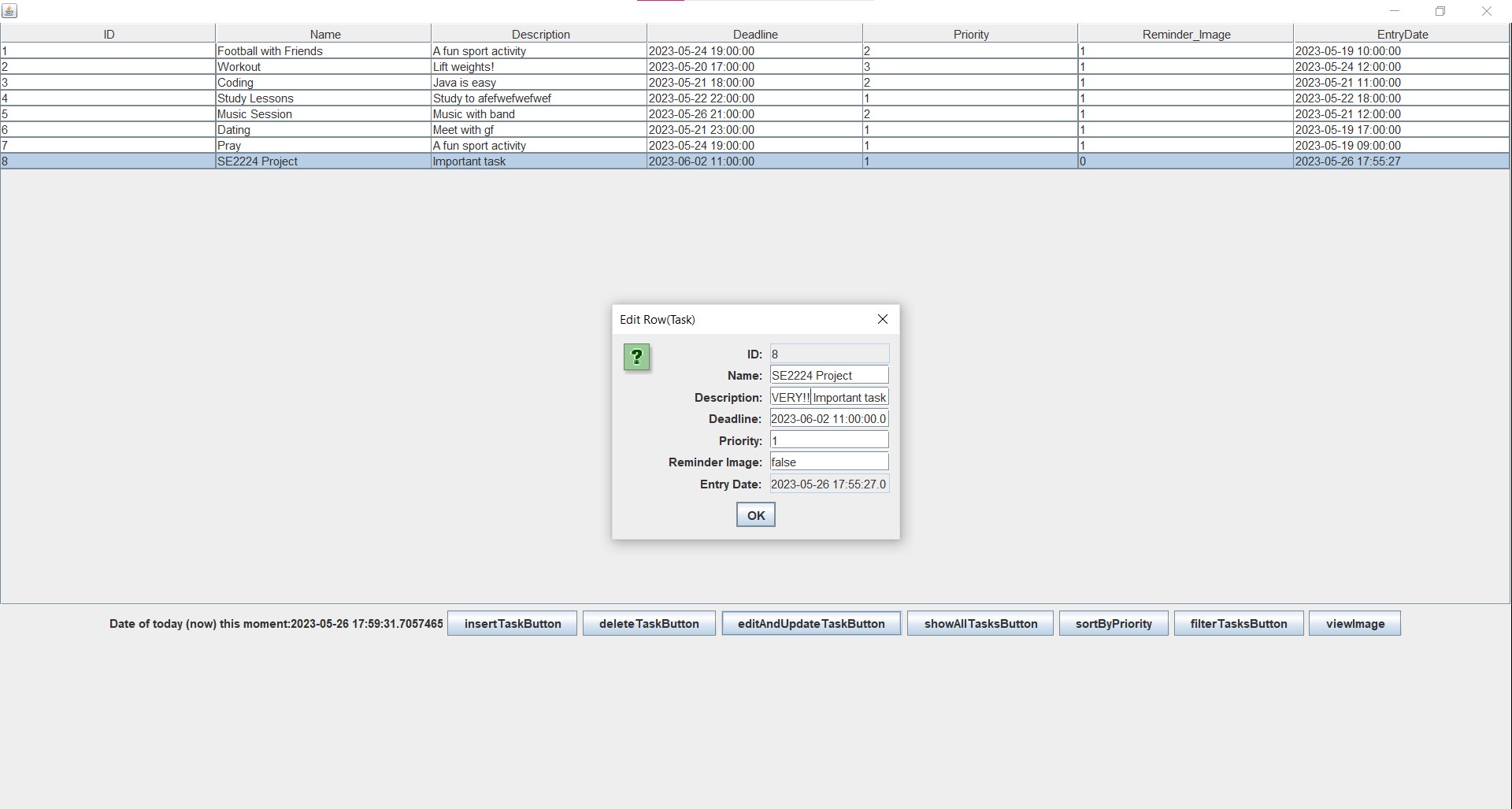
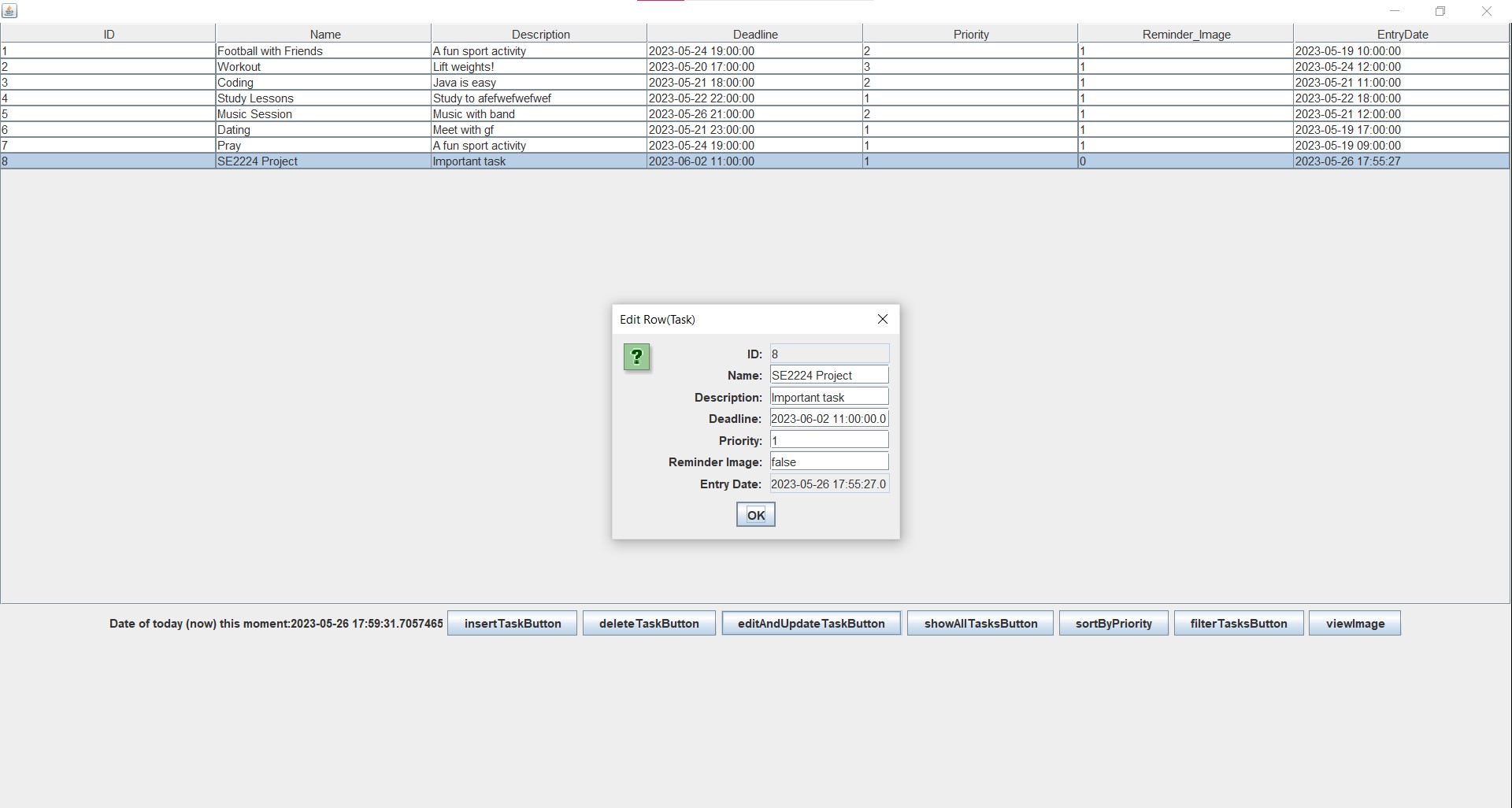
* ***ATTEMPTING TO SELECT MULTIPLE TASKS TO EDIT***



