# Assignment Helper
# Empowering Your Programming Skill
# Power OVERWHELMING co.

Ji-hoon Lee*, Jae-gook Kim†, Kyung-min Kim‡, and Kyo-ho Lee§
†‡§Department of Information System
*Department of Finance, Business School
Hanyang University, Seoul, Korea
Email: *starypoc@hanyang.ac.kr, †claretta@hanyang.ac.kr, ‡kimkmhk@hanyang.ac.kr, §1equal2@hanyang.ac.kr

***Abstract*—Assignment Helper is a program to help people who have difficulties with their programming language assignments. Assignment Helper will able to help the assignment by crawling the data from webs such as programmer forums and then find similar questions and related codes. It will also have a verification function and will check the codes by using CRC or other compilers. It shall check it by testing the input, output and the expectation all together.**

TABLE I
ROLE ASSIGNMENT

| Role | Name | Task Description |
|---|---|---|
| Developer Manager | Kim Jae-gook | Managing whole process of developing the program. |
| Users | Lee Ji-hoon | Seeking for the usefulness compare to other similar programs. |
| Cutomers | Kim Kyung-min | Finding out whether the project is valuable. |
| Developer | Lee Kyu-ho | Implementing the program. |

## I. INTRODUCTION

### A. Motivation

We have started this project with the motivation to help students (especially freshman) who are struggling with their assignments. Other assignments like history or natural science can easily find their assignment sources quickly from just googling it simply. However, it is impossible to do that simple procedure in programming language assignment. Without perfect knowledge about the computer languages, it comes out difficult to solve their programming language assignment.

We discussed about a solution to solve this. We concluded that it is not easy to study programming languages especially to the students who studied only at school and have never lived closed to computer languages; The students who only studied subjects for KSAT, the entrance exam for university!

### B. Problem Statement

It is problematic for those who first met the programming language and getting no additional help. It can be not only unfair but also can cause inefficiency in terms of the social resource allocation. Maybe, some novice students might have natural talents at computer programming. But others needs opportunities to catch up. So, we are going to build a program to ease the life of these novice students.

### C. Research on Any Related Software

Our program aims for helping students who is unfamiliar with programming paradigms. There are some services which provide help to those students in primitive levels than our service.

*1) Education Service: Scratch:* There are many programs for teaching programing language, but Scratch is the most representative one. Scratch does not execute commands by entering code with long instructions. Instead, click and drag the block to move the feature to perform the command. Scratch is a popular child-coding tool for schools and institutions worldwide. Scratch's intuitive interface is useful but has difficulty to find right answer. If one wants to make a program working properly. He should keep try until it matches. It is very frustrating for those who want to know the correct answer right away.

*2) Easier Coding: Emmet:* Emmet is a programming language that makes it easy to write CSS, even though it can save your coding time, the language you should learn for one programming language increases. Also, if you are not aware of the language, it is hard to get used to it. Instead, Assignment Helper will provide the coding style with natural language processing.

*3) Crowdsourcing and Knowledge Sharing: Stackoverflow:* Stackoverflow is a site where programmers ask and answers about programming. Stackoverflow is the largest developer community on the scale. It will be a good idea to ask questions here for the answers comes up very quickly. Since there are a lot of questions that has been answered, the problems that you need answers are mostly up on the forum already. In other words, rather than asking, you are more likely to search and get answers. But the point is for the beginners, it can even be hard to determine what to find. You must define the search keywords yourself. It would be a big hurdle for beginners. The limitation is that there is no recommend keywords.

### D. Distinguishable Features of Assignment Helper

There are countless amounts of services that tries to help the problem we defined. Our Assignment Helper will have differentiating features from existing services. Some distinguishable features are as follows:

*1) Lightening The Burden On Determining Search Keywords:* For some questions for beginners, it could be hard to determine a search keyword by themselves. For example, if they want to know what list object can 'do', it is appropriate to google on 'list method in python'. To google the word 'method', the user should know the word beforehand, which is practically impossible without any additional help.

Assignment Helper will allow to search on more natural-language-like queries, making the user easy to search with their actual thoughts.

