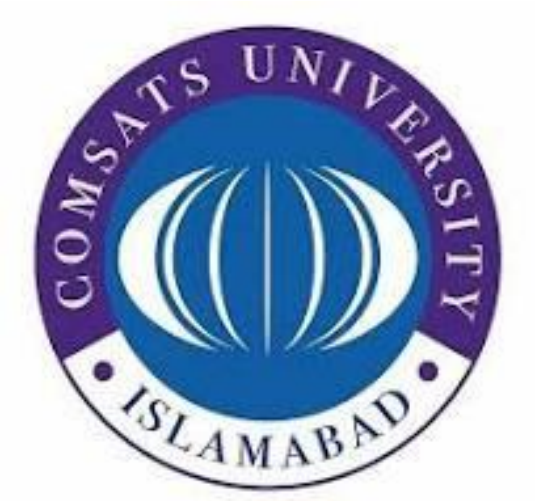


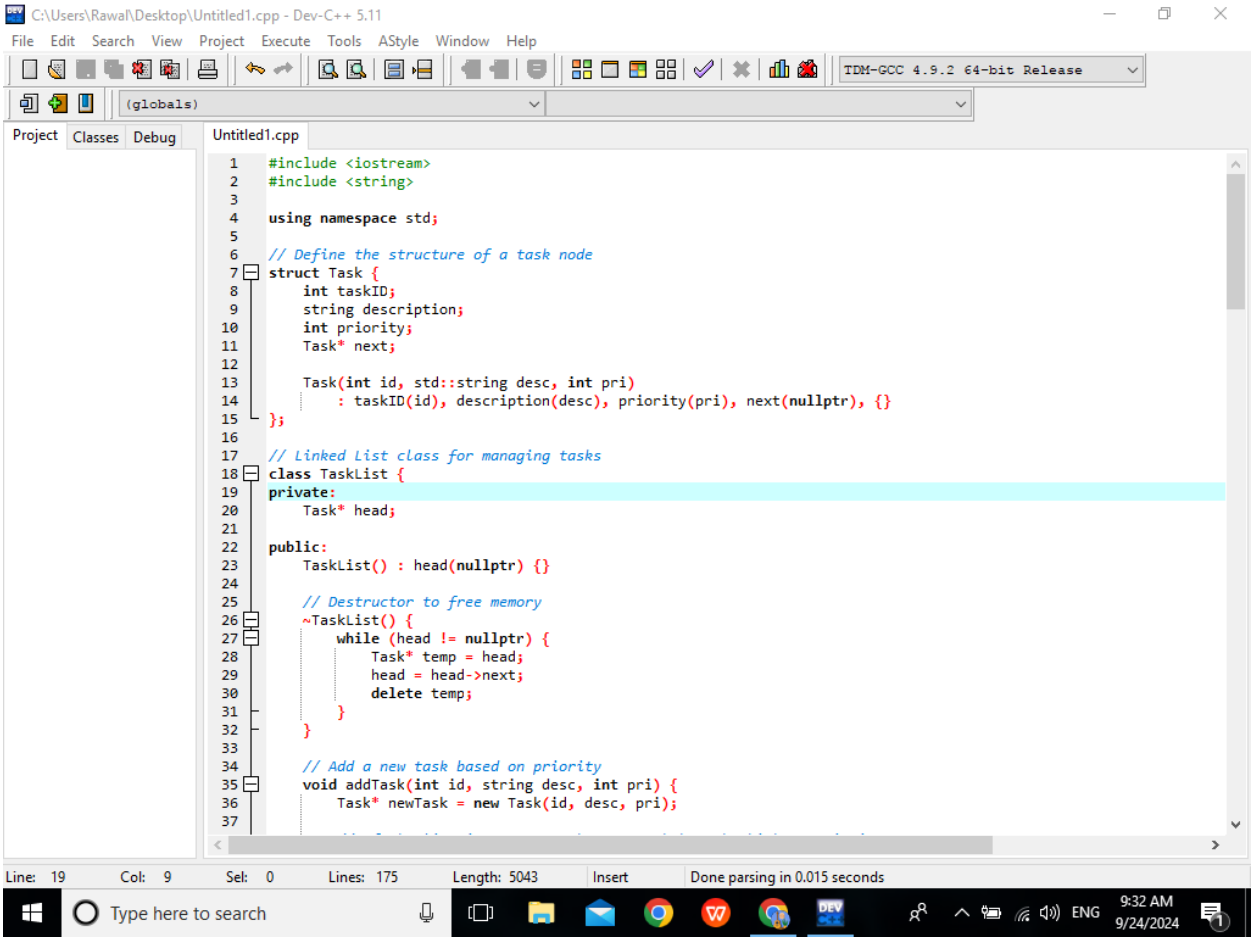
SUBJECT : DS

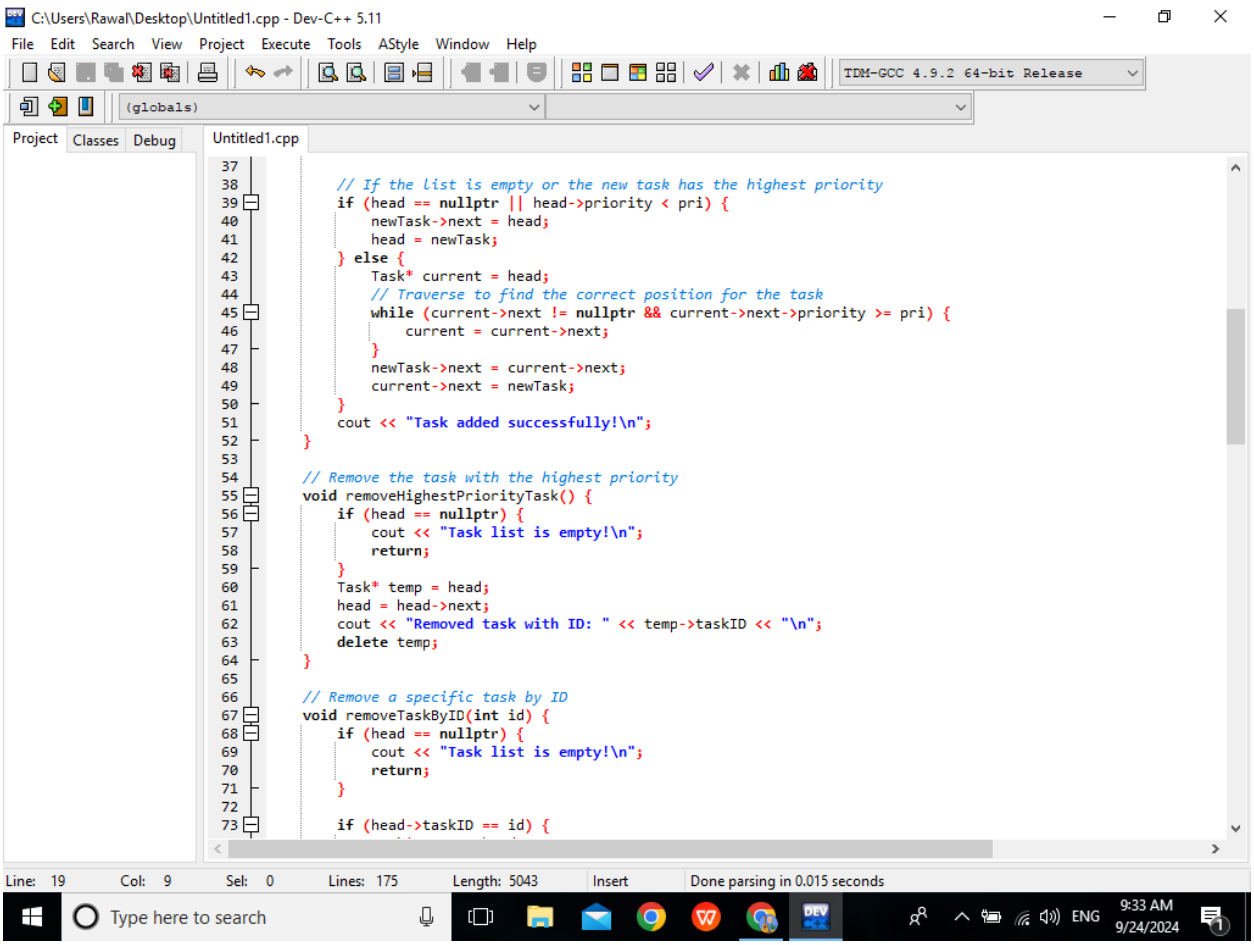
NAME : ARSLAN AHMED

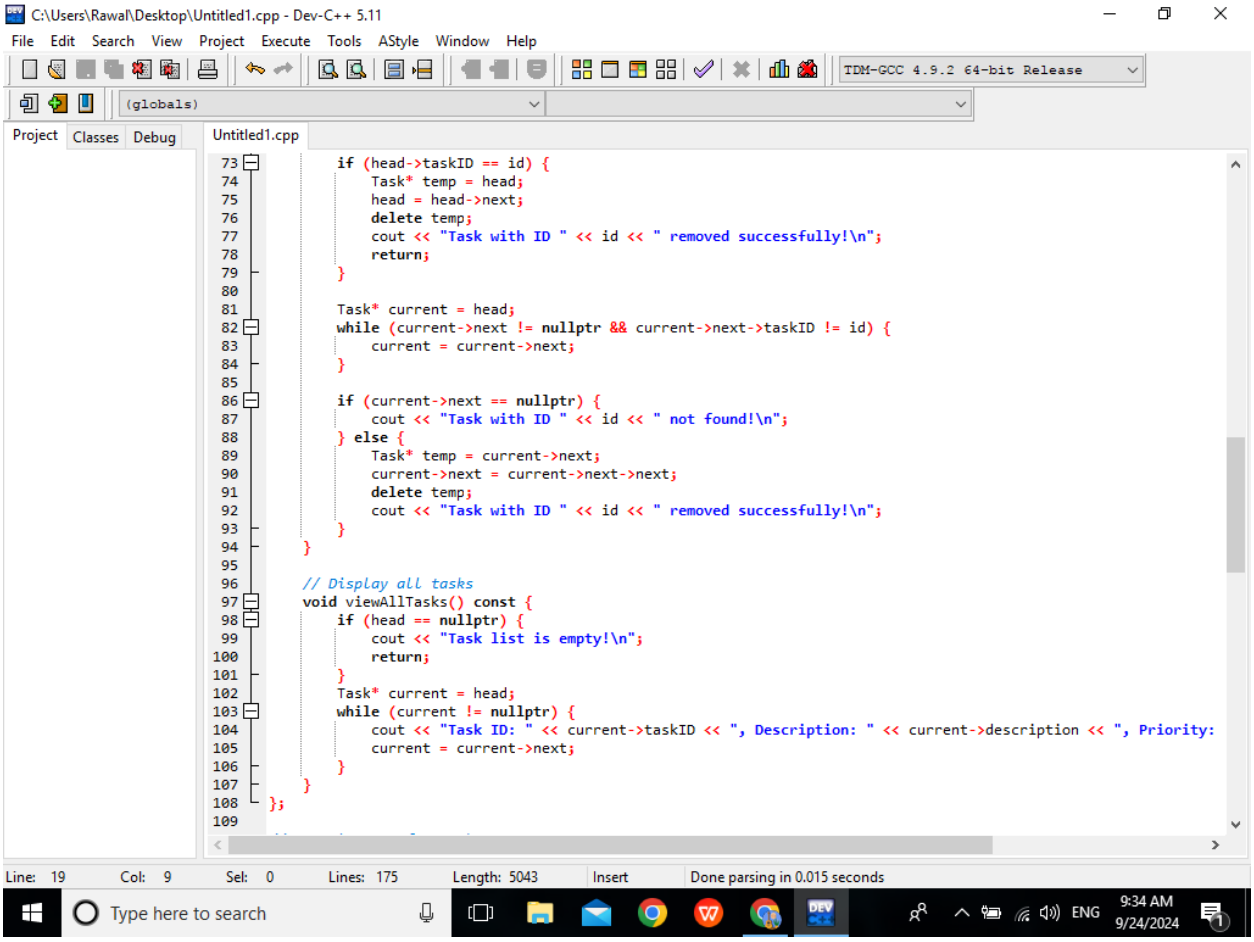
REGISTRATION NO. : SP23-BSE-036

DATE : 24-SEP-2024

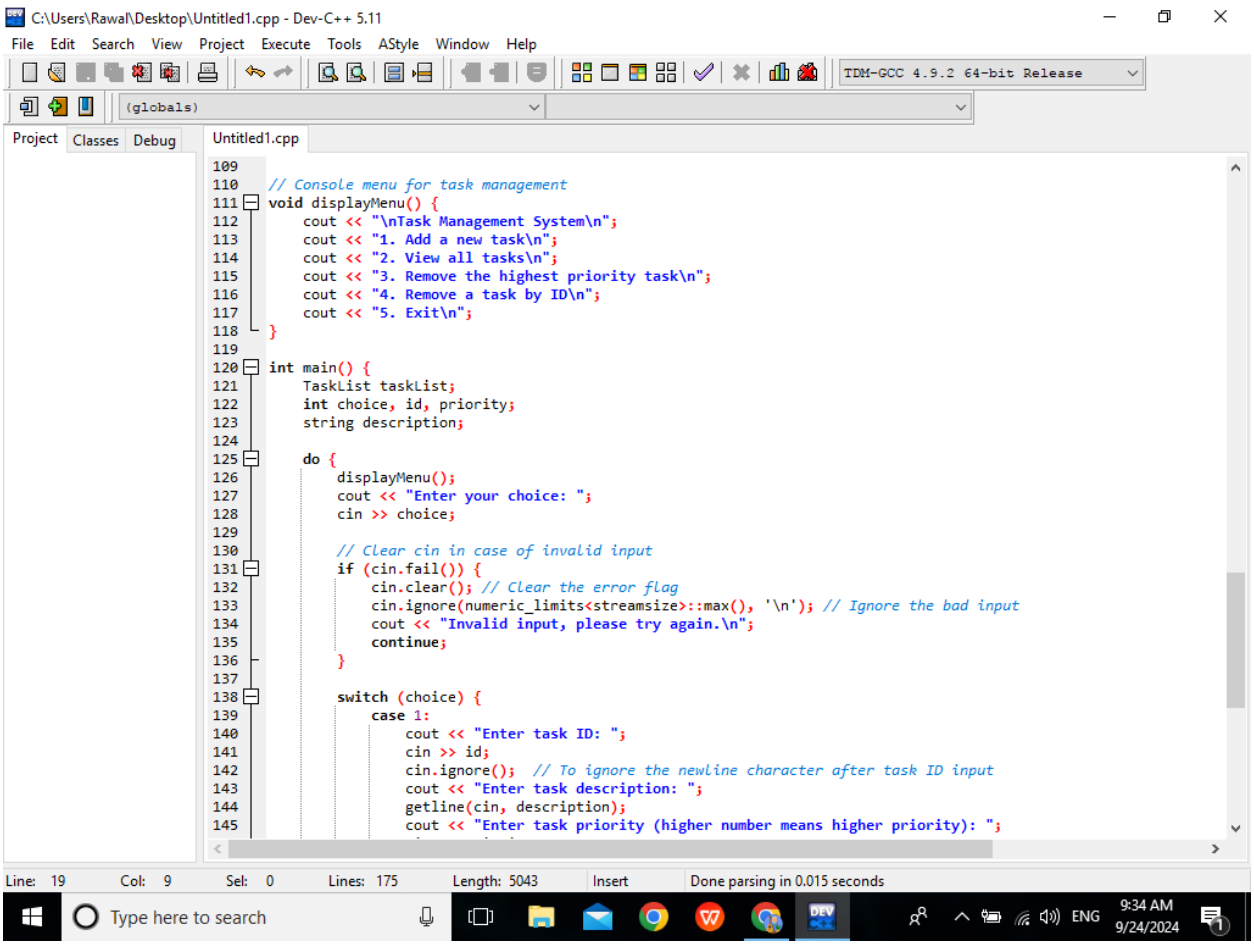
ASSINGMENT : 01


C:\Users\Rawal\Desktop\Untitled1.cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
Untitled1.cpp
#include <iostream>
#include <string>
using namespace std;
// Define the structure of a task node
struct Task {
int taskID;
string description;
int priority;
Task* next;
Task(int id, std::string desc, int pri)
: taskID(id), description(desc), priority(pri), next(nullptr), {}
};
// Linked list class for managing tasks
class TaskList {
private:
Task* head;
public:
TaskList() : head(nullptr) {}
// Destructor to free memory
~TaskList() {
while (head != nullptr) {
Task* temp = head;
head = head->next;
delete temp;
}
}
// Add a new task based on priority
void addTask(int id, string desc, int pri) {
Task* newTask = new Task(id, desc, pri);
Line: 19 Col: 9 Sel: 0 Lines: 175 Length: 5043 Insert Done parsing in 0.015 seconds
Type here to search
ENG 9:32 AM 9/24/2024