* ***ATTEMPTING TO NOT SELECT ANY TASKS TO EDIT***

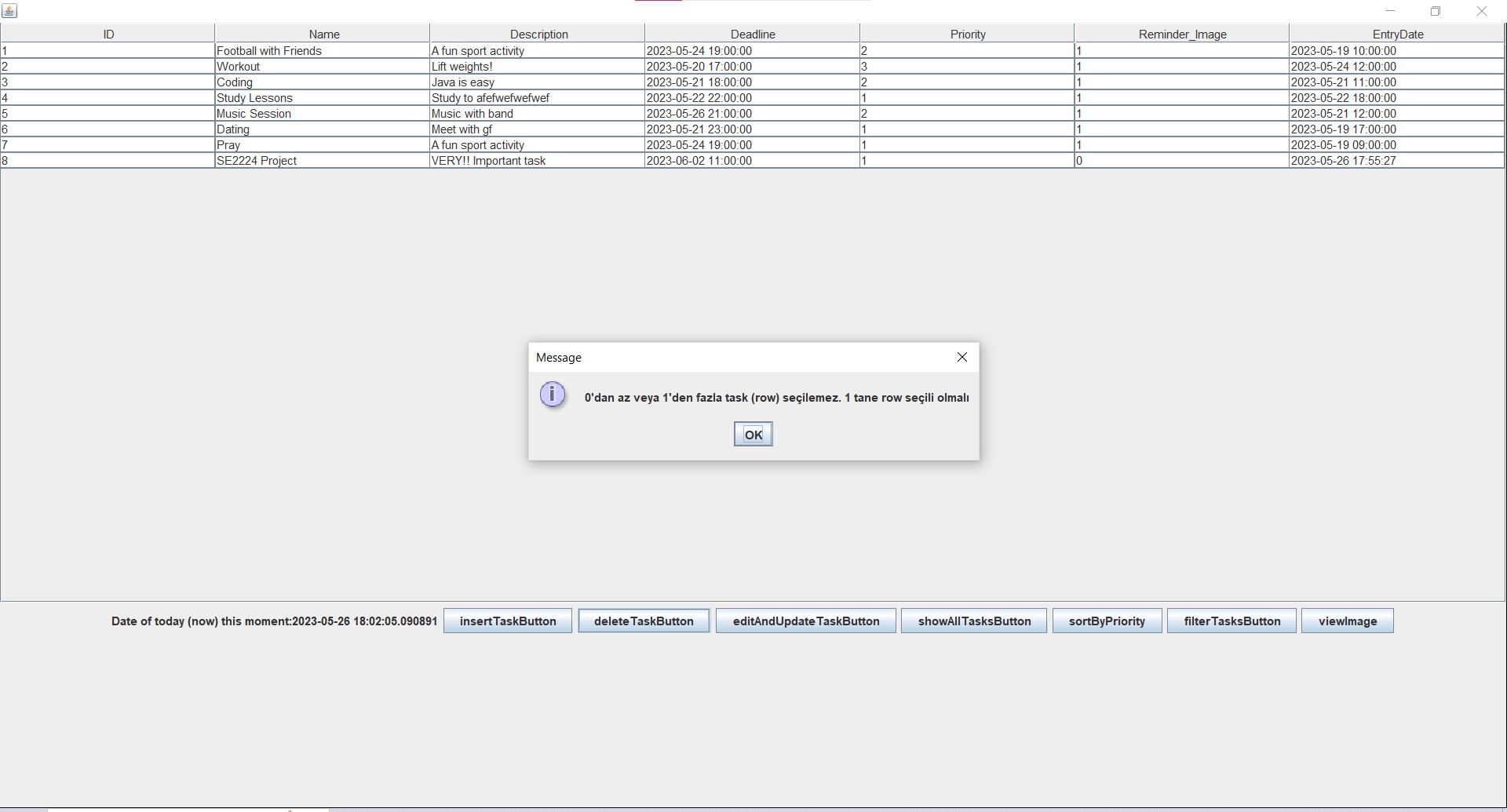


* ***SUCCESSFULLY EDITING AND UPDATING TASK***

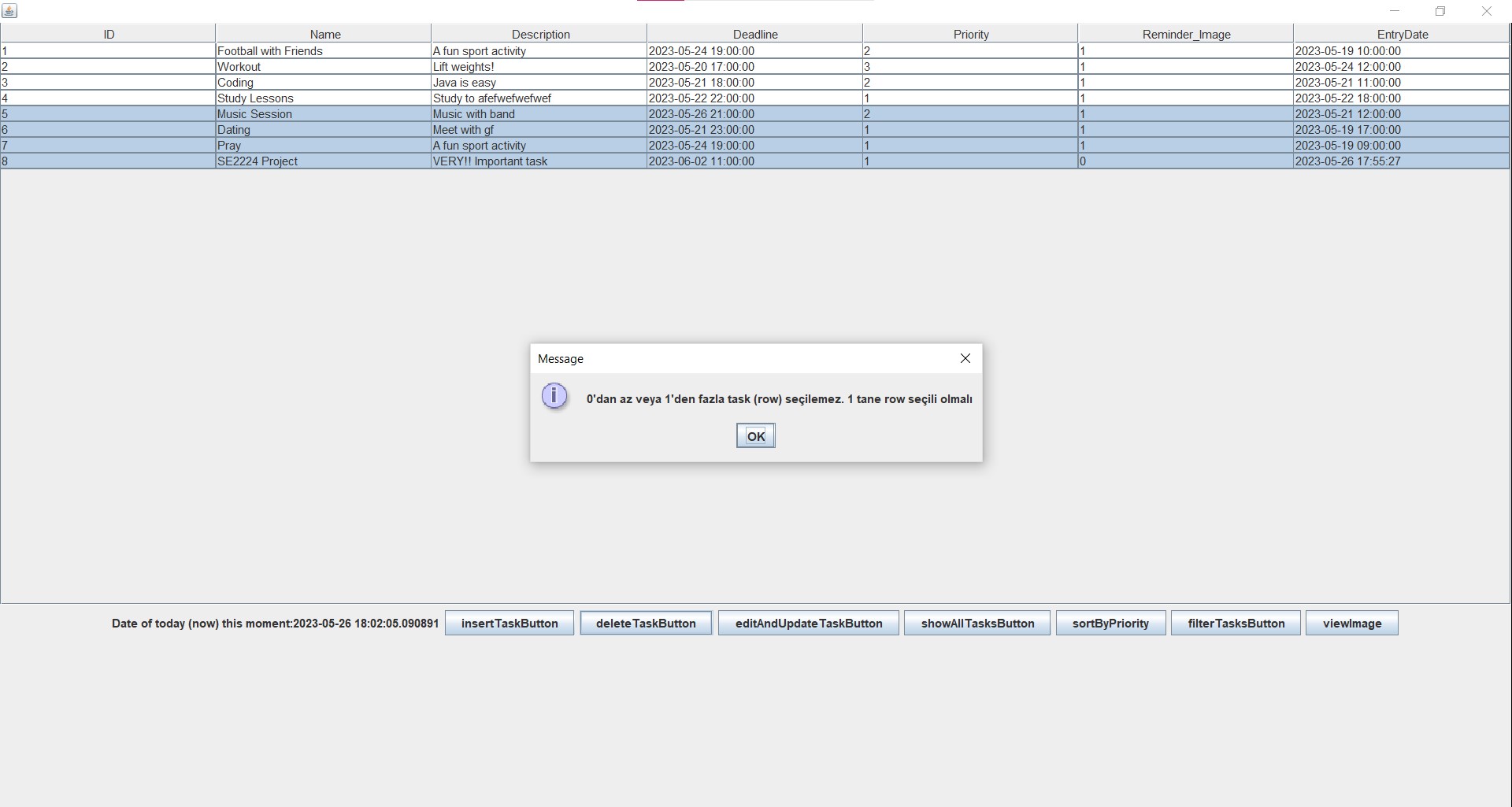


**DELETING TASKS**: User can delete a task from table that he/she wants

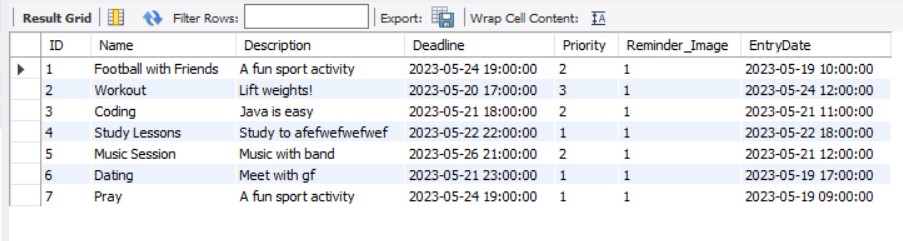
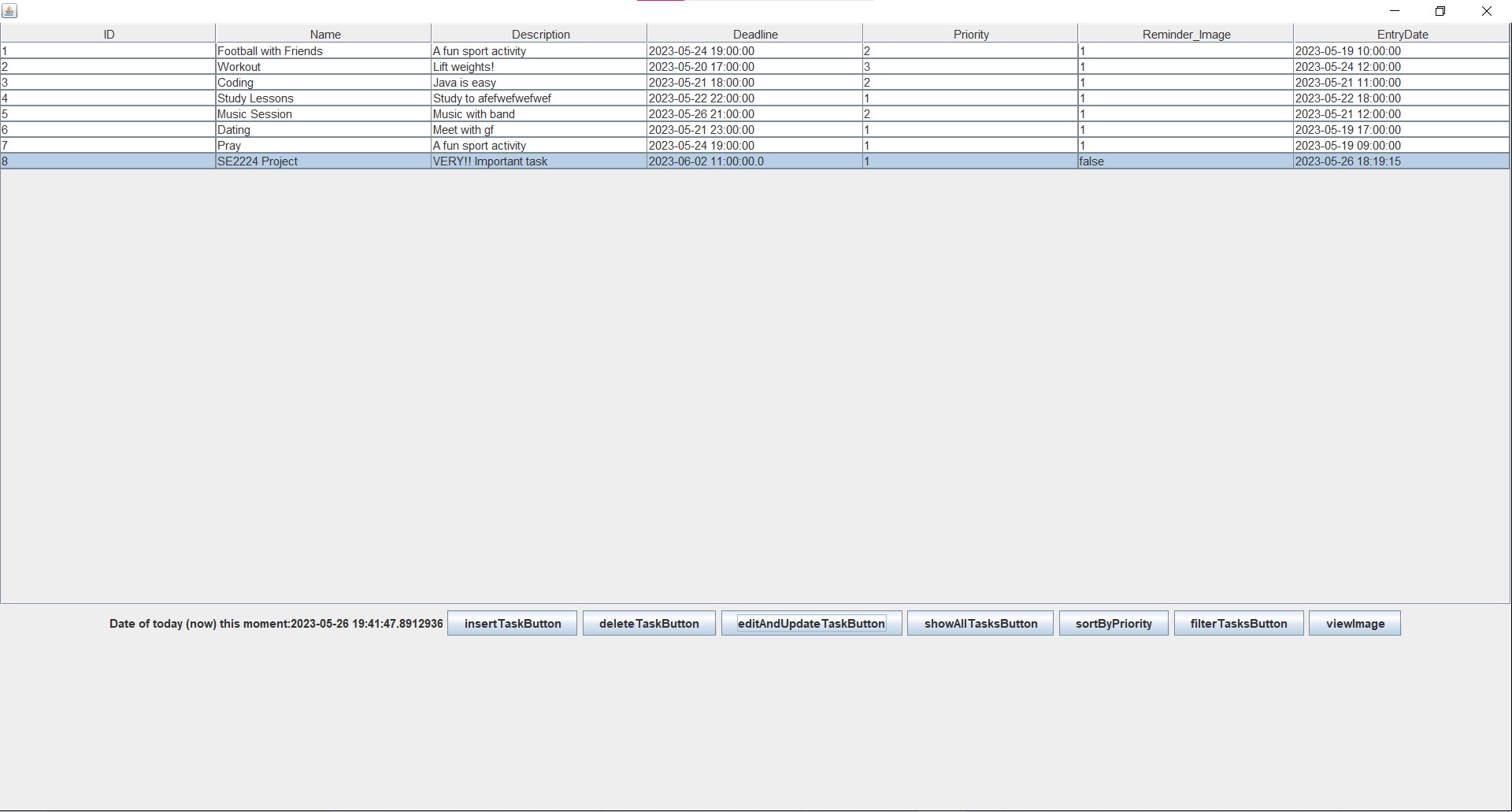
* ***ATTEMPTING TO DELETE TASK WITHOUT SELECTING ANY TASKS***



