SoftArchMidterm

Estimations:

- Scaffolding: 3 hours
- Localization: 3 hours
- Digital signature: 1 hour

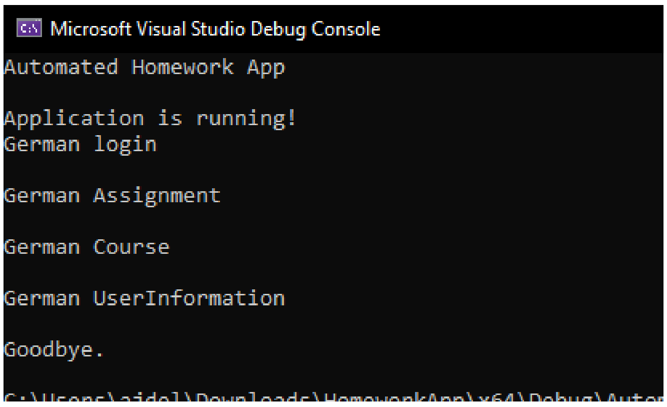
Actual Time Spent:

- Scaffolding: 3 hours, we were pretty accurate with this estimate and it took about as long as expected. Maybe a little less time, but pretty close.
- Localization: 2 hours, this took a little less time than expected. It definitely helped that we already hasd all the tools downloaded and installed, and the majority of our time was spent troubleshooting.
- Digital Signature: 30 minutes, we didn't expect this to take too long, and it went even faster than expected. Again I think it helped that we had recently done this for our homework, so we were prepared.

Description

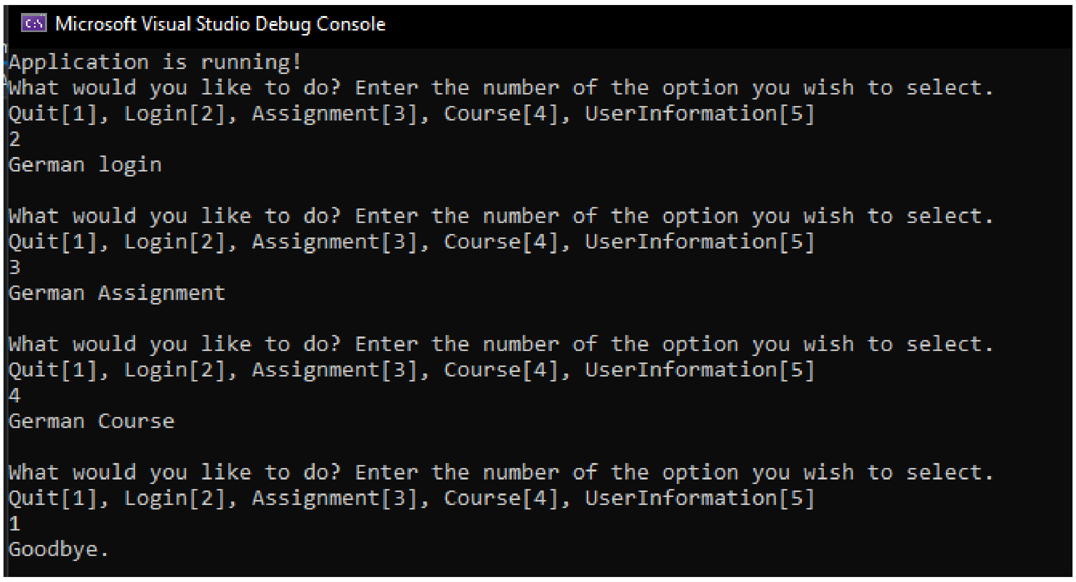
- HomeworkApp is a very simple project that contains several libraries that return a string containing the name of that library. This project is meant to demonstrate several of the topics we have covered in Software Architecture, such as libraries, localization, and digital signatures. Our project has two executables. One is called HomeworkApp.exe, and the other is called AutomatedHomeworkApp.exe. HomeworkApp.exe is an interactable program whereas AutomatedHomeworkApp.exe simply prints everything you need and is not interactable. If this app were to become real, it would be used to keep track of your homework assignments, the deadlines, which course they were for,

AutomatedHomeworkApp.exe



```
Microsoft Visual Studio Debug Console
Automated Homework App

Application is running!
German login

German Assignment

German Course

German UserInformation

Goodbye.

C:\Users\aidel\Downloads\HomeworkApp\x64\Debug\Autor
```

