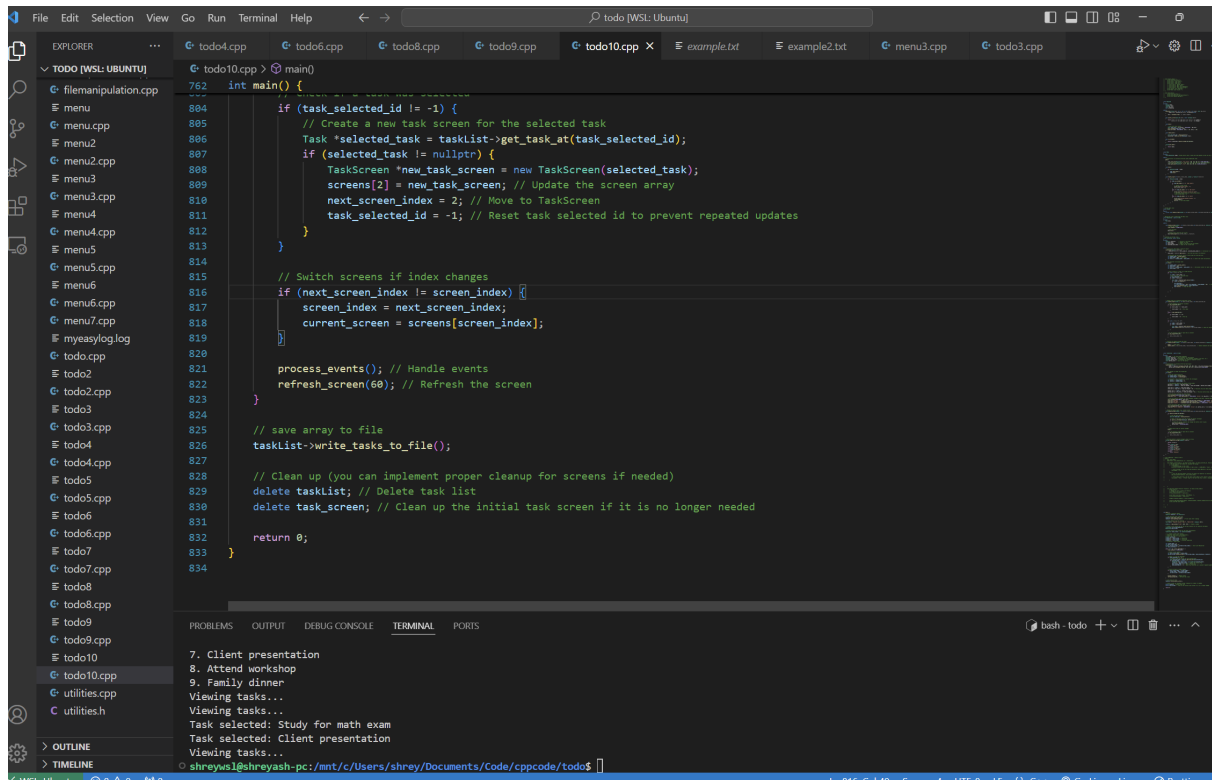


SIT102 H2 Something Awesome

Report



Screenshot of my VS CODE window after project completion

10 different iterations and over 800 lines of code in the final one, here we are! I have finally completed something awesome H2. I have built a to-do app that features OOP, file manipulation, dynamic memory management and a gui interface using splashkit.

I have started this project in D4 and after the first resubmission feedback I received, it has gone through many iterations as I try to implement the most advanced concepts to make my project onto the HD scale.

I have already explained most of the code structure of this project in my D4 project overview submission. Nevertheless, I will explain some of the extra features and changes which has been effected in this final iterations.

More Classes

I have added classes to be able to implement a Menu. I have `MenuItem` class and `Menu` Class.