* ***ATTEMP TO DELETE TASK WHILE SELECTING MULTIPLE ROWS***

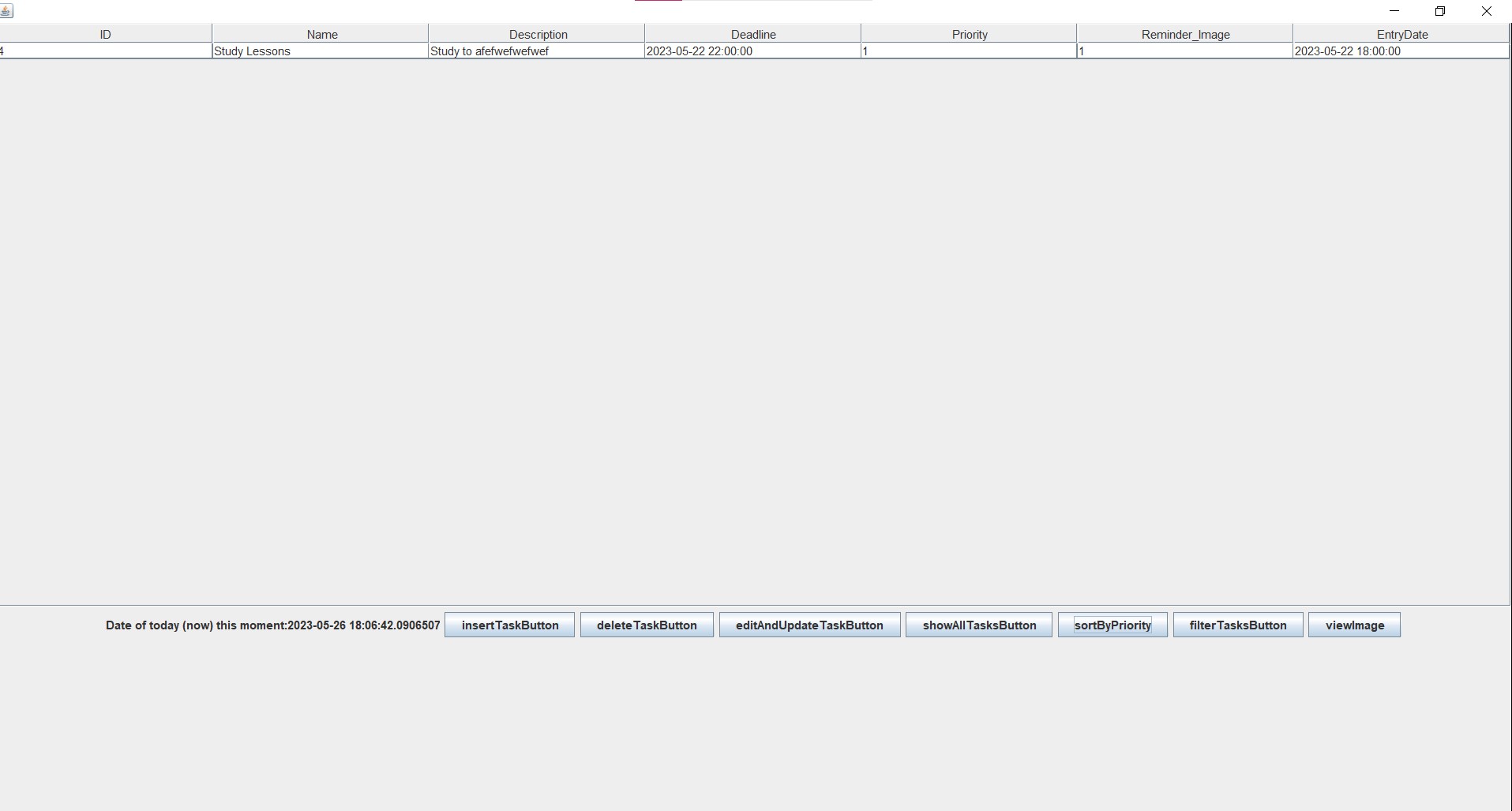
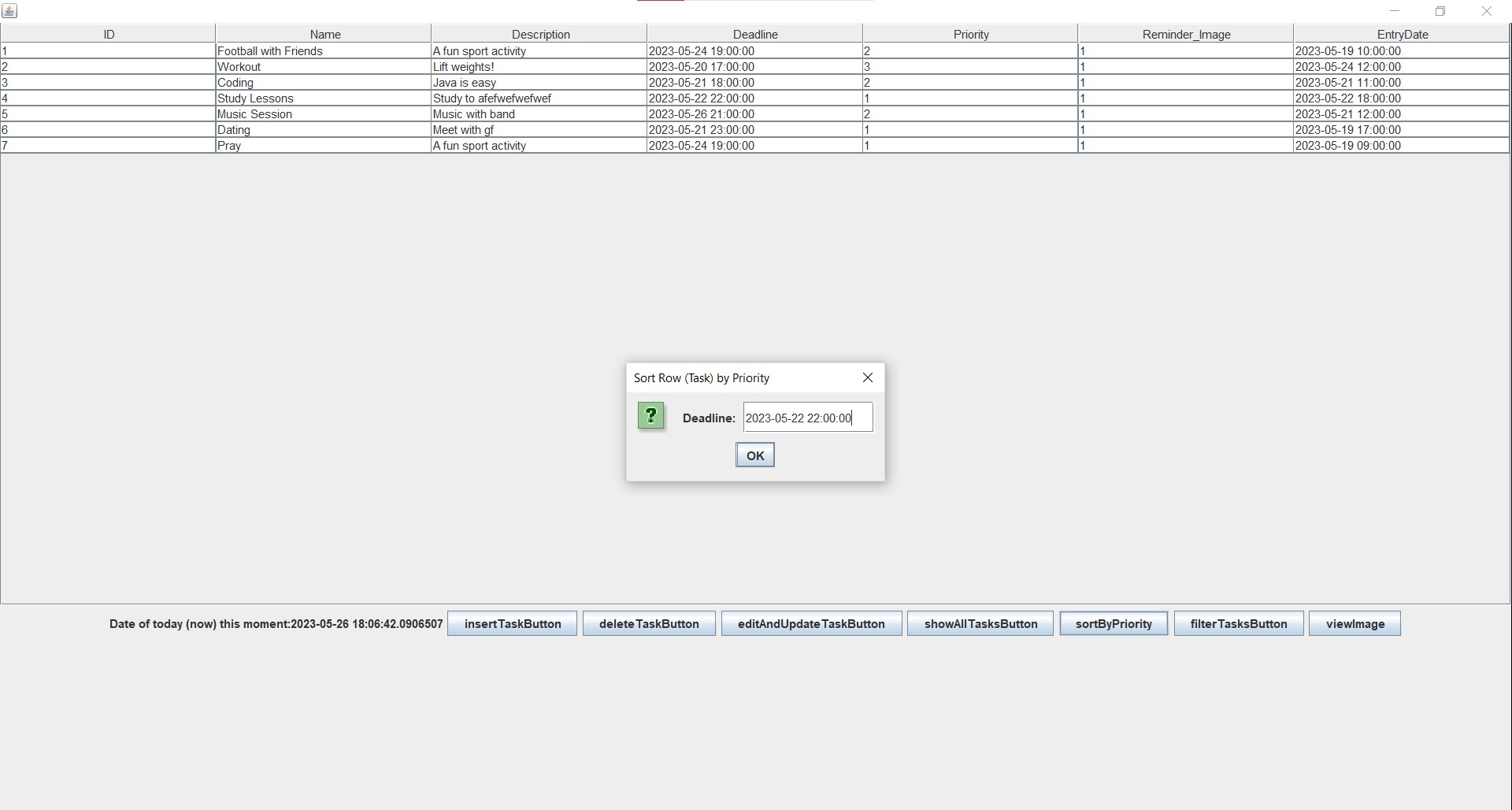