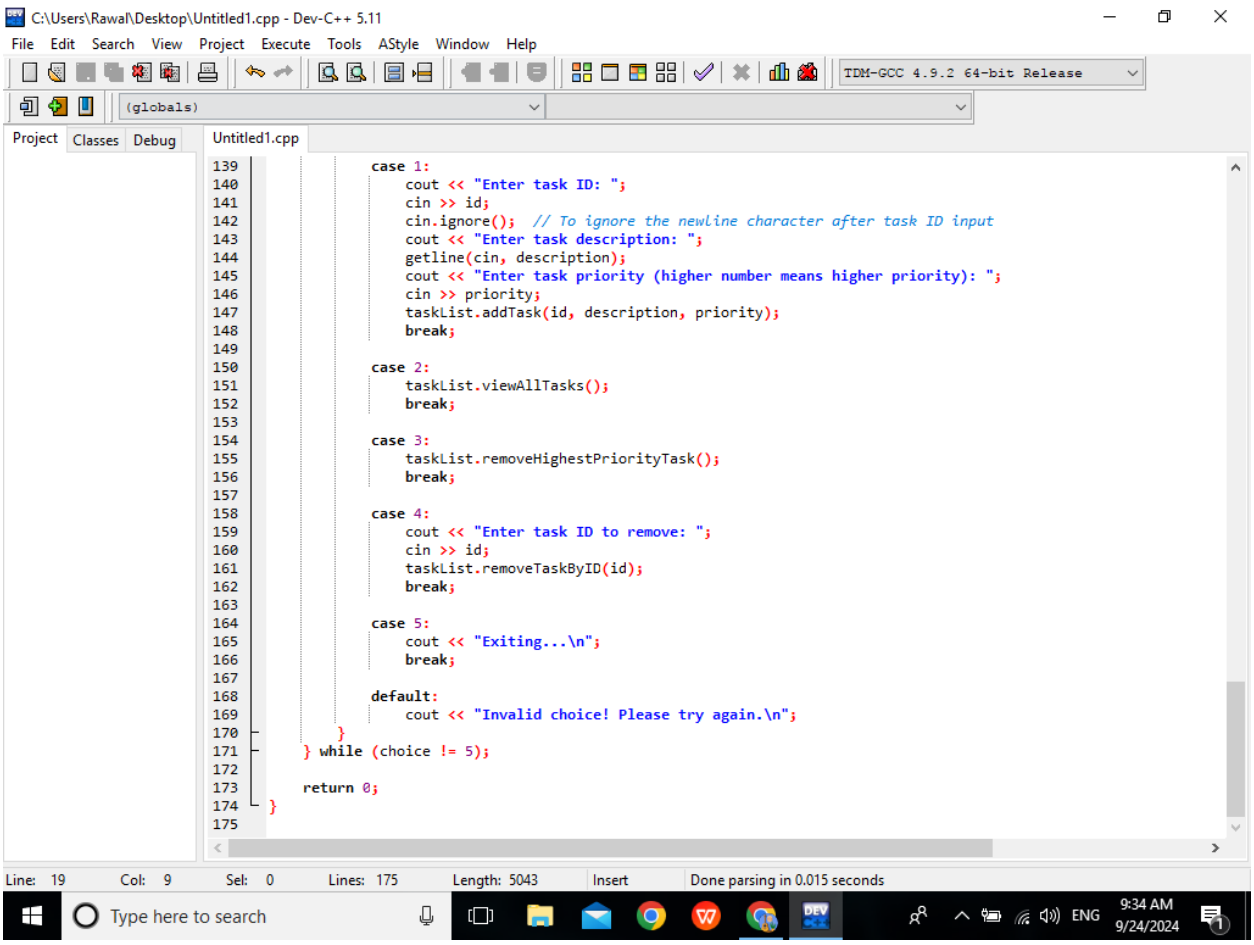

C:\Users\Rawal\Desktop\Untitled1.cpp - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
Untitled1.cpp
// If the list is empty or the new task has the highest priority
if (head == nullptr || head->priority < pri) {
newTask->next = head;
head = newTask;
} else {
Task* current = head;
// Traverse to find the correct position for the task
while (current->next != nullptr && current->next->priority >= pri) {
current = current->next;
}
newTask->next = current->next;
current->next = newTask;
}
cout << "Task added successfully!\n";
}
// Remove the task with the highest priority
void removeHighestPriorityTask() {
if (head == nullptr) {
cout << "Task list is empty!\n";
return;
}
Task* temp = head;
head = head->next;
cout << "Removed task with ID: " << temp->taskID << "\n";
delete temp;
}
// Remove a specific task by ID
void removeTaskByID(int id) {
if (head == nullptr) {
cout << "Task list is empty!\n";
return;
}
if (head->taskID == id) {
Line: 19 Col: 9 Sel: 0 Lines: 175 Length: 5043 Insert Done parsing in 0.015 seconds
Type here to search
ENG 9:33 AM 9/24/2024



```
        if (head->taskID == id) {
            Task* temp = head;
            head = head->next;
            delete temp;
            cout << "Task with ID " << id << " removed successfully!\n";
            return;
        }

        Task* current = head;
        while (current->next != nullptr && current->next->taskID != id) {
            current = current->next;
        }

        if (current->next == nullptr) {
            cout << "Task with ID " << id << " not found!\n";
        } else {
            Task* temp = current->next;
            current->next = current->next->next;
            delete temp;
            cout << "Task with ID " << id << " removed successfully!\n";
        }
    }

    // Display all tasks
    void viewAllTasks() const {
        if (head == nullptr) {
            cout << "Task list is empty!\n";
            return;
        }
        Task* current = head;
        while (current != nullptr) {
            cout << "Task ID: " << current->taskID << ", Description: " << current->description << ", Priority:
            current = current->next;
        }
    }
};
```



```

// Console menu for task management
void displayMenu() {
    cout << "\nTask Management System\n";
    cout << "1. Add a new task\n";
    cout << "2. View all tasks\n";
    cout << "3. Remove the highest priority task\n";
    cout << "4. Remove a task by ID\n";