MenuItem class

This `MenuItem` class represents a clickable menu item in a graphical interface. It contains a label, a rectangle for its position and size, a background color, and a hover state (`is_hovered`). The constructor initializes these attributes, setting the menu item's label and position. The `point_in_rect` method checks whether a given point (such as the mouse cursor) is within the rectangle bounds. The `draw` method renders the menu item, changing its background color when hovered. The `check_hover` method updates the hover state based on the current mouse position. The `is_clicked` method returns true if the item is hovered over and the left mouse button is clicked. The `get_label` method returns the label of the menu item.

```
class MenuItem
{
private:
    string label;
    rectangle rect;
    color bg_color;
    bool is_hovered;

public:
    MenuItem(string label, int x, int y, int width, int height)
        : label(label), bg_color(bg_color), is_hovered(false)
    {
        rect = rectangle_from(x, y, width, height);
    }

    bool point_in_rect(point_2d point, const rectangle &rect)
    {
        return point.x >= rect.x && point.x <= rect.x + rect.w &&
            point.y >= rect.y && point.y <= rect.y + rect.h;
    }

    void draw()
    {
        // Draw with hover effect
    }
}
```

```

        color draw_color = is_hovered ? COLOR_GRAY : bg_color;
        fill_rectangle(draw_color, rect);
        draw_text(label, COLOR_WHITE, rect.x + 20, rect.y + 20);
    }

    void check_hover()
    {
        point_2d mouse_pos = mouse_position();
        is_hovered = point_in_rect(mouse_pos, rect);
    }

    bool is_clicked()
    {
        return is_hovered && mouse_clicked(LEFT_BUTTON);
    }

    string get_label()
    {
        return label;
    }
};

```

MenuClass

```

class Menu
{
private:
    vector<MenuItem> items; //Using vectors here, could have

public:
    // Constructor to initialize the menu with predefined items
    Menu()
    {
        items.push_back(MenuItem("1. View tasks", 200, 150, 400, 80));
        //items.push_back(MenuItem("2. Add task", 200, 250, 400, 80));
        items.push_back(MenuItem("3. Exit", 200, 350, 400, 80));
    }
}

```

```

void draw()
{
    for (MenuItem &item : items)
    {
        item.check_hover();
        item.draw();
    }
}

void handle_input(int &next_screen_index, window w, TaskL
{
    for (MenuItem &item : items)
    {
        if (item.is_clicked())
        {
            if (item.get_label() == "1. View tasks")
            {
                // switch to task screen
                write_line("Viewing tasks...");
                next_screen_index = 1;
            }
            else if (item.get_label() == "2. Add task")
            {
                write_line("Use terminal to add new task"
                // add_task_handler(tasklist, next_screen
                //next_screen_index = 3;
            }
            else if (item.get_label() == "3. Exit")
            {
                write_line("Exiting without saving...");
                close_window(w);
                exit(0); // Exit the program
            }
        }
    }
}
};

```

The `Menu` class represents a menu system with clickable items using a `vector` of `MenuItem` objects. The constructor initializes the menu with predefined items, such as "View tasks" and "Exit." The `draw` method iterates through each `MenuItem`, checking if it's hovered and drawing it.

The `handle_input` method listens for user interactions. If an item is clicked, it triggers specific actions: for "View tasks," it changes the `next_screen_index` to switch screens; for "Exit," it closes the window and terminates the program.

You will notice that I have commented out the part for menu option to add task. I will explain in a later section.

More Screens

Where I previously had only 1 screen in D4, I now have 3 screens. I have created two new screen classes: `MenuScreen` and `TaskScreen`.

MenuScreen Class

The `MenuScreen` class contains the private attribute of `Menu` class and a method `handle_screen` that handles draws the menu screen by calling the `draw` method of the menu object. It also calls the `handle_input` of the menu which controls the user interaction with the GUI.

```
// MenuScreen class integrating the Menu class
class MenuScreen : public Screen
{
private:
    Menu menu;

public:

    void handle_screen(window w, int &next_screen_index,int &
        // Clear the window
        clear_window(w, COLOR_BLACK);

        // Draw the menu
        menu.draw();

        // Handle input (hovering and clicks)
        menu.handle_input(next_screen_index, w, tasklist);
```

```
    }
};
```

TaskScreen Class

The `TaskScreen` class is used to display the details of the selected task on a screen alongside a button to mark the task as complete. It contains a pointer to a `Task` object, along with three rectangle attributes: `big_rect` for the main task details, `info_rect` for metadata such as category and due date, and `button_rect` for a "Mark as Complete" button.