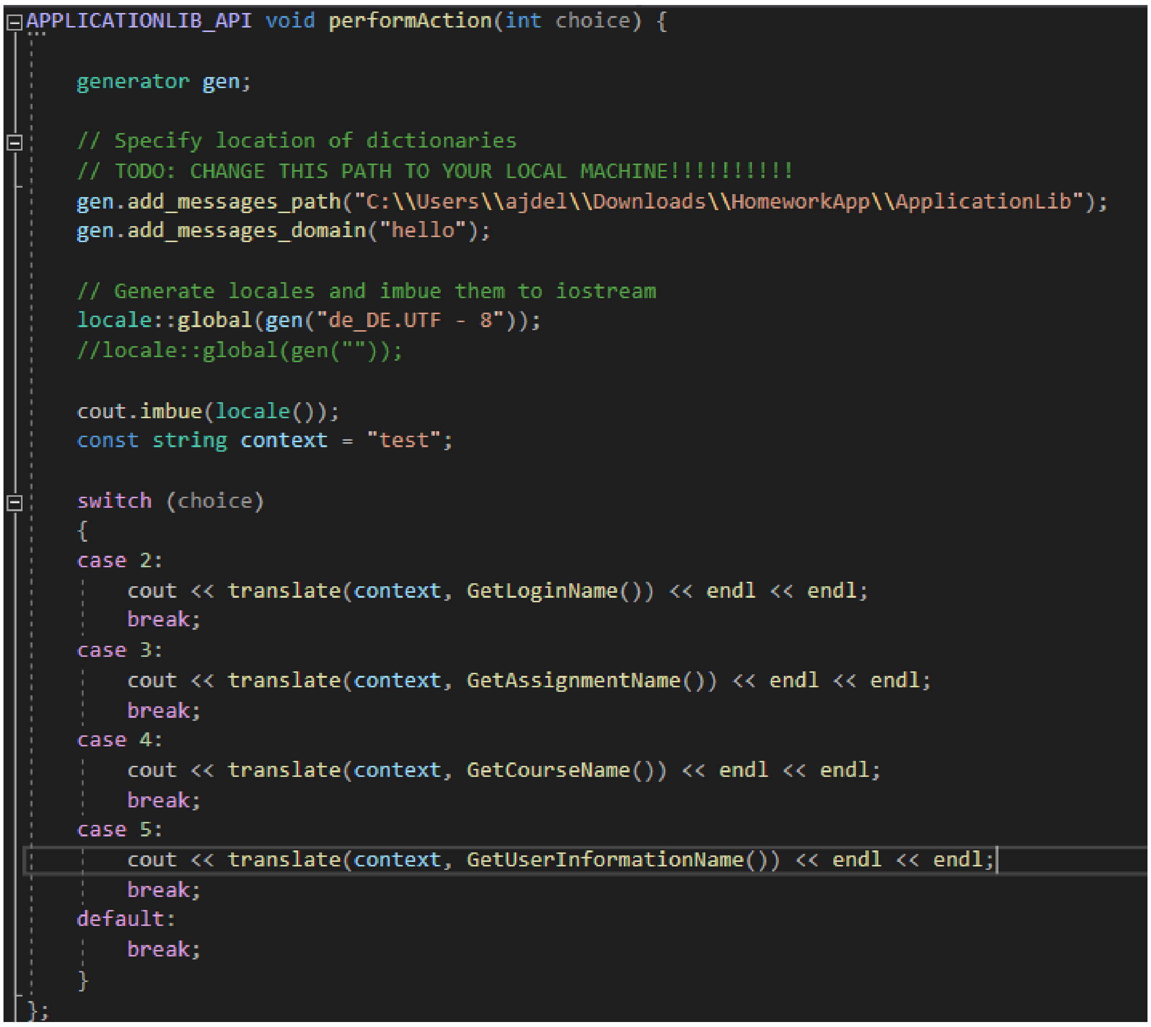
HomeworkApp.exe



```
Microsoft Visual Studio Debug Console
Application is running!
What would you like to do? Enter the number of the option you wish to select.
Quit[1], Login[2], Assignment[3], Course[4], UserInformation[5]
2
German login

What would you like to do? Enter the number of the option you wish to select.
Quit[1], Login[2], Assignment[3], Course[4], UserInformation[5]
3
German Assignment

What would you like to do? Enter the number of the option you wish to select.
Quit[1], Login[2], Assignment[3], Course[4], UserInformation[5]
4
German Course

What would you like to do? Enter the number of the option you wish to select.
Quit[1], Login[2], Assignment[3], Course[4], UserInformation[5]
1
Goodbye.
```