*2) Grouping Related Questions:* With natural-language-like queries, it could be hard to guess what the user intended from the very beginning. Assignment Helper will explore broad range of possibilities and suggest each to the users.

*3) Getting the Working Codes Right Away from the Internet:* There are numerous codes that are floating on the internet. But it is not verified whether the codes on the internet works properly or not. Assignment Helper will check the validity of the code automatically

## II. Requirements

***Frontend Related***

### A. Building installation Manual

### B. Building Web-like Client

### C. Instruction for Users

- Description

### D. Building Search and Select UI

- Textbox for a query
- Cardbox to code select and check
  - Checkbox for choosing code
- Expandability
  - Can scroll infinitely (like Facebook) in case there are lot of result
- Show result code
  - Textbox for code lines

### E. Checking Code Validity

- Let users know the validity of the codes

### F. Providing Information Security

- Prevent from user clicking the button twice
- Immediate response to button can make system overload

***Backend Related***

### G. Search Query Processing

- Natural Language Processing Technique
  - Understanding the input with NLP
- DB for search keyword
  - Collecting keywords for seraching codes in DB to maximize accuracies
- Function to extract keyword from input comparing to DB for search keyword

### H. Learning System for Maximizing Accuracy

- Collect data from user and get feedback from them

### I. Crawling and Parsing

- Crawling data from website like programmer forum
- Parsing program codes
  - Parsing the codes using keywords for finding start-end point
  - ex) including ¡iostream¿, return 0 in c++

### J. Grammar Check

- DB for grammar rules
  - Building rules for testing the codes wheter valid or not
- Algorithm to check the grammar

### K. Testing Compile

- Compiling searched codes using CRC

## III. DEVELOPMENT ENVIRONMENT

### A. Choice of Software Development Platform

1) Operating System
   - OS: Windows
   - Reason: The program aims for students who are not familier with programming. They are highly likely to use Windows and have few knowledge on other platform such as Linux.

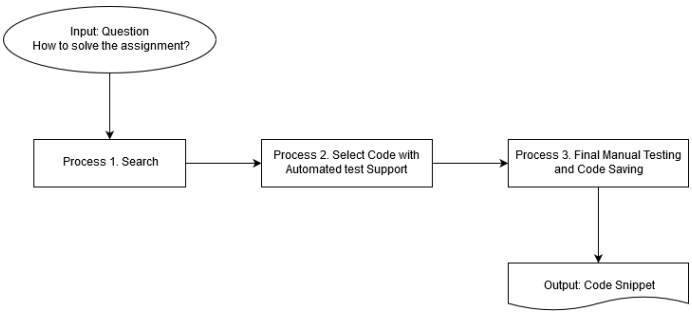
2) Language and Platform

- Language and Platform: Python 3.6 with PyQt4
- Reason: Easy to implement crawling and parsing dealing with text segments as well as leaving rooms to use other packages.

### B. Software in Use

1) *Sublime Text 3*: Text editor for general use
2) *qt designer 4.12*: GUI based GUI designer for Qt environment
3) *pyCharm*: genral IDE for Python
4) *pandas*: Python package to deal with datas
5) *PyQt4*: Python package to deal with Qt gui
6) *stackexchage REST API*: REST API to do stack overflow search and get datas
7) *beautifulsoup4*: Python package for parsing XML and HTML
8) *requests*: Python package that assists to issue an HTML Request
9) *gcc*: C compiler to verify given source codes

## IV. SPECIFICATIONS




Our targeted users do not know how to code.
Assignment Helper gives the code.

Fig. 1. Process Flow of Assignment Helper

### Getting Started

#### A. Setting - Easy Installation

1) Description
   - Executable without installation to facilitate ease of use
   - Works like portable utility
   - Basic files only provide barebone program
     - Programming language to search need be installed later
2) Process(or I/O)
   a) Download files from website

#### B. Setting - Configuration Window

1) Description
   - Basic program is not able to use without installing programming language
   - Provide check window
     - Check each window to install a language pack for a certain language
2) Process(or I/O)
   a) Input: Opening conf.exe
   b) Output: Show conf.exe

#### C. Setting - Installing Language Pack

1) Description
   - Basic program is not able to use without installing programming language
   - Provide check window
     - Check each window to install a language pack for a certain language
2) Process(or I/O)
   a) Input: Selecting language packs and proceed
   b) Output: Pack install on the hard disk

