FINAL PROJECT GROUP 3 PROGRESS REPORT

Work done so far:

1. [✔] Login selection screen.
2. [✔]Registration Screen.
3. [✔]Implement registration system
4. [✔] Login screen for both Teacher and student.
5. [✔] The Calendar layout.
6. [✔] Data Storage.
7. [✔] References in the paper.
8. [✔] The General and specific objectives in the documentation paper.
9. [✔]Modularize the code
10. [✔]Whole code
11. [ ] Data Dictionary
12. [ ] Results and discussion
13. [ ] Flow Chart
14. [ ] Pseudo Code
15. [ ] Conclusion

Contributions:

Braganza, Ralph Angelov F. – added more references and fixed the introduction's grammar and flow. Added a couple more slides to the powerpoint to make the foundation of what we are trying to portray.

Condino, Niel Vincent B. – Helped with the modularization of the code, created the registration system, fixed errors and improved the overall code.

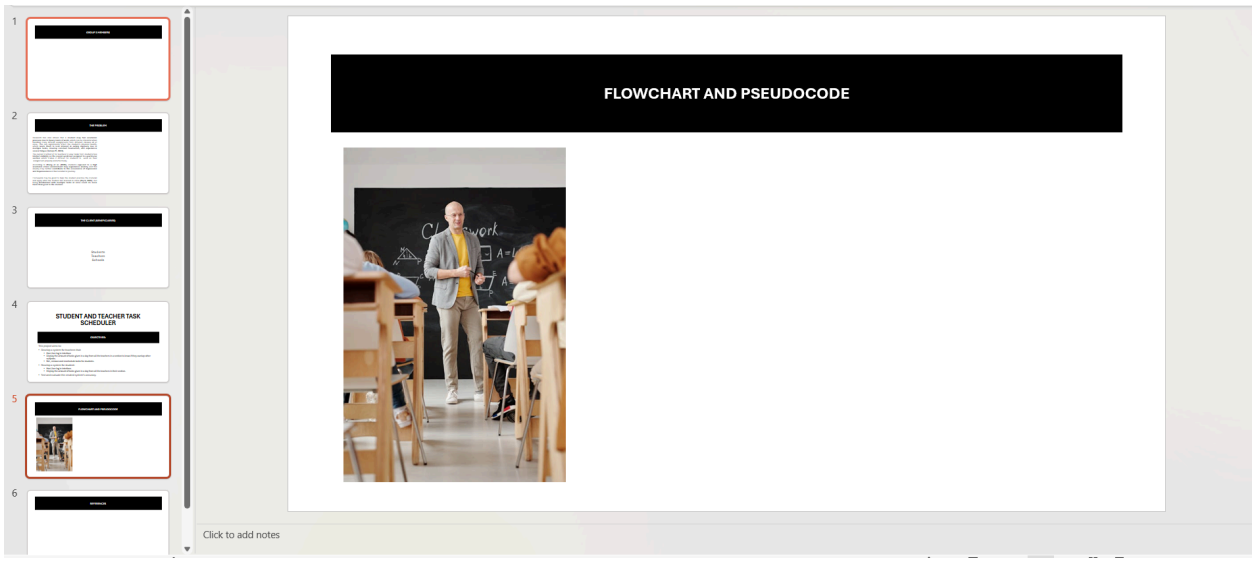
Lisud, Ian Paulo S - Improved the login system and menu to look more appealing.

Lopez, Andrei Dion C – fixed some errors and improved the login system menus that are dynamically interactive and helped with modularization of the code.

FINAL PROJECT GROUP 3 PROGRESS REPORT

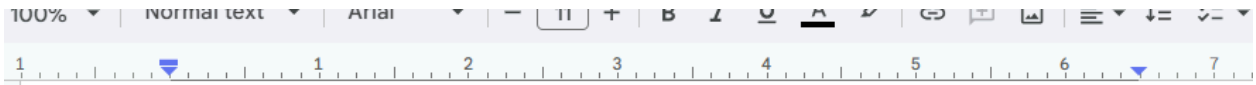
PROOF OF PROGRESS

PowerPoint Progress(Braganza)



FINAL PROJECT GROUP 3 PROGRESS REPORT

References/Documentation (Condino/Braganza)



<conclude results and discussions, add recommendations>

References

<APA Format Alphabetical Order>

REFERENCE (NOT IN ORDER AND NOT IN APA YET)
Lcsw, M. V. (2025, May 28). *Academic pressure: causes, effects, and coping strategies*. Mental Health Center Kids. https://mentalhealthcenterkids.com/blogs/articles/academic-pressure#:~:text=Excessive%20levels%20of%20academic%20pressure,to%20ensure%20their%20well%2Dbeing.