The constructor initializes the `TaskScreen` by setting the task to be displayed and defining the dimensions and positions of the various rectangles to create a layout for displaying task information and controls.

```
class TaskScreen : public Screen
{
private:
    Task *task; // The task being displayed
    rectangle big_rect; // Big rectangle for task details
    rectangle info_rect; // Rectangle for task metadata (cate
    rectangle button_rect; // Rectangle for "Mark as Complete
    color button_color; // Color of the button

public:
    // Constructor to initialize the task and layout
    TaskScreen(Task *t)
        : task(t), big_rect(rectangle_from(100, 100, 600, 150
        , button_rect(rectangle_from(100, 400, 200, 50)), but
    {
    }
}
```

The draw method calculates the positioning of each rectangle and draws the taskscreen elements.

```
/ Draw method to render the TaskScreen
void draw()
{
    // Get the window dimensions
    int window_width = screen_width();
```

```

int window_height = screen_height();

// Calculate X and Y positions to center the rectangle
int center_x = window_width / 2;
int center_y = window_height / 2;

// Set positions based on the center of the window
big_rect.x = center_x - big_rect.width / 2;
big_rect.y = center_y - (big_rect.height + info_rect.height) / 2;

info_rect.x = center_x - info_rect.width / 2;
info_rect.y = big_rect.y + big_rect.height + 10; // Position info rect below big rect

button_rect.x = center_x - button_rect.width / 2;
button_rect.y = info_rect.y + info_rect.height + 10; // Position button below info rect

// Draw the big rectangle for task details
fill_rectangle(COLOR_LIGHT_GRAY, big_rect);
draw_text("Task: " + task->get_detail(), COLOR_BLACK, big_rect.x + 10, big_rect.y + 10);

// Draw the smaller rectangle for task metadata (category, due date, created)
fill_rectangle(COLOR_WHITE, info_rect);
draw_text("Category: " + category_to_string(task->get_category()), COLOR_BLACK, info_rect.x + 10, info_rect.y + 10);
draw_text("Due Date: " + formatDate(task->get_due_date()), COLOR_BLACK, info_rect.x + 10, info_rect.y + 20);
draw_text("Created: " + formatDate(task->get_creation_time()), COLOR_BLACK, info_rect.x + 10, info_rect.y + 30);

// Draw the green "Mark as Complete" button
fill_rectangle(button_color, button_rect);
draw_text("Mark as Complete", COLOR_WHITE, "arial", 24, button_rect.x + 10, button_rect.y + 10);
}

```

Lastly, the `handle_input` method handles input for the clicking on the "Mark as complete" button. Additionally, it listens if the space bar is pressed and set `next_screen_index` to previous screen to send user to previous screen.

```

// Method to handle input (e.g., button click)
void handle_screen(window w, int &next_screen_index,int &
{
    if (mouse_clicked(LEFT_BUTTON))
    {
        // Get the mouse position
        point_2d mouse_pos = mouse_position();

        // Check if the mouse click was inside the button
        if (point_in_rectangle(mouse_pos, button_rect))
        {
            // Mark the task as complete and change the b
            task->mark_complete();
            button_color = COLOR_DARK_GRAY;
        }
    }

    // Redraw the screen to reflect changes
    draw();

    // Go to previous screen if space key is pressed
    if (key_typed(SPACE_KEY))
    {
        next_screen_index = 1;
    }
}

// Utility method to convert Category enum to string
string category_to_string(Category category)
{
    switch (category)
    {
        case Category::Work:
            return "Work";
        case Category::School:
            return "School";
        case Category::Personal:
            return "Personal";
    }
}