#### D. Instruction - README

1) Description
   - Provide README file to give basic instruction
2) Process(or I/O)
   a) N/A

#### E. Instruction - Basic Tutorial

1) Description
   - Give demo-like tutorial on the very first run
2) Process(or I/O)
   a) Input: Clicking tutorial on main window
   b) Input2: Initial program execution
   c) Output: Install selected pack on the hard disk

*F. Instruction - On the Fly*

1) Description
   - Give tooltips on buttons
2) Process(or I/O)
   a) Input: Hovering on a button for 3 seconds
   b) Output: Show up a tool tip

**Execution**

*G. Execution*

1) Description
   - Executing the main program
2) Process(or I/O)
   a) Opening the main program(helper.exe)
   b) Program(Process1. UI) shows up within 1 minute

*H. Closing Whole Program*

1) Description
   - Closing Whole Program
2) Process(or I/O)
   a) Clicking X window on search UI
   b) Program closing

**Process 1. Searching**

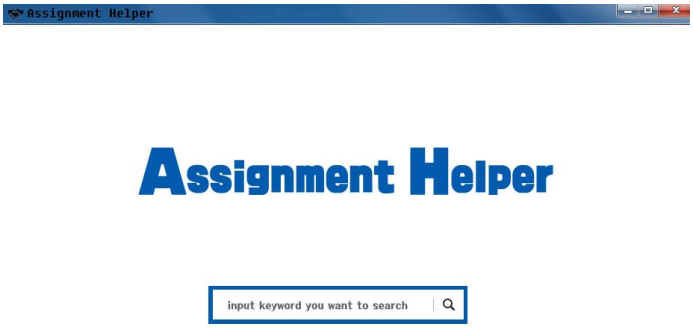



Fig. 2. Concept Image of Search Window

*I. Search - UI*

1) Description
   - Give instinctive search home
2) Process(or I/O)
   a) Clicking search button on search UI
   b) Search begins

*J. Search History*

1) Description
   - Show searh history
2) Process(or I/O)
   a) Output: Show up top 5 from search history list

*K. Search Option*

1) Description
   - Choose in what programming language result represent
     - Show language option buttons
2) Process(or I/O)
   a) Input: click on a python button
   b) Output: Show result on a python language
   c) Input: click on a C button
   d) Output: Show result on a C language
   e) Input: click on a C++ button
   f) Output: Show result on a C++ language

*L. Search Auto Completion*

1) Description
   - Show match from search ketwords.csv
2) Process(or I/O)
   a) Input: Each letters typed
   b) Output: Show all match words from search ketwords.csv

*M. Request Submission by Key Press*

1) Description
   - Submit request by typing enter
2) Process(or I/O)
   a) Input: type enter key
   b) Output: Submit request

*N. Request Submission by Clicking*

1) Description
   - Submit request by click
2) Process(or I/O)
   a) Input: mouse click
   b) Output: Submit request

*O. Request Submission - Waiting UI*

1) Description
   - Halt user interface while submission
2) Process(or I/O)
   a) input: Submit request
   b) Output: Freeze UI

*P. Request Submission - Abort*

1) Description
   - User press cancel
   - Error occurs
   - End submission
   - Back to search window
2) Process(or I/O)
   a) input: Click cancel button
   b) input: Error occurs
   c) Output: Stop submission
   d) Output: Back to start page

*Q. Request Submission  Extracting Keyword*

1) Description
   - Extract keywords from submitted
2) Process(or I/O)
   a) input: Submit request
   b) Output: Freeze UI

*R. Request Processing - Crawling(Stackoverflow)*

1) Description
   - Crawl popular codes
   - Organize codes by keywords
   - Process(or I/O)
   - Crawl Stackoverflow by keywords.csv
   - Give code with popularity over 3/5 to code page
   a) Make keyword a key
   b) Organize code page by keyword order

*S. Request Processing - Crawling(Google)*

1) Description
   - Crawl popular codes
   - Organize codes by keywords
   - Process(or I/O)
   - Crawl Google by keywords.csv
   - Give code with popularity over 3/5 to code page
   a) Make keyword a key
   b) Organize code page by keyword order