Academic Pressure: Causes, Effects, and Coping Strategies – Mental Health Center Kids

2. Associations Between Academic Stress and Depressive Symptoms Mediated by Anxiety Symptoms and Hopelessness Among Chinese College Students - PMC

3. How to Focus on Homework to Get It Done on Time | American Public University

4. The Importance of Homework in Students' Life - 21K School

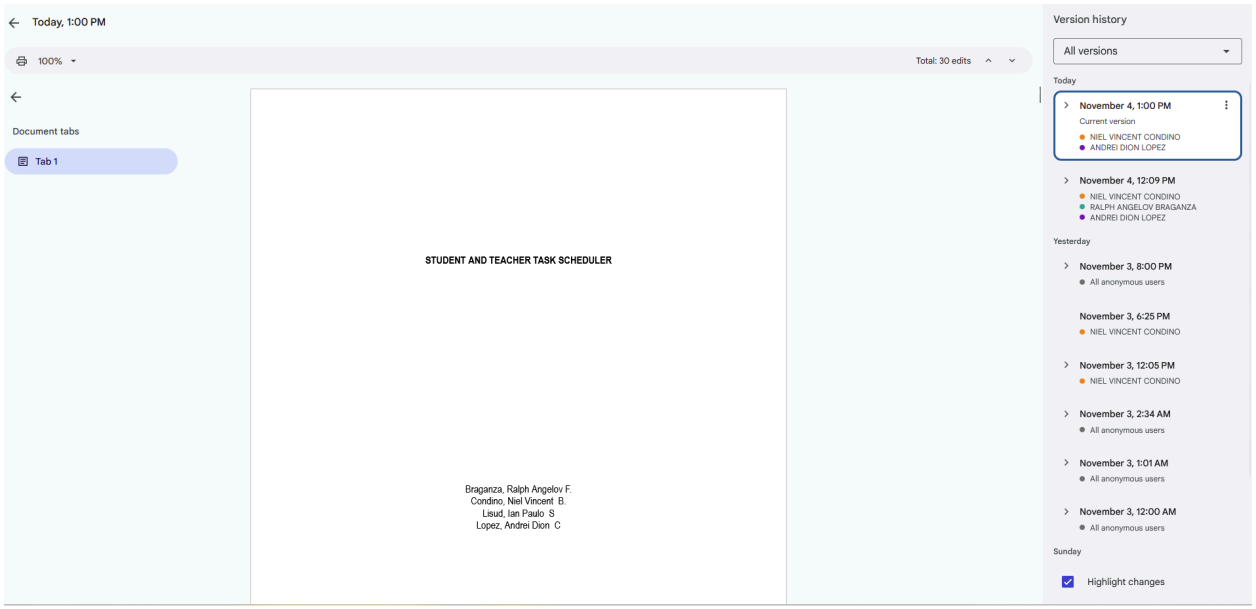
5. How Homework Affects Students: The Pros And Cons | NSHSS | National Society of High School Scholars

IMAGES:
Teacher Standing in Front of a Blackboard · Free Stock Photo

https://docs.google.com/document/d/1btWejPXBIf97OL9f-pWholXbwmrXW_uBohhYvY_ko8Q/edit?usp=sharing