```



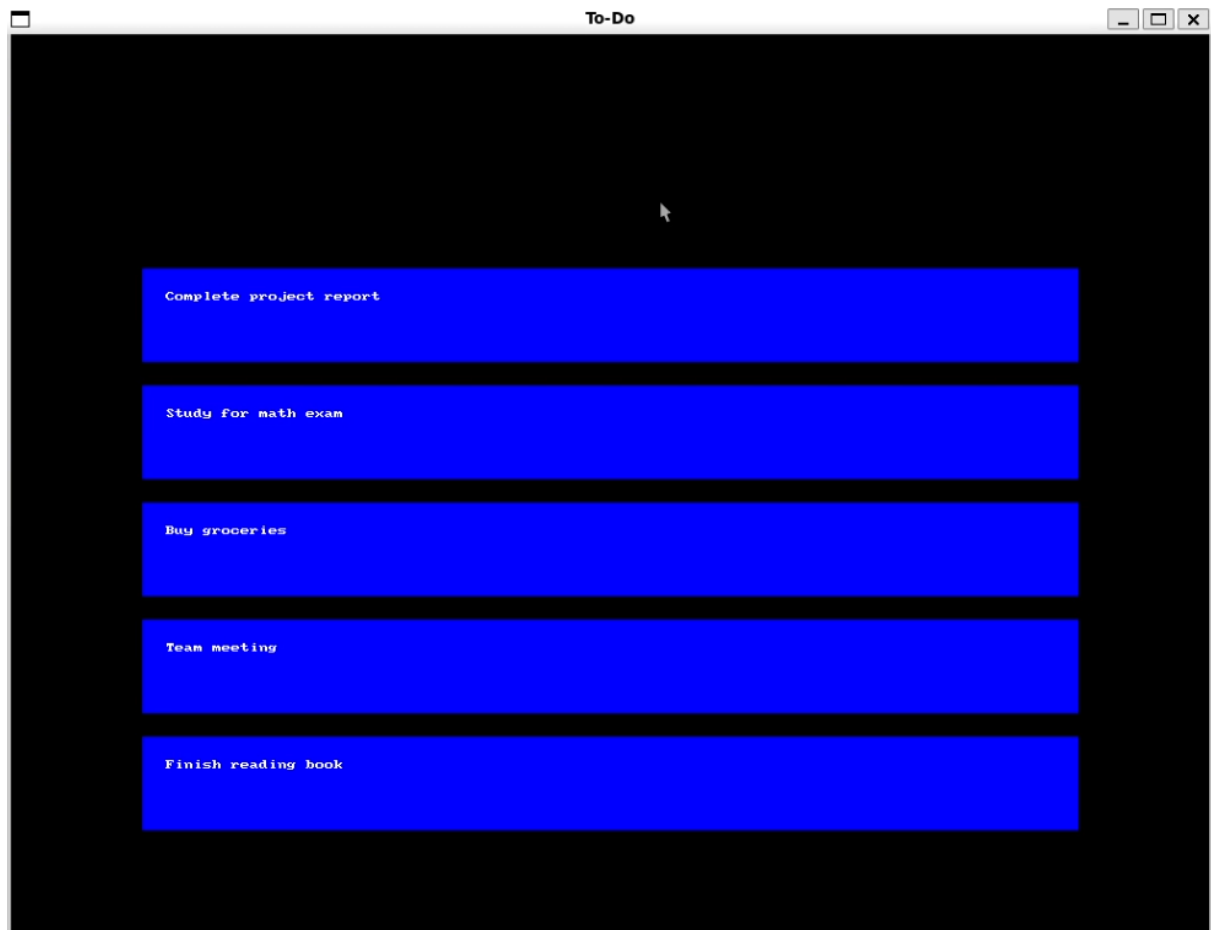
```
        default:  
            return "Unknown";  
        }  
    }  
};
```