* ***DELETE TASK SUCCESSFULLY***

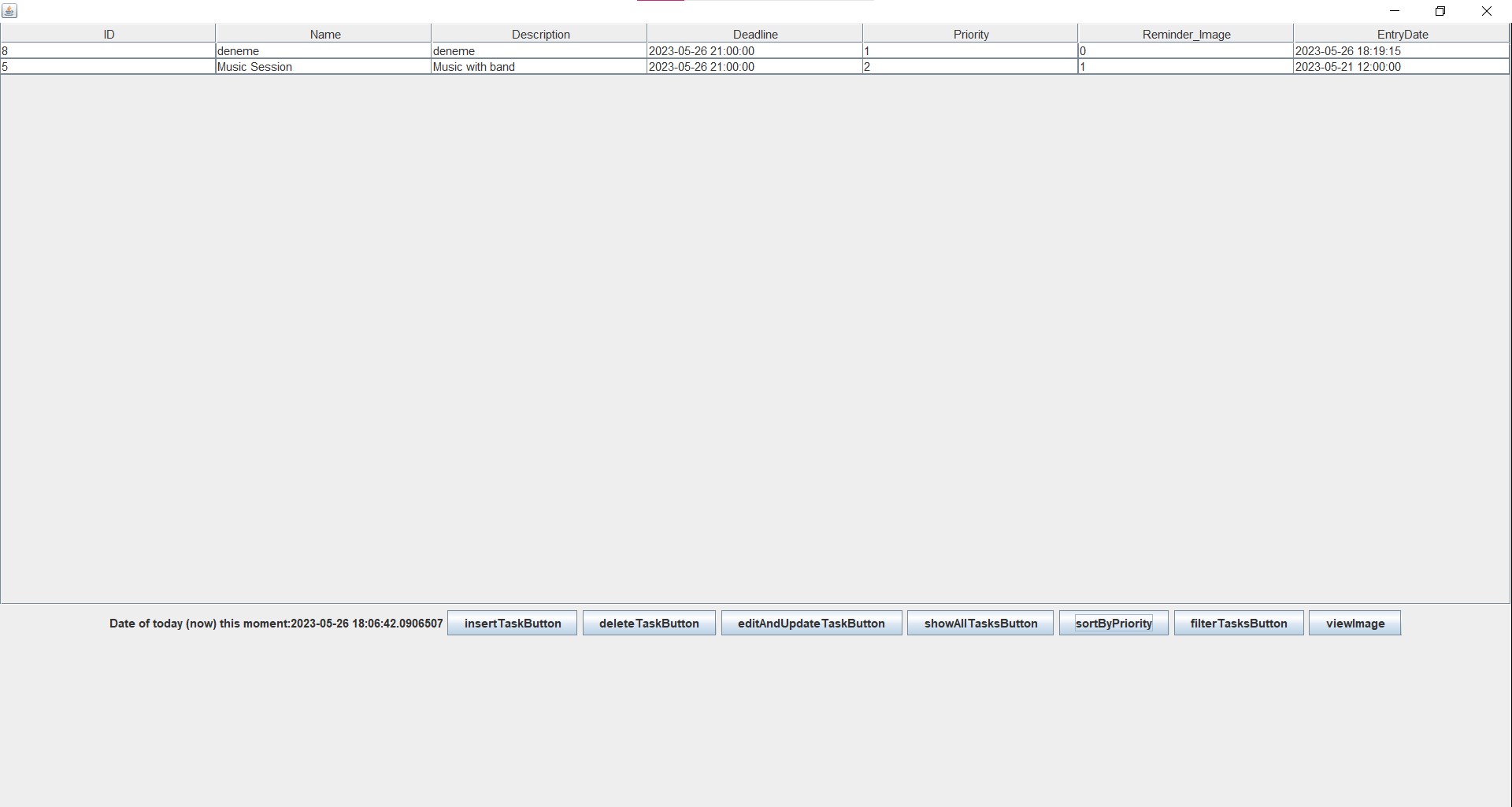
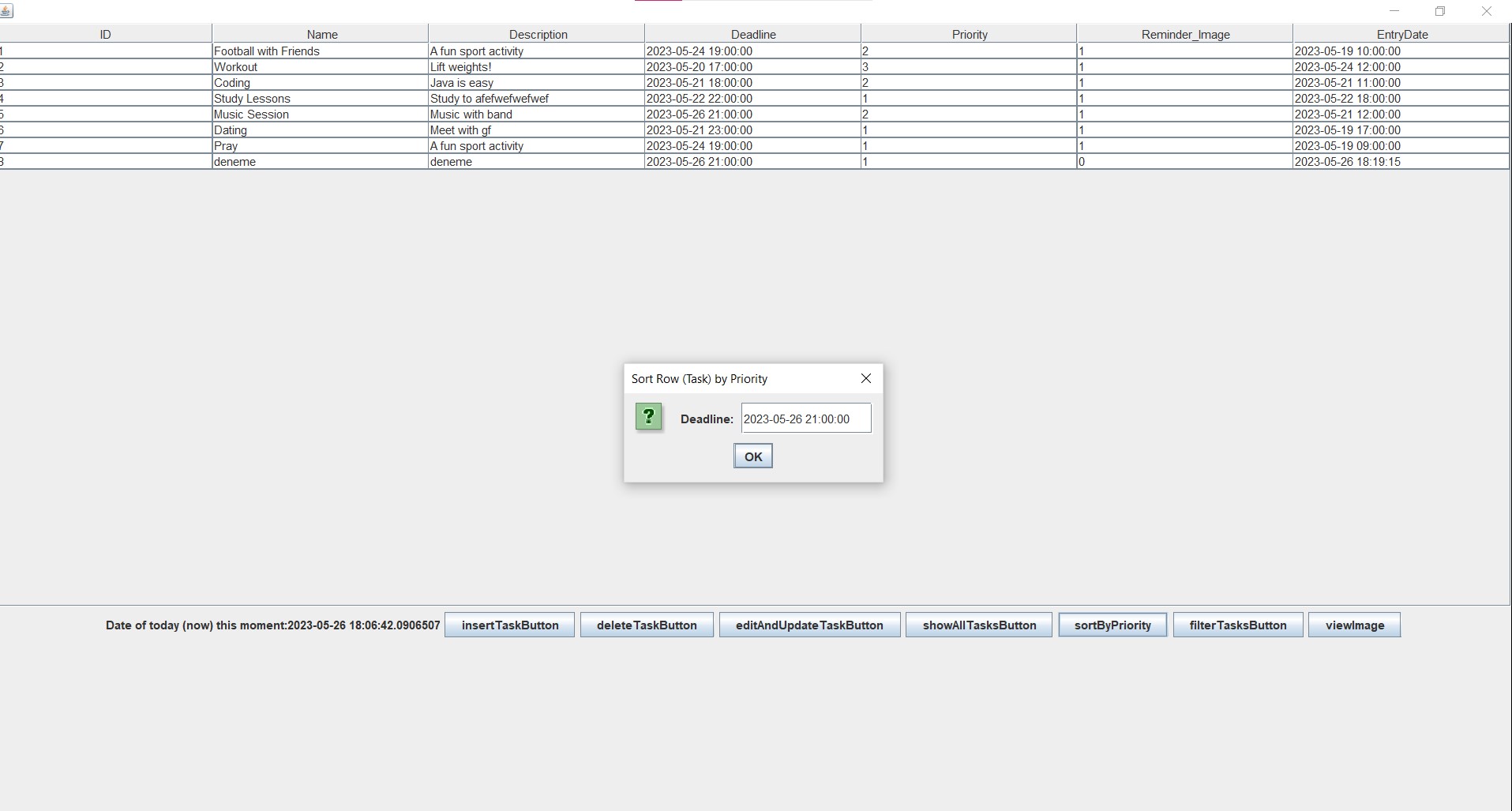


**Order tasks which has the same deadline sorted by their priority**: User enters a deadline to filter tasks and these filtered tasks are sorted by their priority