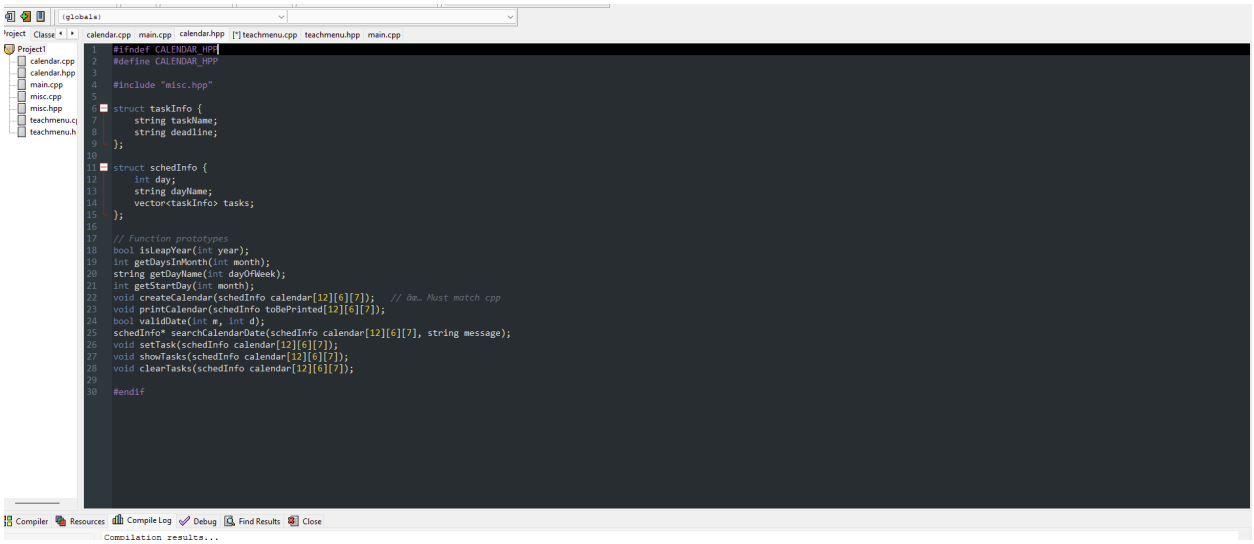
249 for Center dot

FINAL PROJECT GROUP 3 PROGRESS REPORT



Code Progress (Condino/Lisud/Lopez)

FINAL PROJECT GROUP 3 PROGRESS REPORT



Code:

```cpp
#include "calendar.hpp"
#include "menuGUI.hpp"
#include "credentials.hpp"

// ------- Task and Calendar Info ------- //

schedInfo mainCalendar[12][6][7];

// ------- Task CRUD ------- //

int main() {
    createCalendar(mainCalendar);
    opening();
    return 0;
}
```

Figure 3. Code of the file named “main.cpp”.

```cpp
#include "calendar.hpp"

/*
=================================
Calendar creation essentials
=================================
*/

const int year = 2025;
```

FINAL PROJECT GROUP 3 PROGRESS REPORT

```
// Check for leap year
bool isLeapYear(int year) {
    return (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);
}

// Get number of days in a given month
int getDaysInMonth(int month) {
    const int daysInMonth[12] = {31,28,31,30,31,30,31,31,30,31,30,31};
    // February
    if (month == 1 && isLeapYear(year)){
        return 29;
    }
    return daysInMonth[month];
}

string getDayName(int dayOfWeek){
    switch (dayOfWeek){
        case 0:
            return "Sunday";
        case 1:
            return "Monday";
        case 2:
            return "Tuesday";
        case 3:
            return "Wednesday";
        case 4:
            return "Thursday";
        case 5:
            return "Friday";
        case 6:
            return "Saturday";

    }
    return " ";
}

// Compute which weekday the month starts on (0 = Sunday, 6 = Saturday)
int getStartDay(int month) {
    // Zellerâ€™s congruence from online
    int m = month + 1;
    int y = year;
```