*T. Request Processing - Detect Code Part*

1) Description
   - Do encoding test to know if it is right code
2) Process(or I/O)
   a) Encode code page
   b) output: error code

*U. Request Completed - UI*

1) Description
   - Give code page to selection
2) Process(or I/O)
   a) Give all crawled code page to selection phase

**Process 2. Code Selction**

*V. Code Selection - Basic UI*

1) Description
   - User should select code among results before program starts other process
2) Process(or I/O)
   a) Clicking code among results
   b) Next process starts within 5 seconds

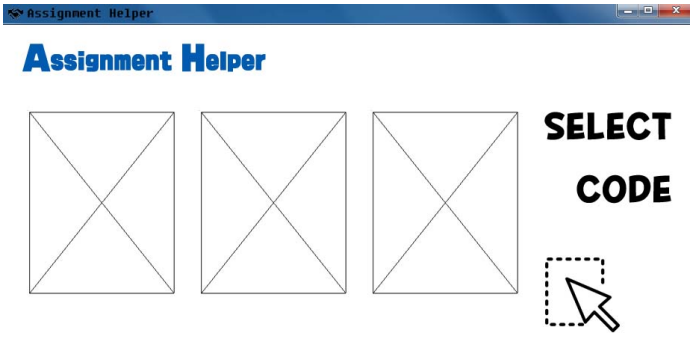



Fig. 3. Concept Image of Code Select Window

### W. Auto Compile Test - Requesting

1) Description
   - Right after user selects code, program is compiling the code(See Figure 4 )
2) Process(or I/O)
   a) Input : Text that user chose
   b) Output: Test. Program format


Fig. 4. Concept Image of Code Compiling

### X. Auto Compile Test - Success

1) Description
   - If compiling is finished successfully, without error code, code box goes green
2) Process(or I/O)
   a) Waiting for finish of compiling
   b) Check whether error code exists or not (No error code = Success)

### Y. Auto Compile Test - Failure

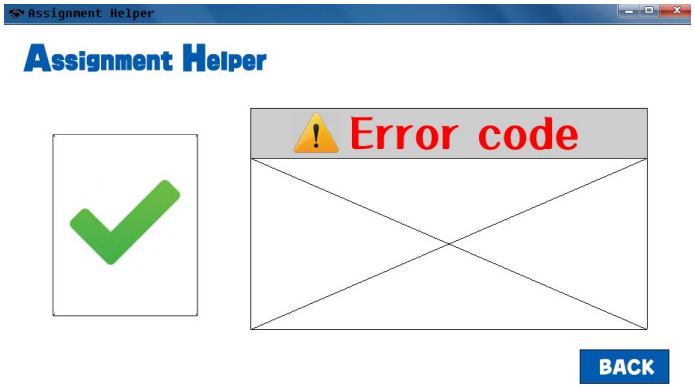



Fig. 5. Concept Image of Code Testing Failure

1) Description
   - If there is error code after compiling, program shows the error code to user (see Figure 5)
2) Process(or I/O)
   a) Waiting for finish of compiling
   b) Show error code result to user
   c) Enable users to edit codes if they can find the error.

### Z. Auto Compile Test - Edit Code Snippet

1) Description
   - When the auto compile test ended with error, users are able to edit code manually
2) Process(or I/O)
   a) Input: Error signal from auto compile test
   b) Output: Enabling users to edit code

### AA. Cancellation(2) - Clicking Return Button

1) Description
   - After finishing compiling or during compiling, user can go back with clicking return button
2) Process(or I/O)
   a) Clicking return button
   b) Go back to the code selection window

*AB. Cancellation(2) - Cliking X Window Button*

1) Description
   - After finishing compiling or during compiling, user can exit program
2) Process(or I/O)
   a) Clicking X window button
   b) Terminate program

*AC. Selection - Proceed without testing*

1) Description
   - Users are able to procced without auto compile test
   - Autocompile tested bits off
2) Process(or I/O)
   a) Users click next button
   b) Proceed to process 3