    cout << "5. Exit\n";
}

int main() {
    TaskList taskList;
    int choice, id, priority;
    string description;

    do {
        displayMenu();
        cout << "Enter your choice: ";
        cin >> choice;

        // Clear cin in case of invalid input
        if (cin.fail()) {
            cin.clear(); // Clear the error flag
            cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Ignore the bad input
            cout << "Invalid input, please try again.\n";
            continue;
        }

        switch (choice) {
            case 1:
                cout << "Enter task ID: ";
                cin >> id;
                cin.ignore();  // To ignore the newline character after task ID input
                cout << "Enter task description: ";
                getline(cin, description);
                cout << "Enter task priority (higher number means higher priority): ";
```



REPORT

Reports from a task management system using a single linked list

Introduction:

This function implements a task management system in C++ using a single linked list. Each task is represented as a node in a list with a unique ID, description, and

priority. The system allows users to add jobs, view all jobs, and remove jobs by priority or a specific job ID.

Special Features:

- Store the project's unique ID, description, and priority.
- Adds a pointer to the next task (next) in the linked list.

2. Business Planning CoursE:

- Manage tasks using linked lists one by one. It allows you to add, remove, and view tasks.
- AddTask(*: Adds a new task to the appropriate location based on priority (high priority task is placed first).
- RemoveHighestPriorityTask(): Remove the highest priority task (i.e. the first task in the list).
- RemoveTaskByID(): Remove the task by finding its unique ID.
- ViewAllTasks(): Displays all tasks in the list.

3. Special Function:

- Provides a console-based menu for user interaction, where the user:
  - Add additional functionality.
  - Check all activities.
  - Extract the most important work.
  - Remove the task by its unique ID.
  - Exit the system.

Example flow:

- When someone adds a task, it is placed in the list according to priority.
- The viewer lists all current transactions with their ID, description and priority.
- The system also allows to remove work by ID or by highest priority.

This process helps manage projects on a priority basis, ensuring that the highest priority projects are always focused on.