Progress Report

- The code has been finished and is fully functional. It is currently undergoing code structure design and will have more changes but no guarantee of any major changes to the code.
- The flowchart and pseudo code are adjusting in accordance to the current code structure that is used in creating the program itself.

```
#include <iostream>
#include <string>
#include <cstdlib>
#include <cctype>
#include <ctime>
using namespace std;

#define MAX_ITEMS 50
#define MAX_HISTORY 100
#define MAX_USERS 5 // number of allowed student IDs

//------------------------------------ DATA STRUCTURE FOR CLAIM
HISTORY ------------------------------------//

struct Data {
string gmail;
string cpnum;
string itemClaimed;
string dateClaimed;
};

//------------------------------------ HELPER FUNCTION TO LOWERCASE A
STRING ------------------------------------//
string toLowerStr(string s) {
for (int i = 0; i < (int)s.length(); i++) {
s[i] = tolower(s[i]);
}
return s;
}

//------------------------------------ FUNCTION TO GET CURRENT DATE
AND TIME ------------------------------------//
string getCurrentDateTime() {
```

```
time_t now = time(0);
tm *ltm = localtime(&now);

char buffer[50];
// Format: YYYY-MM-DD HH:MM:SS
sprintf(buffer, "%04d-%02d-%02d %02d:%02d:%02d",
1900 + ltm->tm_year,
1 + ltm->tm_mon,
ltm->tm_mday,
ltm->tm_hour,
ltm->tm_min,
ltm->tm_sec);
return string(buffer);
}

//------------------------------------ ITEM LOGGER CLASS
------------------------------------//

class ItemLogger {
private:
string accessories[MAX_ITEMS];
string electronics[MAX_ITEMS];
string clothes[MAX_ITEMS];
int accCount, elecCount, clothCount;

Data history[MAX_HISTORY];
int historyCount;

public:
ItemLogger() {
accCount = elecCount = clothCount = 0;
historyCount = 0;
}

void logItem(string section, string itemName) {
section = toLowerStr(section);
if (section == "accessories") {
if (accCount < MAX_ITEMS) accessories[accCount++] = itemName;
else cout << "Accessories section full!\n";
} else if (section == "electronics") {
if (elecCount < MAX_ITEMS) electronics[elecCount++] = itemName;
else cout << "Electronics section full!\n";
} else if (section == "clothes") {
if (clothCount < MAX_ITEMS) clothes[clothCount++] = itemName;
else cout << "Clothes section full!\n";
} else {
cout << "Invalid section name.\n";
return;
}
cout << "? \"" << itemName << "\" added under " << section << ".\n";
}

void claimItem(string section, string itemName) {
section = toLowerStr(section);
if (section == "accessories") {
claimFromSection(accessories, accCount, itemName, "Accessories");
} else if (section == "electronics") {
claimFromSection(electronics, elecCount, itemName, "Electronics");
} else if (section == "clothes") {
claimFromSection(clothes, clothCount, itemName, "Clothes");
} else {
cout << "Invalid section name.\n";
}
}

void displayLogs() const {
cout << "\n--- Current Item Logs ---\n";
cout << "Accessories:\n";
if (accCount == 0) cout << " (No items)\n";
for (int i = 0; i < accCount; i++) cout << " - " << accessories[i] <<
"\n";

cout << "Electronics:\n";
if (elecCount == 0) cout << " (No items)\n";
for (int i = 0; i < elecCount; i++) cout << " - " << electronics[i]
<< "\n";

cout << "Clothes:\n";
```