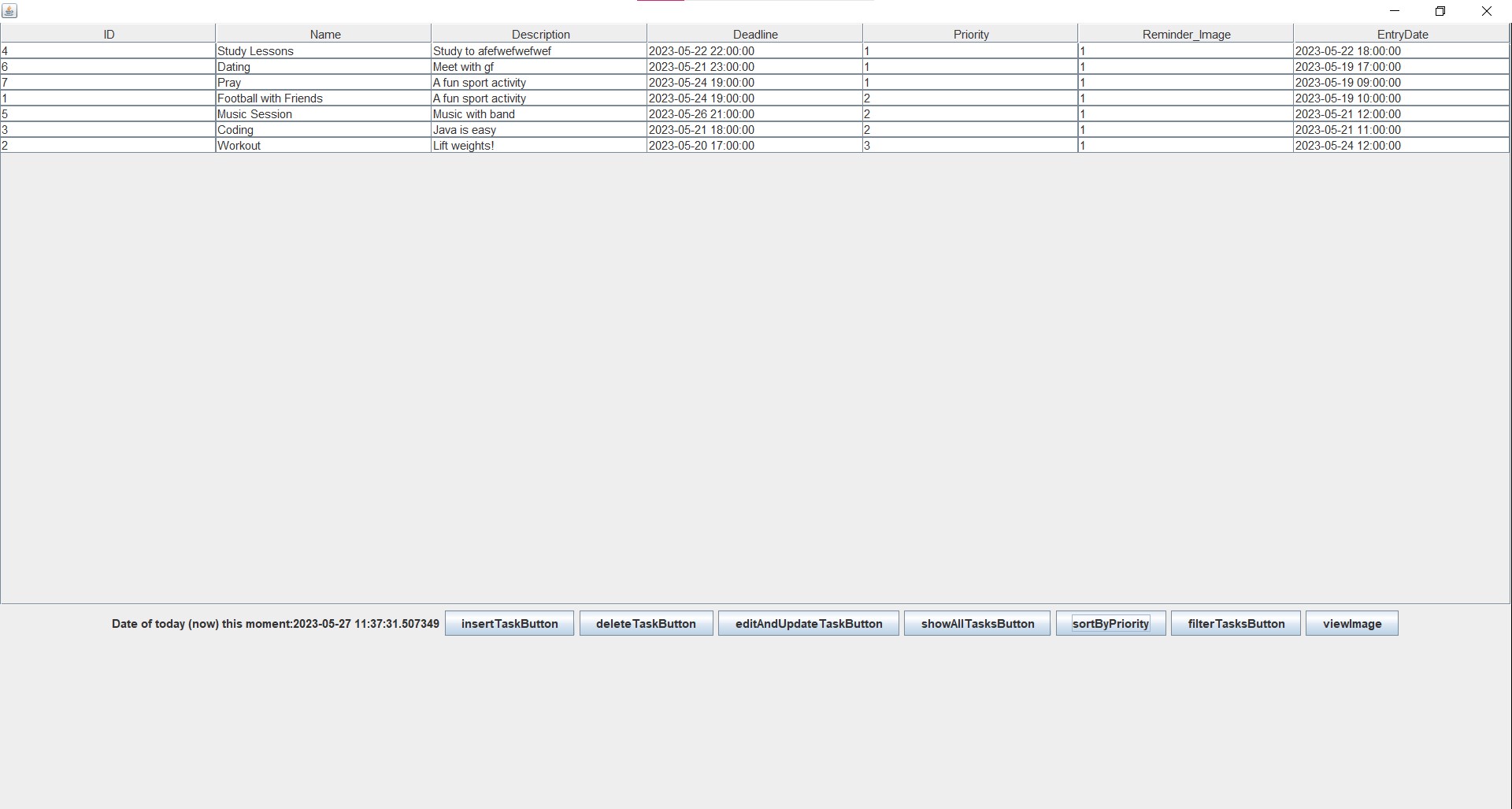
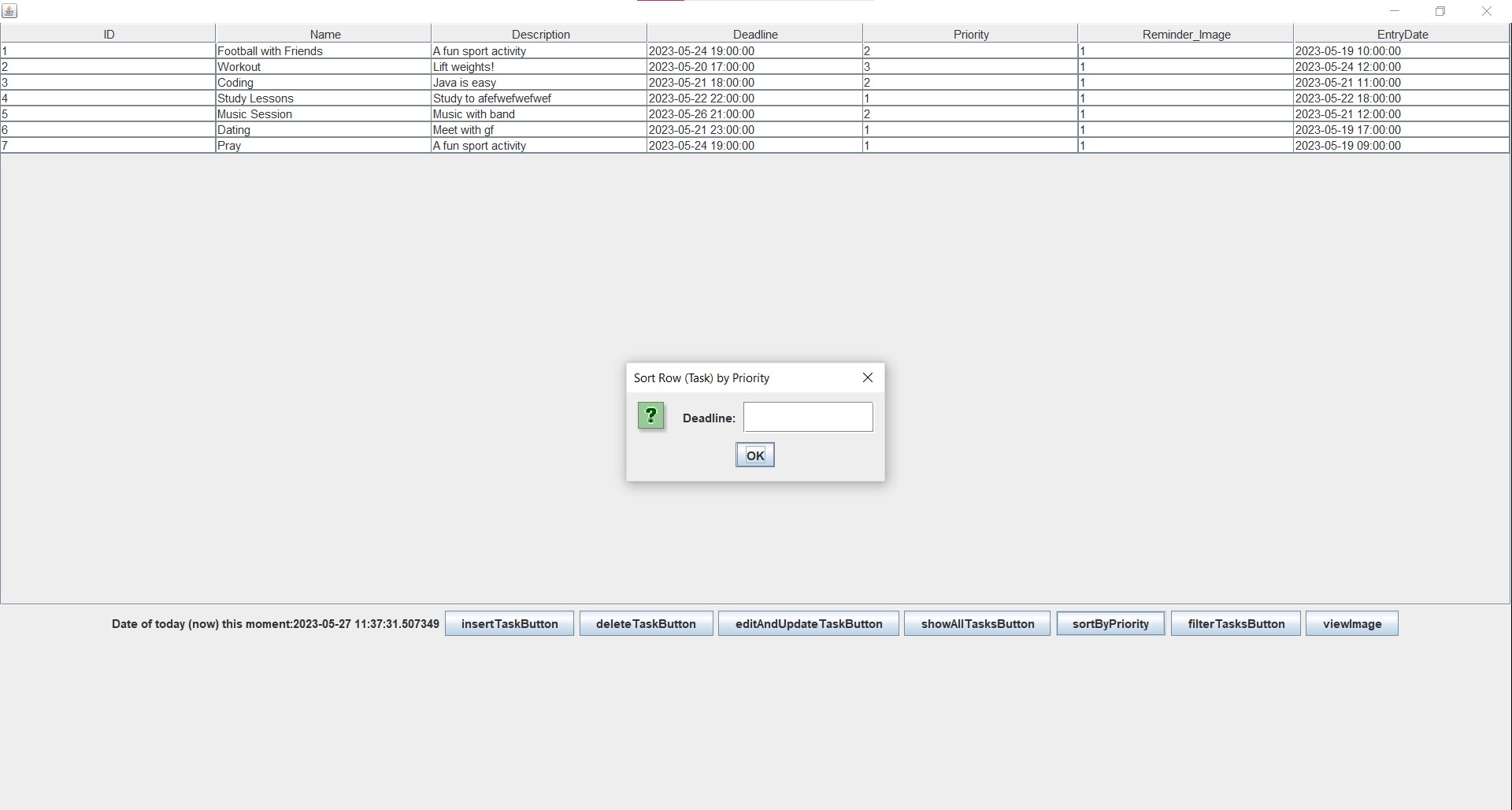
* ***SUCCESSFULLY ORDERING TASKS WHICH HAS SAME PRIORITY BY DEADLINE***
  + ***Example 1 – Proof that filtering according to deadline works***