Screen Switching Mechanism

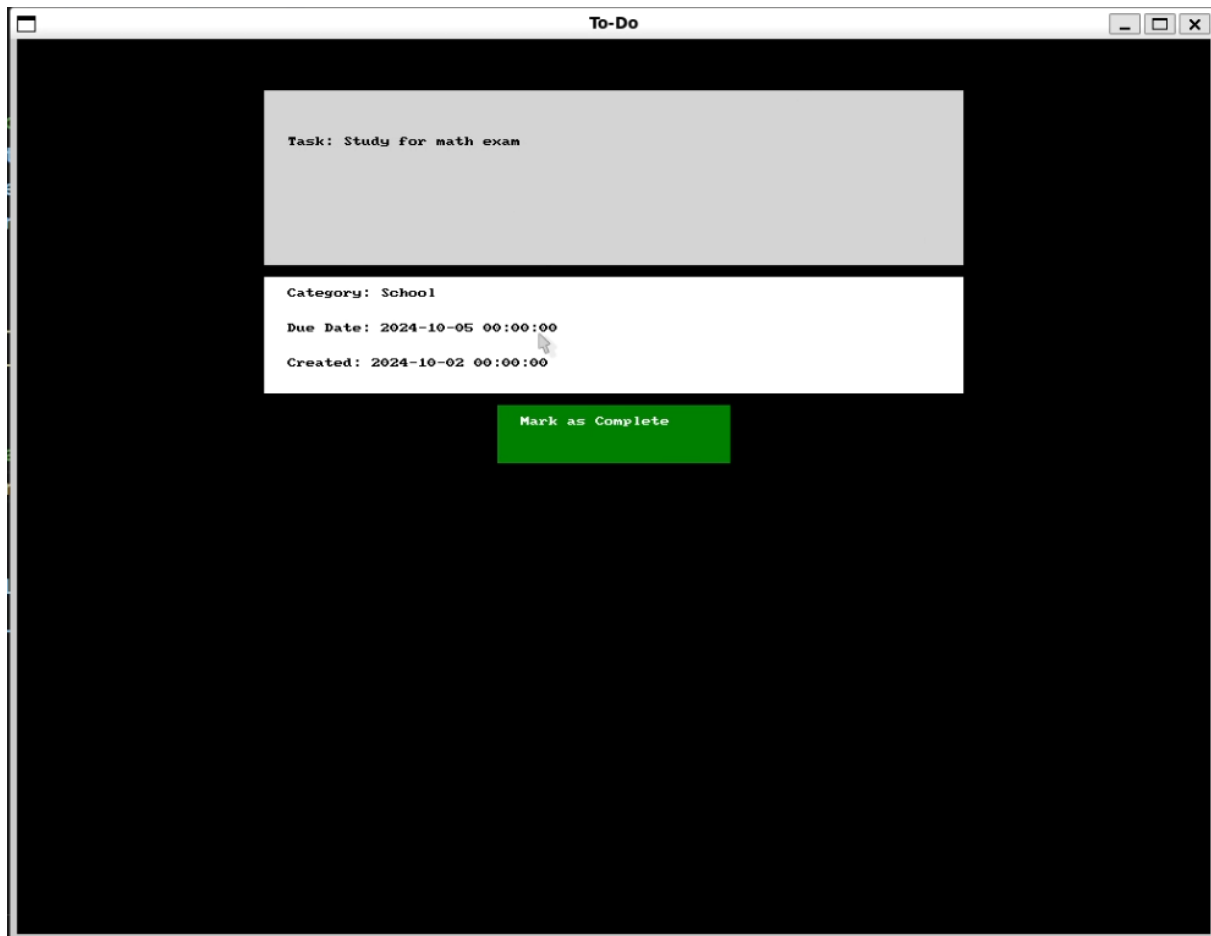
Below are the three screens present in my program.



Menu Screen



List Screen



Task Screen

Since I am dealing with three different screens in my program, I had to implement a screen switcher mechanism to be able to switch screens accordingly. This is what I came up with.

```
/ Create a List Screen and set the current tasklist to its ta
ListScreen list_screen(taskList);
MenuScreen menu_screen;

// Create initial task screen for the task (placeholder)
TaskScreen *task_screen = new TaskScreen(&task);

// // Create an add screen object
// AddScreen *add_screen = new AddScreen();
// Create an array of screen pointers
Screen* screens[4];
screens[0] = &menu_screen; // MenuScreen
screens[1] = &list_screen; // ListScreen
```

```

screens[2] = task_screen;    // Initial TaskScreen
// screens[3] = add_screen;

int screen_index = 0;
int next_screen_index = 0;
Screen* current_screen = screens[screen_index]; // Start
int task_selected_id = -1;

while (true && !(quit_requested())) {
    clear_window(w, COLOR_BLACK);

    // Handle screen updates
    int next_screen_index = screen_index;
    current_screen->handle_screen(w, next_screen_index, t

    // Check if a task was selected
    if (task_selected_id != -1) {
        // Create a new task screen for the selected task
        Task *selected_task = taskList->get_task_at(task_
        if (selected_task != nullptr) {
            TaskScreen *new_task_screen = new TaskScreen(
            screens[2] = new_task_screen; // Update the s
            next_screen_index = 2; // Move to TaskScreen
            task_selected_id = -1; // Reset task selected
        }
    }

    // Switch screens if index changes
    if (next_screen_index != screen_index) {
        screen_index = next_screen_index;
        current_screen = screens[screen_index];
    }

    process_events(); // Handle events
    refresh_screen(60); // Refresh the screen
}