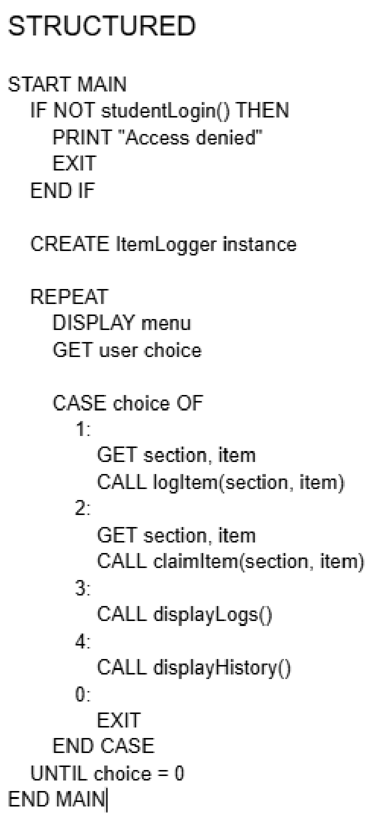
```
if (clothCount == 0) cout << " (No items)\n";
for (int i = 0; i < clothCount; i++) cout << " - " << clothes[i] <<
"\n";
}

void displayHistory() const {
cout << "\n--- Claimed Item History ---\n";
if (historyCount == 0) {
cout << "(No claimed items)\n";
return;
}
for (int i = 0; i < historyCount; i++) {
cout << "[" << i + 1 << "] Gmail: " << history[i].gmail
<< "\nMobile #: " << history[i].cpnum
<< "\nItem Claimed: " << history[i].itemClaimed
<< "\nDate & Time Claimed: " << history[i].dateClaimed << "\n\n";
}
}

private:
void claimFromSection(string arr[], int& count, const string&
itemName, const string& sectionName) {
int foundIndex = -1;
for (int i = 0; i < count; i++) {
if (toLowerStr(arr[i]) == toLowerStr(itemName)) {
foundIndex = i;
break;
}
}

if (foundIndex != -1) {
if (historyCount < MAX_HISTORY) {
// Ask claimant info
Data logData;
cout << "\n--- Claimant Information ---\n";
cout << "Enter Gmail: ";
cin >> logData.gmail;
cout << "Enter Mobile #: ";
cin >> logData.cpnum;
```

```cpp
logData.itemClaimed = arr[foundIndex];
logData.dateClaimed = getCurrentDateTime(); // Add timestamp

// Add to history log
history[historyCount++] = logData;
cout << "? \"" << arr[foundIndex] << "\" claimed successfully on " <<
logData.dateClaimed << ".\n";
}

// Remove claimed item from section
for (int j = foundIndex; j < count - 1; j++) {
arr[j] = arr[j + 1];
}
count--;
} else {
cout << "? Item not found in " << sectionName << ".\n";
}
}
};

//------------------------------------- STUDENT LOGIN FUNCTION
-------------------------------------//

bool studentLogin() {
string validIDs[MAX_USERS] = {"202501", "202502", "202503", "202504",
"202505"};
string inputID;
cout << "\t|=== STUDENT LOGIN ===|\n";
cout << "\t| Enter Student ID: ";
cin >> inputID;

for (int i = 0; i < MAX_USERS; i++) {
if (inputID == validIDs[i]) {
cout << "---------------------------------------------\n";
cout << "? Login successful! Welcome, Student " << inputID << ".\n";
cout << "---------------------------------------------\n";
return true;
}
}
```

```
cout << "? Invalid Student ID! Access denied.\n";
return false;
}

//------------------------------------ MAIN MENU
------------------------------------//

int main() {
if (!studentLogin()) {
cout << "\nProgram terminated due to invalid login.\n";
return 0;
}

ItemLogger logger;
int choice;
string section, item;

do {
cout << "\n --------------------------";
cout << "\n ==== ITEM LOGGER MENU ====\n";
cout << " --------------------------\n";
cout << "[1.] | Log New Item |\n";
cout << "[2.] | Claim Item |\n";
cout << "[3.] | Show Current Logs |\n";
cout << "[4.] | Show Claim History |\n";
cout << "[0.] | Exit |\n";
cout << "---------------------------\n";
cout << "Choose an option: ";
cin >> choice;
cout << "---------------------------\n";
cin.ignore();

system("CLS"); // clear screen (optional)

switch (choice) {
case 1:
cout << "------------------------------------------\n";
cout << " ===============| Section |===============\n"
<< "------------------------------------------\n"
<< " |( Accessories | Electronics | Clothes )|\n";
cout << "------------------------------------------\n Enter Section: 
";
getline(cin, section);
cout << "------------------------------------------\n";
cout << "Enter item name: ";
getline(cin, item);
cout << "------------------------------------------\n";
logger.logItem(section, item);
break;

case 2:
cout << "Enter section: ";
getline(cin, section);
cout << "Enter item name to claim: ";
getline(cin, item);
logger.claimItem(section, item);
break;

case 3:
logger.displayLogs();
break;

case 4:
logger.displayHistory();
break;

case 0:
cout << "Exiting program...\n";
break;

default:
cout << "Invalid choice!\n";
}

} while (choice != 0);

return 0;
```

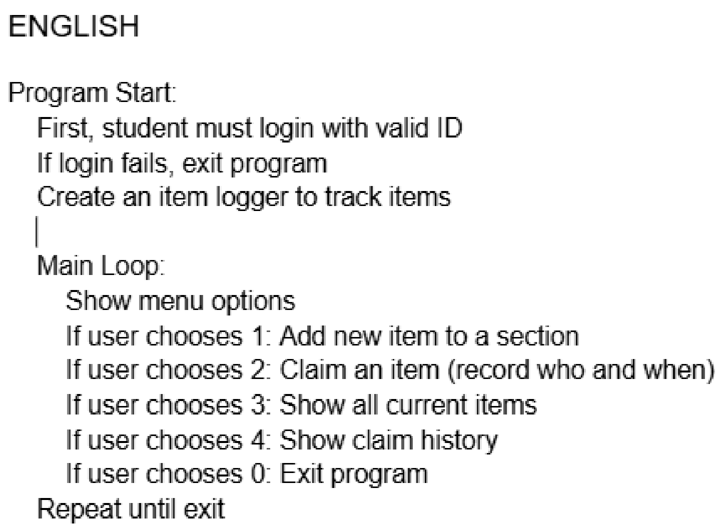
Pseudocode and Flowchart

- The current pseudocode is adjusting to the changes and any possible new changes to the code and its structure