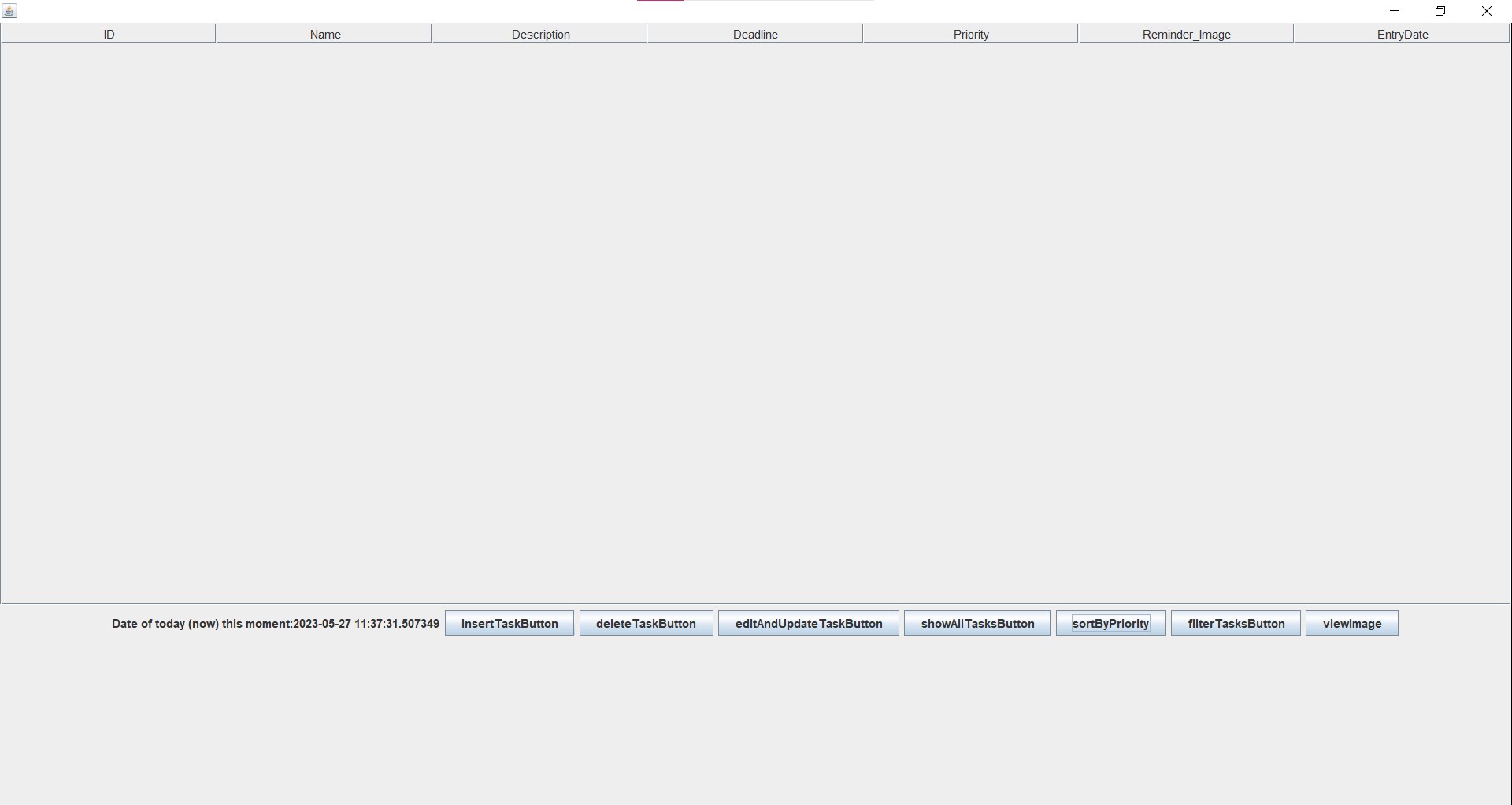
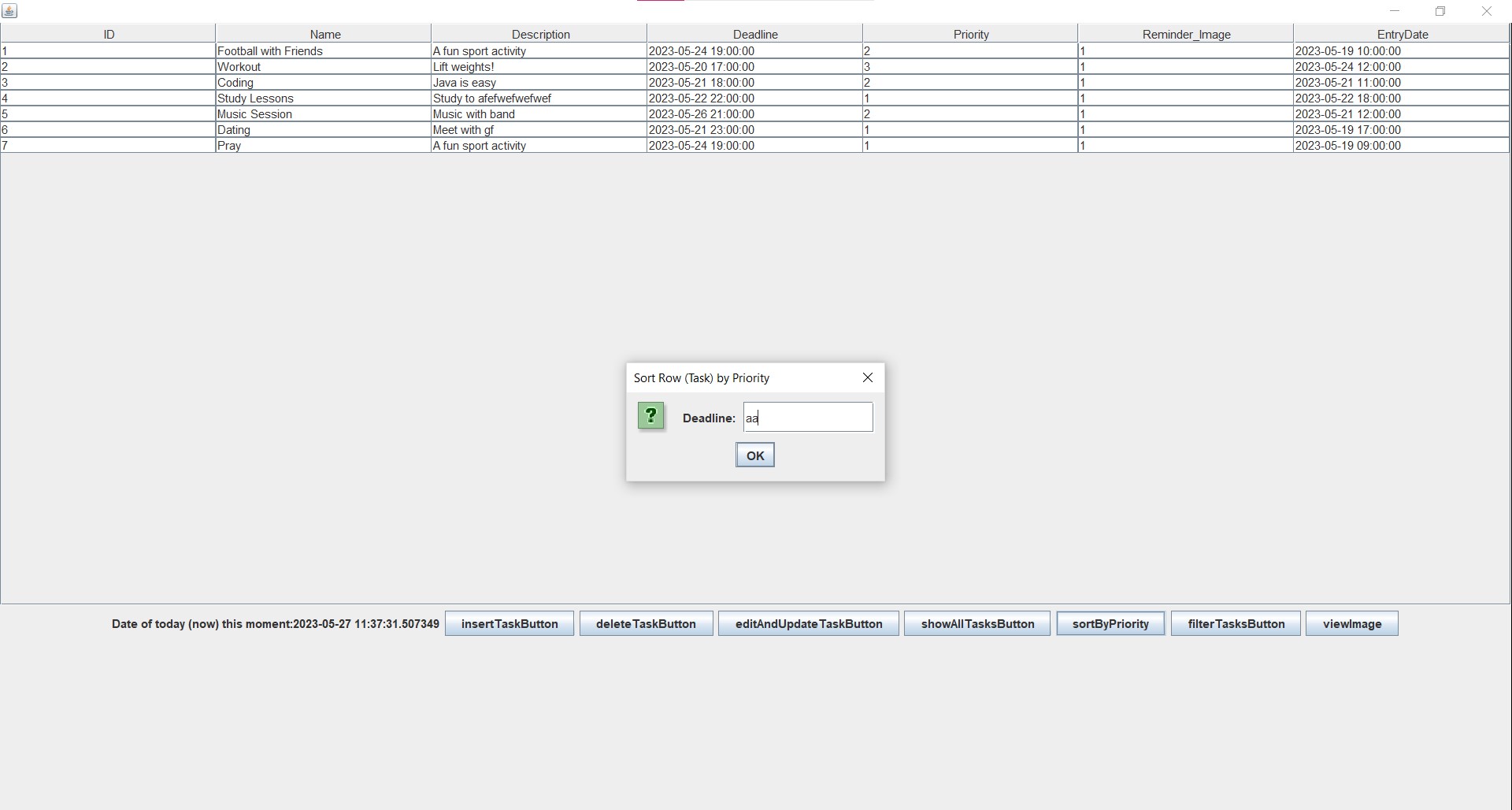
* + ***Example 2 – Proof that filtered tasks are sorted by priority***



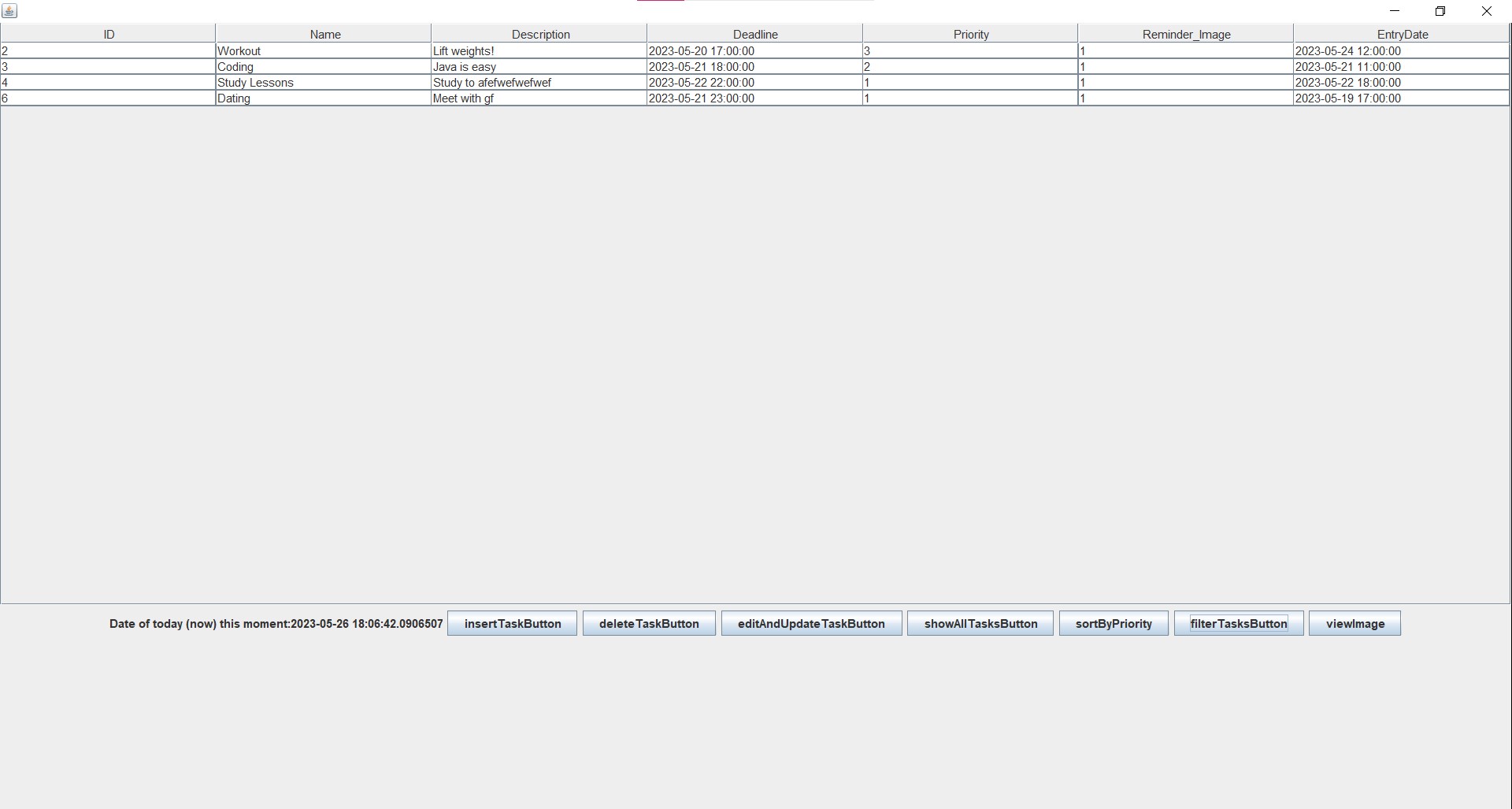
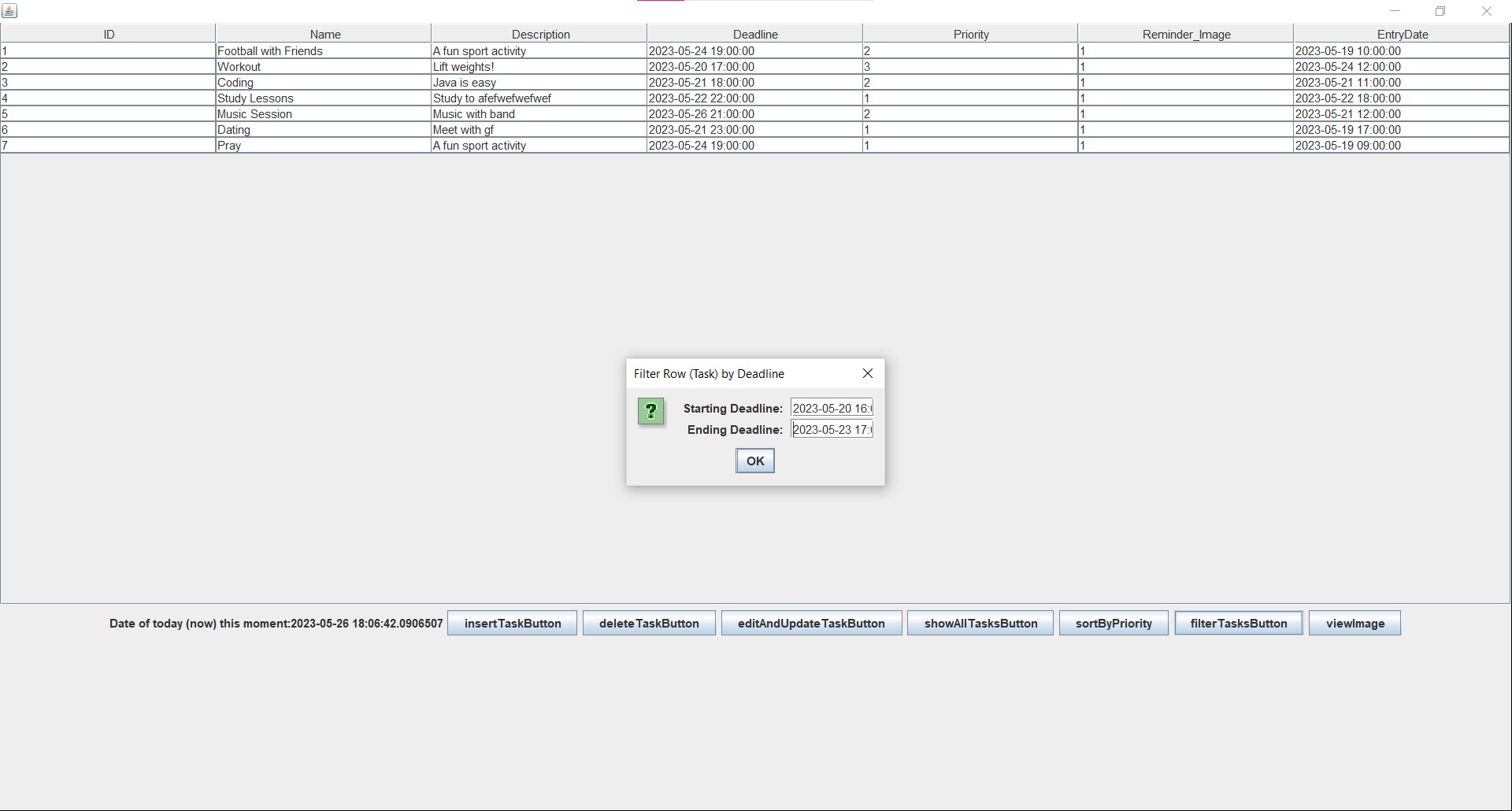
* ***NOT ENTERING ANY VALUES AS DEADLINE:*** This does not filters any tasks but just sorts the tasks by priority (works as a “sort” button)