```

So, in my main method, I have created an object for each screen and assigned them to an index in the array `screens[4]` of class `Screen` and size 4. (The array is of size 4 because I was supposed to have 4 screens but more on that later!) I also have a pointer `current_screen` of type `Screen` that points to the current screen. The `current_screen` allows us to handle the screen as it will always point to the current screen from the screens array and hence we can continuously call the `handle_screen` method of the pointer `current_screen`.

The `handle_screen` method takes in an argument `next_screen_index` which is responsible for instructing whether to change screen and to which screen.

Further below, we see an if condition block that check if `next_screen_index` is not equal to `screen_index`, then it updates the `screen_index` to `next_screen_index`. In other words, whenever the value of `next_screen_index` changes to a new value after the `handle_screen` method call, the program is being instructed to switch to the next screen.

Then the pointer `current_screen` changes to point to the new index from `screens` array. Hence the screen has been switched.

Inheritance

You will notice that array `screens` is of type class `Screen` but it has elements each of different classes. Someone new to OOP might question how this is possible. There is where inheritance comes in. You notice that in the declarations of `TaskScreen`, `ListScreen`, and `MenuScreen`, they all have `: public Screen`. This means that they have inherited the base class `Screen`. Thus we are able to store them in the array `screens`. `Screen` becomes the parent class and the others are all Child Classes. The Child classes will all inherit the properties(attributes and methods) of their base class.

Polymorphism

Since all the Child classes, `TaskScreen`, `ListScreen`, and `MenuScreen` inherit the method `handle_screen` but they all have slightly different behaviours, we use polymorphism to make it work.

You will notice that in each of the `handle_screen` method declaration of the child classes there is an `override` next to it. Similarly, you will see a `virtual` in the method declaration in the base class. `virtual` and `override` work together to enable polymorphism.

Base class declaration

```
// Screen base class
class Screen
{
public:
    virtual void handle_screen(window w, int &next_screen_index, int &tasklist);
};
```

method declaration in the child classes

```
void handle_screen(window w, int &next_screen_index, int &tasklist);
```