FINAL PROJECT GROUP 3 PROGRESS REPORT

```
    if (m < 3) { m += 12; y--; }
    int q = 1;
    int k = y % 100;
    int j = y / 100;
    int h = (q + (13*(m + 1))/5 + k + (k/4) + (j/4) + (5*j)) % 7;
    return (h + 6) % 7; // 0 is sunday ,6 is saturday
}

void createCalendar(schedInfo calendar[12][6][7]){
    // 12 months, 6 weeks ,7 days

    for (int month = 0; month < 12; month++) {
        int startDay = getStartDay(month);
        int days = getDaysInMonth(month);

        int week = 0;
        int dayOfWeek = startDay;

        for (int day = 1; day <= days; day++) {
            schedInfo *currentDay = &calendar[month][week][dayOfWeek];
            currentDay -> day = day;
            currentDay -> dayName = getDayName(dayOfWeek);

            dayOfWeek++;

            if (dayOfWeek > 6) {
                dayOfWeek = 0;
                week++;// move to next week
            }
        }
    }
}

/*
============================================================
Calendar utilization
============================================================
*/

void printCalendar(schedInfo toBePrinted[12][6][7]) {
    for (int month = 0; month < 12; ++month) {
        cout << "Month " << month + 1 << ":\n";
```

```
        cout << "Su  Mo  Tu  We  Th  Fr  Sa\n";

        for (int week = 0; week < 6; week++) {
            bool emptyWeek = true;
            for (int day = 0; day < 7; day++) {
                if (toBePrinted[month][week][day].day != 0){
                    emptyWeek = false;
                }
            }

            if (emptyWeek) break; // skip empty weeks at end

            for (int day = 0; day < 7; day++) {
                if (toBePrinted[month][week][day].day == 0){
                    cout << setw(4) << " ";  // keep spacing consistent
                }
                else{
                    if (toBePrinted[month][week][day].tasks.empty()){
                        cout << setw(4) <<
toBePrinted[month][week][day].day;//bold
                    }
                    else {
                        cout << setw(4) << "[" +
to_string(toBePrinted[month][week][day].day) + "]" ;
                    }

                }
            }
            cout << "\n";
        }
        cout << "\n";
    }
}

bool validDate(int m,int d){
    if ((m > 12 || m <= 0) || (d > getDaysInMonth(m-1) || d <= 0)){
        return false;
    }
    return true;
}

schedInfo* searchCalendarDate(schedInfo calendar[12][6][7],string message){
```

```
    int m,d;
    bool isDateValid;
    do {
        cout << message;
        cout << "Month:";
        cin >> m;
        cout << "Day: ";
        cin >> d;
        isDateValid = validDate(m,d);
        if (!isDateValid){
            cout << "Invalid date\n";
        }
        cin.ignore();
    }while(!isDateValid);

    for (int i = 0; i < 6; i++){
        for (int j = 0; j < 7;j++){
            if (calendar[m][i][j].day == d){
                return &calendar[m][i][j];
            }
        }
    }
    return nullptr;
}

/*
================================================================
CRUD utility
================================================================
*/

void setTask(schedInfo calendar[12][6][7]){
    int countTasks;

    schedInfo* chosenDate = searchCalendarDate(calendar,"Select month and
day for deadline:\n");
    cout << "Enter task amount: ";
    cin >> countTasks;
    cin.ignore();

    for (int i = 1; i <= countTasks;i++){
        cout << endl;
```

```
        string taskName;
        cout << "Enter task name: ";
        getline(cin,taskName);
        string deadlineTime;
        cout << "Enter deadline time: ";
        getline(cin,deadlineTime);
        chosenDate -> tasks.push_back({taskName,deadlineTime});
        cout << endl;
    }
    cout << "\n=================== Task Sucessfully Added
====================\n";
}

void showTasks(schedInfo calendar[12][6][7]){
    schedInfo* chosenDate = searchCalendarDate(calendar,"Show tasks from
the date...\n");
    cout << "Available Task " <<" (" << chosenDate->dayName << "):\n";
    if (chosenDate->tasks.empty()) {
        cout << "  No tasks scheduled.\n";
    } else {
        for (const auto &task : chosenDate->tasks) {
            cout << "  - " << task.taskName << " (Deadline time: " <<
task.deadline << ")\n";
        }
    }
}

void clearTasks(schedInfo calendar[12][6][7]) {
    schedInfo* chosenDate = searchCalendarDate(calendar, "Select task to be
deleted\n");
    if (chosenDate->tasks.empty()) {
        cout << "  No tasks scheduled.\n";
        return;
    }
    for (size_t i = 0; i < chosenDate->tasks.size(); i++)
        cout << "(" << i + 1 << ") " << chosenDate->tasks[i].taskName << "
(Deadline: " << chosenDate->tasks[i].deadline << ")\n";

    cout << "\n To be deleted(if multiple, seperate each with comma(,) and
no spaces.Example :1,2) : ";
    string tbDeleted;
    getline(cin, tbDeleted);
```

```
    stringstream ss(tbDeleted);
    string temp;
    vector<int> toDelete;
    while (getline(ss, temp, ',')) {
        int idx = stoi(temp);
        if (idx >= 1 && idx <= (int)chosenDate->tasks.size()){
            toDelete.push_back(idx - 1);//collect the index
        }
    }

    // delete in reverse order
    for (int i = toDelete.size() - 1; i >= 0; i--){
        chosenDate->tasks.erase(chosenDate->tasks.begin() + toDelete[i]);
    }