* ***ENTERING A VALUE WHICH IS OTHER THAN A TIMESTAMP:*** Instead of producing an error message, system just filters whole tasks and no task is displayed

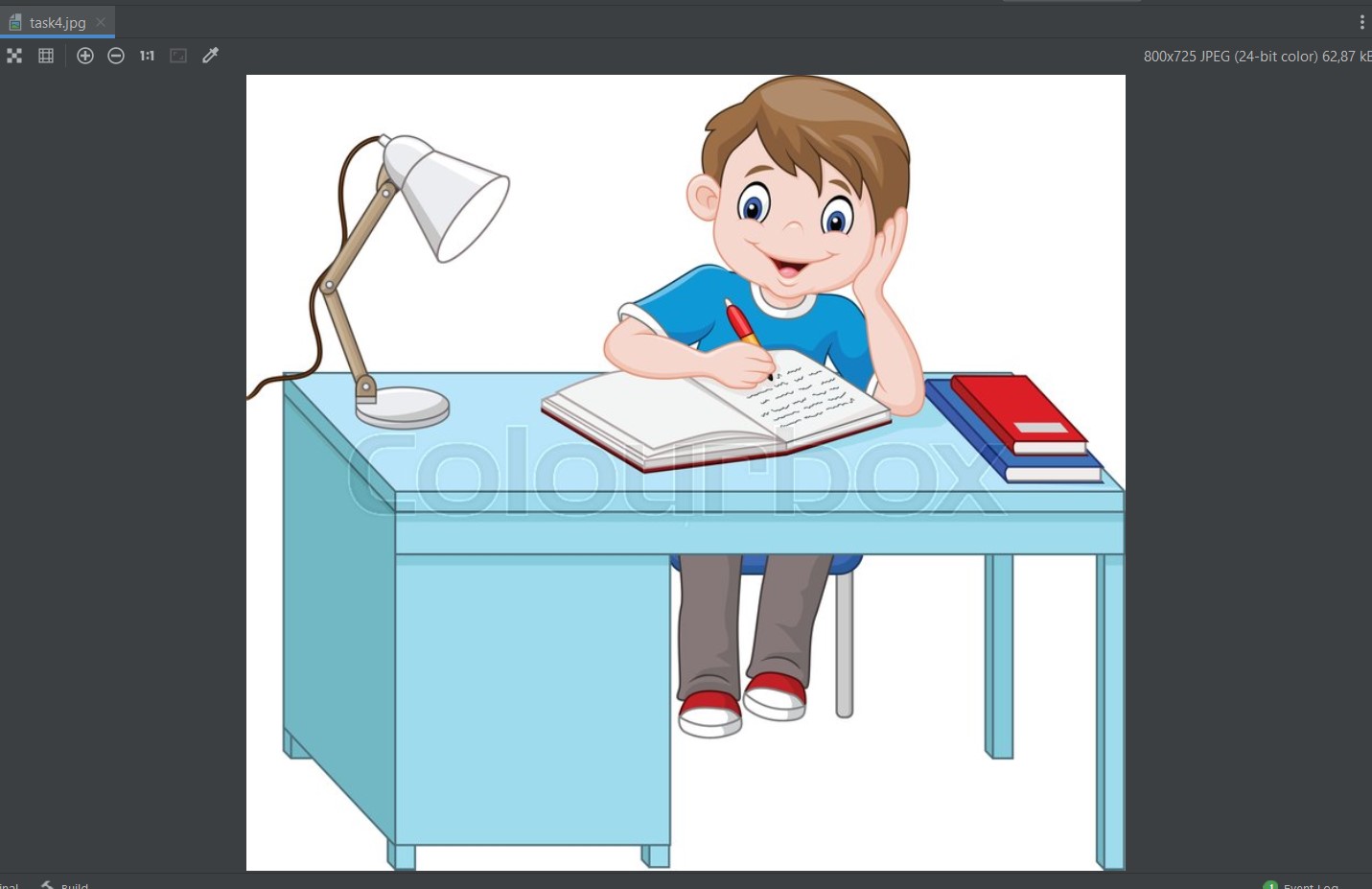


**FILTER TASK BY DEADLINE**: User can filter tasks by entering a starting deadline and an ending deadline

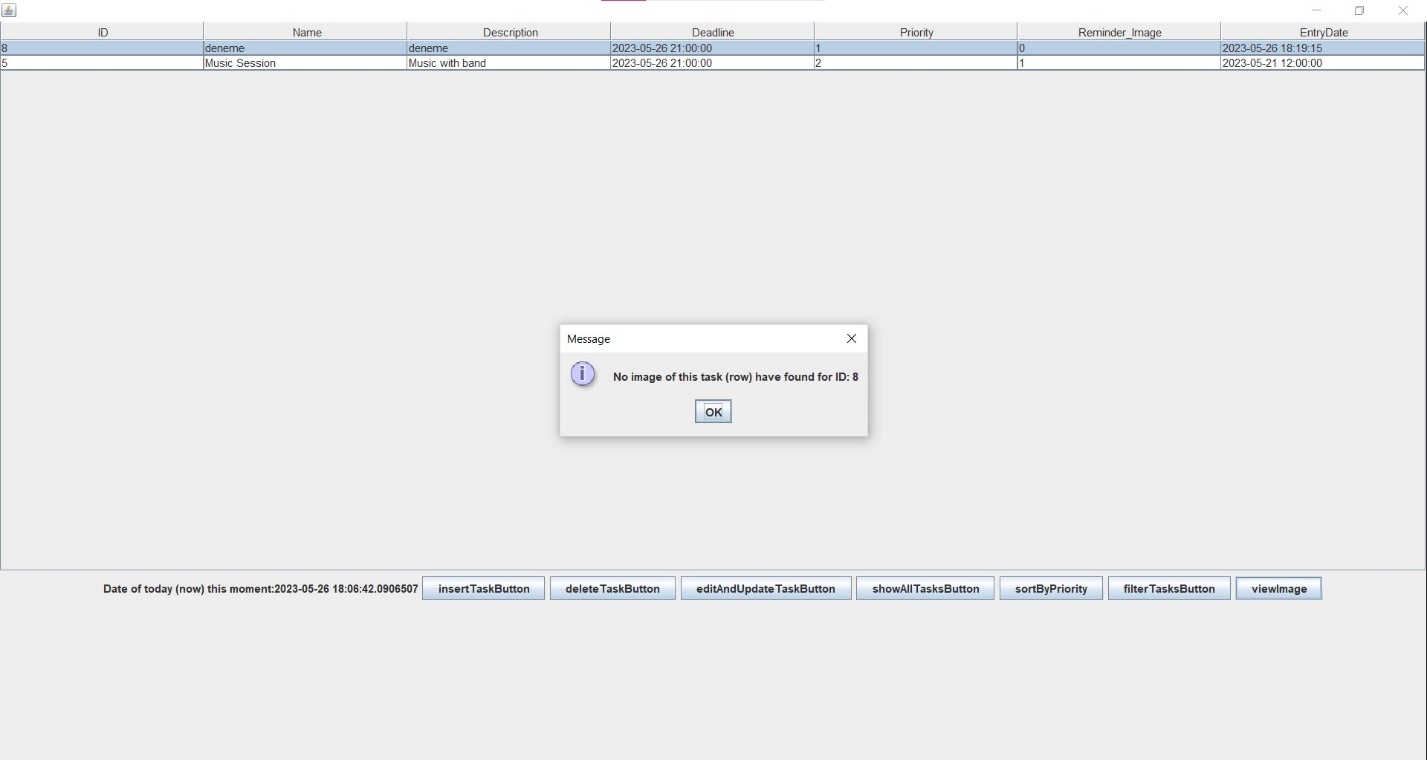


**VIEW THE REMINDER IMAGE OF THE TASK**: Some tasks has a reminder image and some images don’t. If system can find the image associated with that ID, image is shown