STRUCTURED

```
START MAIN
  IF NOT studentLogin() THEN
    PRINT "Access denied"
    EXIT
  END IF

  CREATE ItemLogger instance

  REPEAT
    DISPLAY menu
    GET user choice

    CASE choice OF
      1:
        GET section, item
        CALL logItem(section, item)
      2:
        GET section, item
        CALL claimItem(section, item)
      3:
        CALL displayLogs()
      4:
        CALL displayHistory()
      0:
        EXIT
    END CASE
  UNTIL choice = 0
END MAIN
```

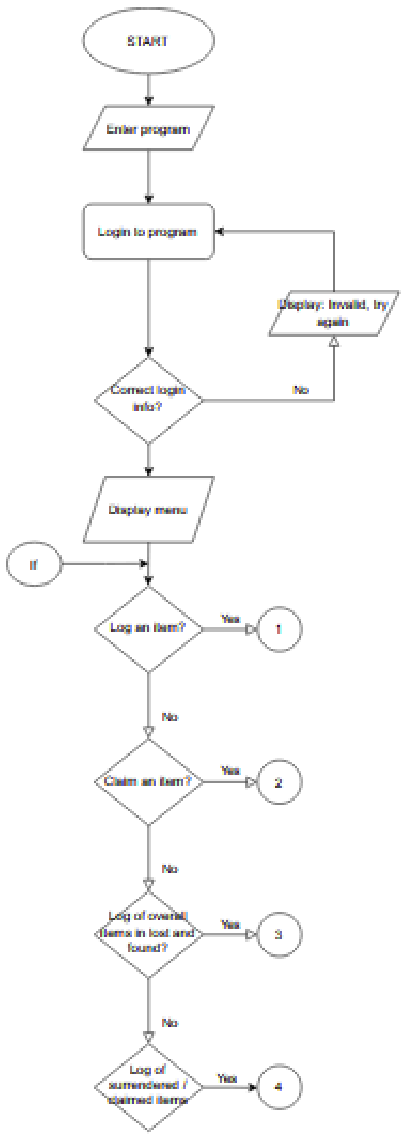


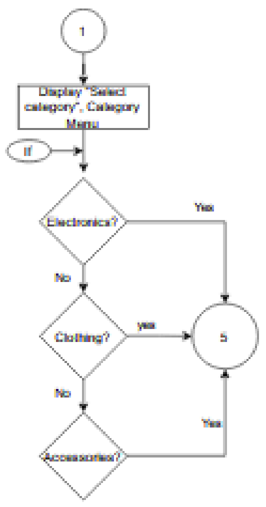
ENGLISH

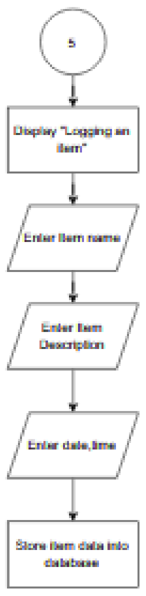
```
Program Start:
  First, student must login with valid ID
  If login fails, exit program
  Create an item logger to track items

  Main Loop:
    Show menu options
    If user chooses 1: Add new item to a section
    If user chooses 2: Claim an item (record who and when)
    If user chooses 3: Show all current items
    If user chooses 4: Show claim history
    If user chooses 0: Exit program
  Repeat until exit
```

- In the pseudocode De Vera has created 2 versions of it following the structured pseudo code and English based pseudo code.

- The flowchart is also adjusting to the changes and structure of the current code. (I am not able to get new pictures of the updated and finished code as my groupmate is on site right now.)


START
Enter program
Login to program
Display: Invalid, try again
Correct login info?
No
Display menu
If
Log an item?
Yes
1
No
Claim an item?
Yes
2
No
Log of overall items in lost and found?
Yes
3
No
Log of surrendered / claimed items
Yes
4


1
Display "Select category", Category Menu
If
Electronics?
Yes
No
Clothing?
yes
5
No
Accessories?
Yes


5
Display "Logging an item"
Enter item name
Enter item Description
Enter date,time
Store item data into database

- Accomplishment timeline has been allotted.
⇒ Contribution Report:
- Programming and coding structure ⇒ Verano
- Pseudocode ⇒ De Vera
- Flowchart ⇒ Baula

Conclusion: So far the code has been finished and undergoing structural design, The pseudocode and flowchart are prepared for any possible changes that may happen to the structural code.