    cout << "Deleted selected tasks.\n";
}
```

```
#include "menuGUI.hpp"

// ------- Headers ------- //
void printTMenu() {
    printMenu("TEACHER MENU",teachMenu,teachMENU_ITEMS);
}

void printSMenu() {
    printMenu("STUDENT MENU",studentMenu,studentMENU_ITEMS);
}

// ------- Selection Menu ------- //

void teachMain(){
    int selected = 0;
    bool isRunning = true;
    while (isRunning){
        printTMenu();
        int selected = selection(teachMENU_ITEMS,printTMenu);
        switch (selected){
```

```
            case 4:
                system("CLS");
                isRunning = false;
                opening();
                break;
            case 0:
                setTask(mainCalendar);
                wait(2);
                break;
            case 1:
                showTasks(mainCalendar);
                wait(2);
                break;
            case 2:
                clearTasks(mainCalendar);
                wait(2);
                break;
            case 3:
                printCalendar(mainCalendar);
                wait(10);
                break;
        }
    }
}

void studentMain(){
    int selected = 0;
    bool isRunning = true;
    while (isRunning){
        printSMenu();
        int selected = selection(studentMENU_ITEMS,printSMenu);
        switch (selected){
            case 2:
                system("CLS");
                isRunning = false;
                opening();
                break;
            case 0:
                showTasks(mainCalendar);
                wait(2);
                break;
            case 1:
```

```
                printCalendar(mainCalendar);
                wait(10);
                break;
        }
        break;
    }
}
```

```
#include "credentials.hpp"

struct Credentials {
    string username;
    string password;
};

Credentials studentCreds[credSize];
Credentials teachCreds[credSize];

// ------- Bring back to homescreen function -------//
void backToHome(string message){
    cout << message;
    wait(3);
    opening();
}

// ------- Credential Search Function ------- //

void searchCreds(string username, string password, int mode) {
    if (username.empty() || password.empty()) {
        backToHome("Credentials cannot be empty!");
    }
    bool found = false;
    for (int i = 0; i < credSize; i++) {
        if (mode == 2) {
            if (studentCreds[i].username == username) {
                found = true;
                if (studentCreds[i].password == password) {
                    studentMain();
```

```
                } else {
                    backToHome("Incorrect username or password");
                    wait(3);
                    opening();
                }
                break;
            }
        } else {
            if (teachCreds[i].username == username) {
                found = true;
                if (teachCreds[i].password == password) {
                    teachMain();
                } else {
                    backToHome("Incorrect username or password");
                }
                break;
            }
        }
    }
    if (!found) {
        backToHome("Account does not exist");
    }
}

// ------- Login Function ------- //

void login() {
    system("CLS");
    printHeader("LOGIN");
    int mode = 0;
    string username;
    string password;

    cout << "Log-in\n";
    cout << "Username: ";
    getline(cin, username);
    cout << "Password: ";
    getline(cin, password);

    isNumInRange("Login as teacher(1) or as a student(2)\n", mode, 1, 2);

    searchCreds(username, password, mode);
```

```
}

// ------- Registration Functions ------- //

void pushToArray(string username, string password, int mode) {
    if (username.empty() || password.empty()) {
    backToHome("Credentials cannot be empty!");
    }
    for (int i = 0; i < credSize; i++) {
        if (mode == 2) {
            if (studentCreds[i].username == username) {
                backToHome("username already exists");
            }
            if (studentCreds[i].username.empty()) {
                studentCreds[i].username = username;
                studentCreds[i].password = password;
                return;
            }
        } else {
            if (teachCreds[i].username == username) {
                backToHome("username already exists");
            }
            if (teachCreds[i].username.empty()) {
                teachCreds[i].username = username;
                teachCreds[i].password = password;
                return;
            }
        }
    }
}

void registerCreds() {
    system("CLS");
    printHeader("REGISTER");
    int mode = 0;
    string username, password;

    cout << "Register\n";
    cout << "Username: ";
    getline(cin, username);
    cout << "Password: ";
    getline(cin, password);
```

```
    isNumInRange("Register as teacher(1) or as a student(2)\n", mode, 1,
2);

    pushToArray(username, password, mode);
    //login();
    opening();
}

// ------- Main selection screen ------- //

void printOpeningMenu() {
    printMenu("School Task Scheduler", openMenuItems, openMenuSize);
}

void opening() {
    int selected = 0;
    printOpeningMenu();
    selected = selection(openMenuSize, printOpeningMenu);
    switch (selected) {
        case 0: registerCreds(); break;
        case 1: login(); break;
        case 2: exitProg(); break;
    }
}
```