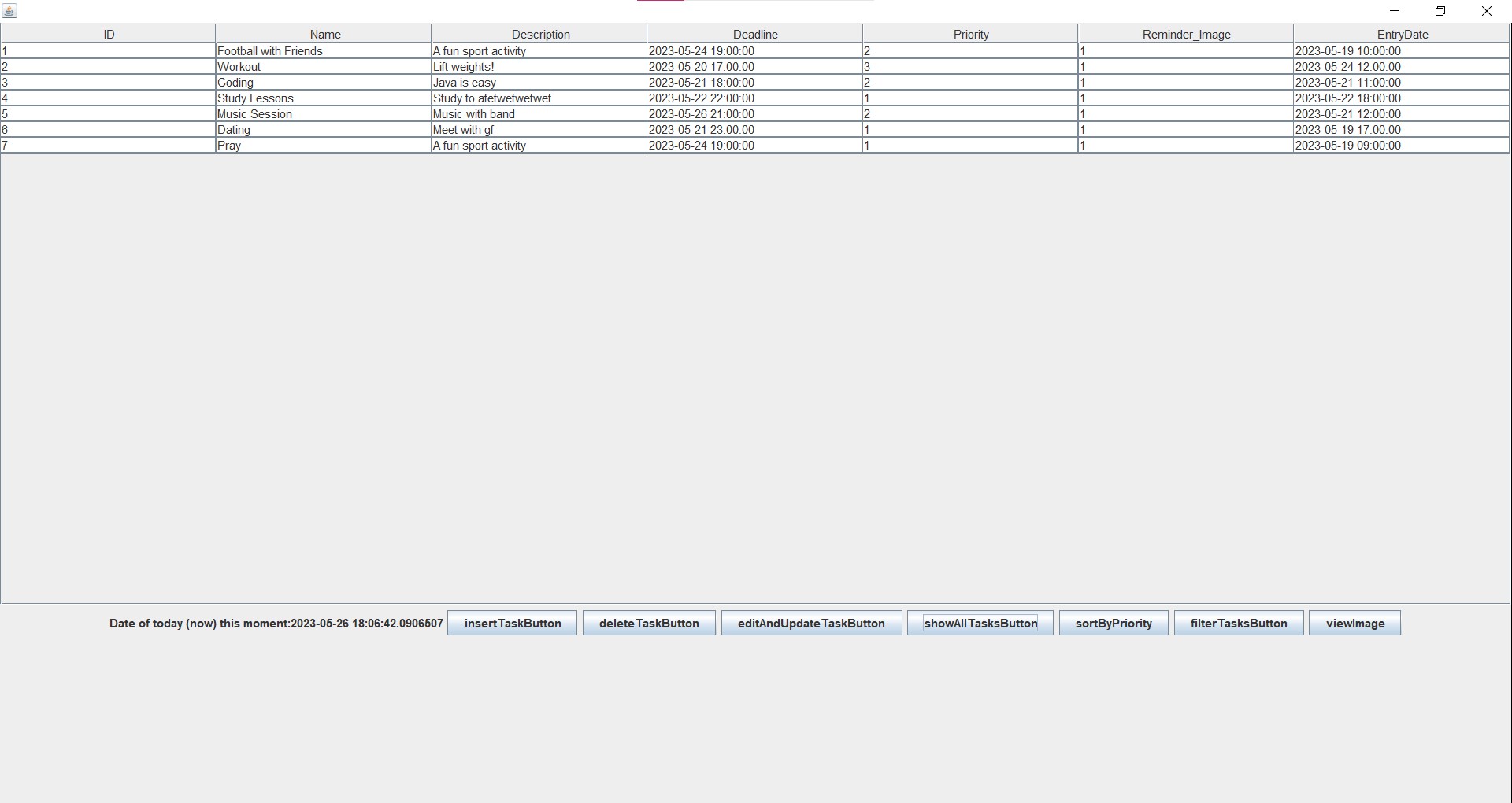
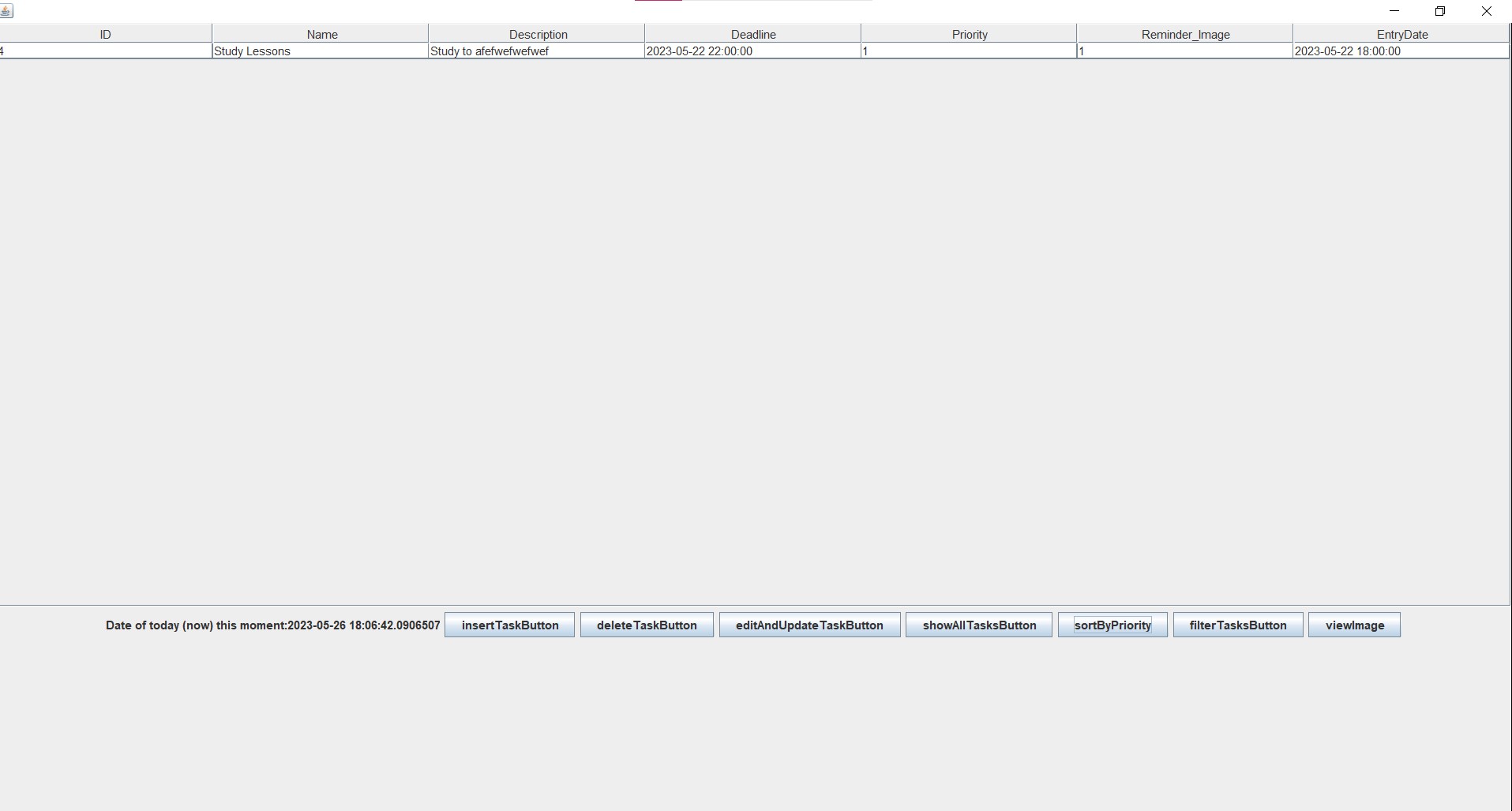
* ***SUCCESSFULLY DISPLAY REMINDER IMAGE OF A TASK (WHICH HAS A REMINDER IMAGE)***



* ***ATTEMPTING TO SHOW THE REMINDER IMAGE OF A TASK, WHICH DOES NOT ACTUALLY HAS A REMINDER IMAGE***



**SHOW ALL TASKS**: User presses “showAllTasksButton” and then system shows all the datas to the screen (by retrieving table data from database



# **Conclusion and Future Work**

This application is an app which shows the tasks that users have by fetching the data from a database. Users can make some operations on the tasks: inserting, updating, deleting, filtering, sorting by a priority (sorts tasks which has that entered Timestamp data) and view reminder image of the tasks.

Operations made on the tasks are both applied to the database and to the displayed JTable to the user

These features and/or functions can be added to my project:

* Register
* Adding tasks to favorites list
* “Recently deleted tasks” section
* Auto cleaning tasks whose deadline had already passed