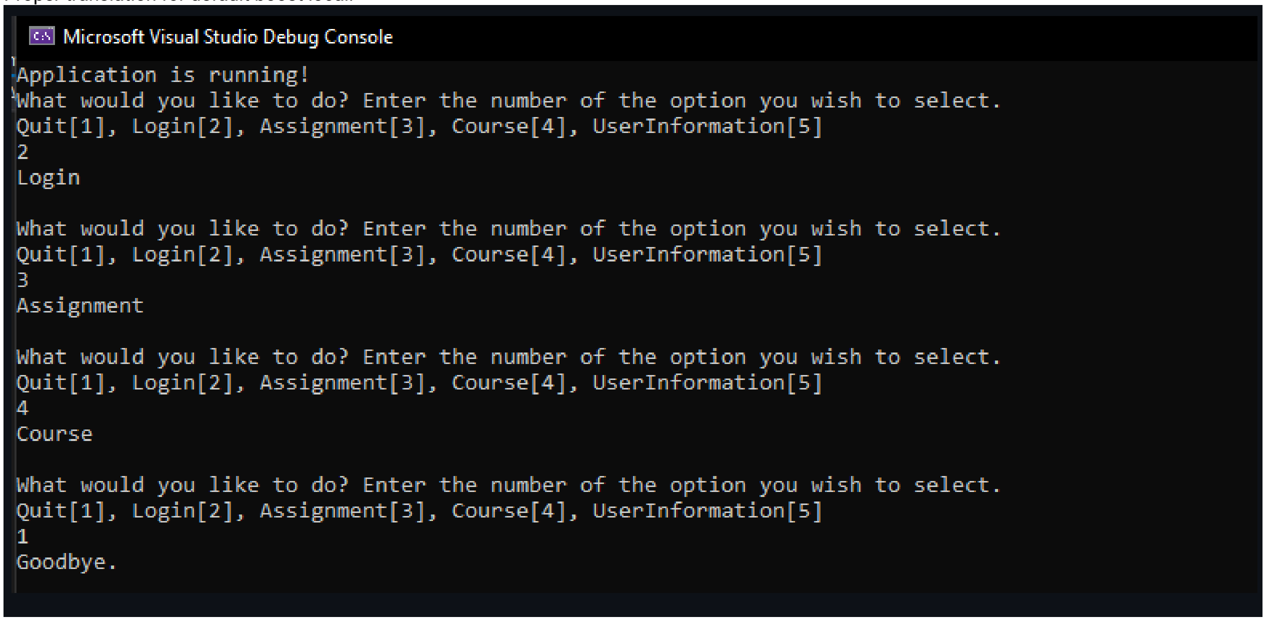
Boost

- Here is an example of how we use boost. As you can see we pass in a given context "test" and we are properly returned the translations for each string. This can be seen by the above demo. If we comment out the global gen local and uncomment the default local, we also get the proper translations. Example is below.



```cpp
APPLICATIONLIB_API void performAction(int choice) {

    generator gen;

    // Specify location of dictionaries
    // TODO: CHANGE THIS PATH TO YOUR LOCAL MACHINE!!!!!!!!!!
    gen.add_messages_path("C:\\Users\\ajdel\\Downloads\\HomeworkApp\\ApplicationLib");
    gen.add_messages_domain("hello");

    // Generate locales and imbue them to iostream
    locale::global(gen("de_DE.UTF - 8"));
    //locale::global(gen(""));

    cout.imbue(locale());
    const string context = "test";

    switch (choice)
    {
    case 2:
        cout << translate(context, GetLoginName()) << endl << endl;
        break;
    case 3:
        cout << translate(context, GetAssignmentName()) << endl << endl;
        break;
    case 4:
        cout << translate(context, GetCourseName()) << endl << endl;
        break;
    case 5:
        cout << translate(context, GetUserInformationName()) << endl << endl;
        break;
    default:
        break;
    }
};
```

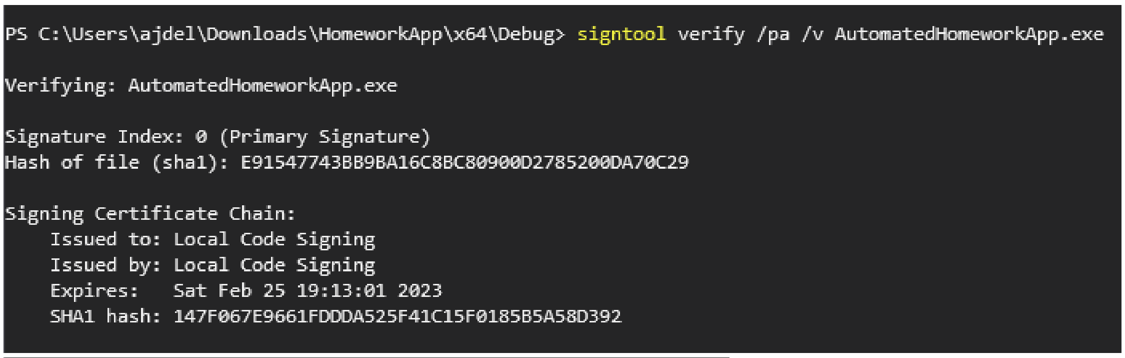
- Proper translation for default boost local:



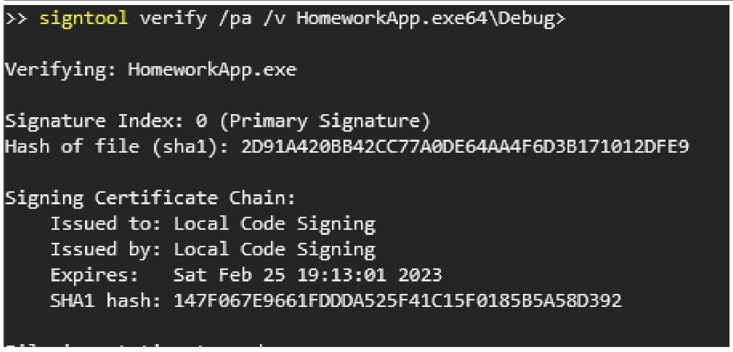
Microsoft Visual Studio Debug Console

```
Application is running!
What would you like to do? Enter the number of the option you wish to select.
Quit[1], Login[2], Assignment[3], Course[4], UserInformation[5]
2
Login

What would you like to do? Enter the number of the option you wish to select.
Quit[1], Login[2], Assignment[3], Course[4], UserInformation[5]
3
Assignment

What would you like to do? Enter the number of the option you wish to select.
Quit[1], Login[2], Assignment[3], Course[4], UserInformation[5]
4
Course

What would you like to do? Enter the number of the option you wish to select.
Quit[1], Login[2], Assignment[3], Course[4], UserInformation[5]
1
Goodbye.
```

Digital signing

- 

```
PS C:\Users\ajdel\Downloads\HomeworkApp\x64\Debug> signtool verify /pa /v AutomatedHomeworkApp.exe

Verifying: AutomatedHomeworkApp.exe

Signature Index: 0 (Primary Signature)
Hash of file (sha1): E91547743BB9BA16C8BC80900D2785200DA70C29

Signing Certificate Chain:
    Issued to: Local Code Signing
    Issued by: Local Code Signing
    Expires:   Sat Feb 25 19:13:01 2023
    SHA1 hash: 147F067E9661FDDDA525F41C15F0185B5A58D392
```

- 

```
>> signtool verify /pa /v HomeworkApp.exe64\Debug>

Verifying: HomeworkApp.exe

Signature Index: 0 (Primary Signature)
Hash of file (sha1): 2D91A420BB42CC77A0DE64AA4F6D3B171012DFE9

Signing Certificate Chain:
    Issued to: Local Code Signing
    Issued by: Local Code Signing
    Expires:   Sat Feb 25 19:13:01 2023
    SHA1 hash: 147F067E9661FDDDA525F41C15F0185B5A58D392
```

Licensing Questions:

a. *If we are consider using a toolkit like FlexNet (aka FlexLm), where would we put the calls to initialize the license server? Would this approach be too heavy handed for your application, if yes why?* The calls to initialize the server would go in the core library, so that way as soon as the app was started the license server would be active. I do believe this would be too heavy handed for our application, as it does not need to be this controlled. There aren't really any hidden features or special proprietary code. It would take a lot to fully integrate this into our code, and propbably wouldn't be worth it.

b. *Another alternative, could be an DRM like approach, where some type of hash\key is stored on the computer that indicates what is allowed. Would this approach be better.*
This is like the model of steam, entering in a product key, and so on. This approach would make much more sense for our application. It is lighter, and we could offer our core features for free while charging to use the rest. That way we are only locking down and charging for some of our application.

c. *If your boss\product manager told you to think about having multiple licenses for different functionality (instead of monolithic, but more atomic). How you would you suggest to organizing the entitlements (i.e. what pieces of functionality to have against what license)?* I think the types of licences that would make sense for our application would be perpetual or subscription, and give the user the option of having lifetime access or monthly access. Both of these would give the user full access to the entire application. I also think Add-ons would make sense, so the user can use the core features of the app for free, but then be able to buy access to other more advanced features if they want to use them.