Add Task Dilemma

I tried to implement an Add Task feature to my To-Do app. I tried for hours but to no avail. Basically, I had to create a new screen class Add Task and equip it with the appropriate properties to handle the add task feature in the terminal. Below is the code I tried to use.

```
class AddScreen : public Screen {
public:
    bool task_added;
    AddScreen(): task_added(false) {} // Constructor

    void handle_screen(window w, int &next_screen_index, int &tasklist) {
        // If task has not been added yet, display the message
        if (!task_added) {
            // Display message on the screen
            draw_text("Please use the terminal to add a task.");

            // After drawing, we set the flag and proceed to the next screen
            task_added = true;
        }
        else {
            // Call the add_task_handler function to handle the task
            add_task_handler(tasklist, next_screen_index);

            // After adding the task, set the next screen index
            task_added = false; // Reset flag for the next time
        }
    }
};
```

```

    }

}

void add_task_handler(TaskList *taskList, int &next_screen) {
    write_line("");
    // Prompt the user to enter task details
    write("Enter task detail: ");
    string newtask_detail = read_line();

    write("Enter due date (format: YYYY-MM-DD): ");
    string newtask_duedate = read_line();

    Category newtask_category = select_category();

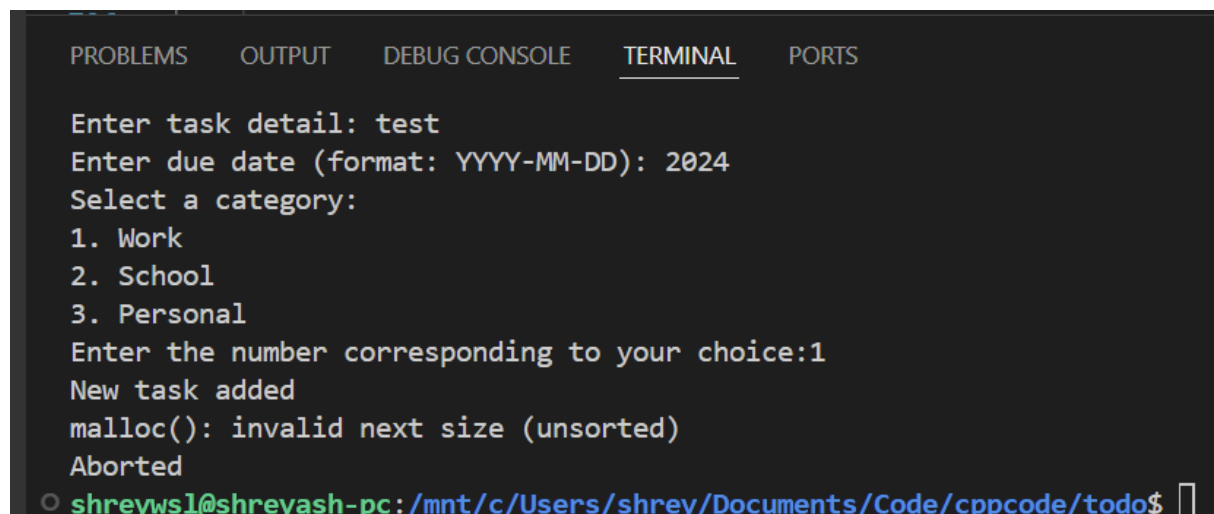
    taskList->add_task(newtask_detail, newtask_duedate, newtask_category);
    write_line("New task added");

}

};

```

I tried multiple variations or different mechanisms/tweaks but I simply could not get it to work. This is the error I was getting.



```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Enter task detail: test
Enter due date (format: YYYY-MM-DD): 2024
Select a category:
1. Work
2. School
3. Personal
Enter the number corresponding to your choice:1
New task added
malloc(): invalid next size (unsorted)
Aborted
shreyws1@shreyash-pc: /mnt/c/Users/shrey/Documents/Code/cppcode/todo$

```

Apparently, this seems to be an issue arising due to the window which is rendering at a certain refresh rate and the terminal part of the program which prompts user to enter new task details, were not in sync. So I simply disabled the Add Task feature for now. Maybe in a later update, I can resolve it 😊
Anyways, so this dilemma explains the inconsistencies in my code and the leftover traces of my desperate attempt.

Reflection

I cross-checked the task sheet and it did not require any reflection for this task. Therefore, I shall provide only a short one as a precaution. Most of the thing that I have to say about this project has already been covered in the above section anyway.

I affirm that this project is of HD level for its complexity. The number of issues/bugs that I have ran into is innumerable. I acknowledge that in some cases I have sought assistance from Co-Pilot Ai to help me debug my program, yet this code and project is fully mine. In this project, I have went well outside the scope of the learning material while demonstrating mastery over the taught concepts by integrating it with my own learning. I have used OOP extensively which solely elevates the standard of my program - it was not just classes but inheritance and polymorphism as well. The different mechanism/algorithms I had to figure out to make the screen switcher and to integrate all these classes and methods seamlessly all together demonstrate high level problem-solving skills from my part. Additionally, I have paid heed to all the feedback of my tutor in my previous submissions to make this my best work. I have implemented more comments and produced a well-detailed report. The many hours - or days rather- spent in this project were well worth it. I am satisfied of my work and most importantly of my learning.

I forgot to mention above that I have so many different iterations because I was trying out different mechanisms/features separately before using them in my main program. In my cases, I have used dummy modules/structures for testing.

Video:

<https://youtu.be/YuEpE4CKCJU>