*AD. Prompt - Cancellation*

1) Description
   - When user requests cancellation, prompt window appears, reconfirming cancellation
2) Process(or I/O)
   a) Clicking Return or X window button
   b) Reconfirming cancellation, Yes or No

*AE. Prompt - Proceed*

1) Description
   - Before proceeding compiling, prompt window appears, reconfirming proceeding it
2) Process(or I/O)
   a) Clicking code
   b) Reconfirming proceeding, Yes, or No

**Process 3. Manual Testing and Result Saving**

*AF. Result - UI*

1) Description
   - Showing result and providing process check and copy function to user (see Figure 6.
2) Process(or I/O)
   a) 3 button Input, Output, Copy Code

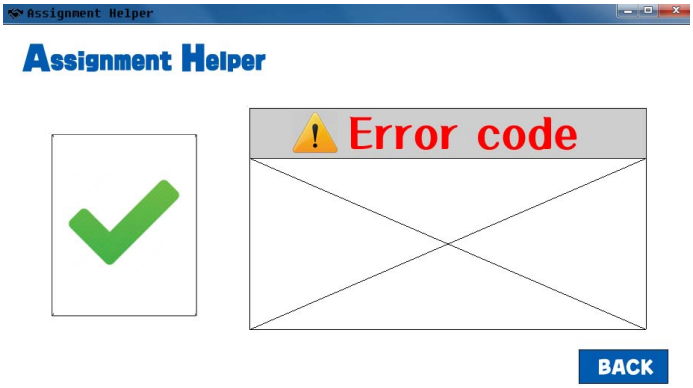



Fig. 6. Concept Image of Code Validation

*AG. Process Checking - Input & Output*

1) Description
   - User can check whether code is what they wanted or not through manual checking process
   - Once user pushes input value into the program, they can see there is expected output or not.
2) Process(or I/O)
   a) Input: Input value into program formed with selected code
   b) Output: Print output from pushed input

*AH. Copy Code*

1) Description
   - Providing copy the whole code which is selected, easily
2) Process(or I/O)
   a) Clicking copy button
   b) Copy selected code on the clipboard

*AI. Cancellation(4) - Clicking Return Button*

1) Description
   - User can go back with clicking return button
2) Process(or I/O)
   a) Clicking return button
   b) Go back to the code selection window

*AJ. Cancellation(4) - Cliking X Window Button*

1) Description
   - User can exit program

2) Process(or I/O)
   a) Clicking X window button
   b) Program closing

## V. Architecture Design and Implementation

### A. Overall architecture

see Figure 11

### B. Directory Organization

TABLE II
DIRECTORY ORGANIZATION

| Directory | File Names | Module Names |
|---|---|---|
| project/src/keyword/ | synonym_test.csv<br>keyword_search.py<br>topic_analysis.py | keyword_search |
| project/src/crawling/ | crawling_common.py<br>crawling_stack.py<br>crawling_google.py | get_code |
| project/src/comp_exec/ | error_argument_C++.py<br>execution_C++.py<br>error_argument_py.py<br>execution_py.py<br>... | validation |
| project/src/GUI/ | search.py<br>candidates.py<br>compiling.py<br>error.py<br>success.py | GUI |

**Module Description**

### C. keyword - keyword_search

This module is to take real language from user in sentence or words, split up, and replace them to keywords for search. Our program is to detect keywords from natural language, and make coded files for output. For that, a csv file for keywords is needed. If this list of keywords is too short, all it can make is an error sign. So, we first made 2000 words for key, and make it extend. What we need for search a word in some sentences is first, split function. By that, we can save our resources for replacement, reduce error cases, or decrease misunderstanding. Then it replaces natural language to limited number of keyword. Those keywords are sent to next stage. And it analyzes frequency, deviation, and tendency of keyword composition for later use. The synonym_word.csv is for those keywords, but there will be needs of complement. So, with analysis of keywords, the csv file extends.

see Figure 7

### D. get_code - Crawling and Selecting

This module search for appropriate codes using keywords from the *query* module. *get_code* imports *requests* for query requesting to the server, mostly using REST API supported from forums. For raw html files, the module imports *beautifulsoup4* package to parse the code. After crawling codes from sites, this module also evaluates the quality of the code.

see Figure 8

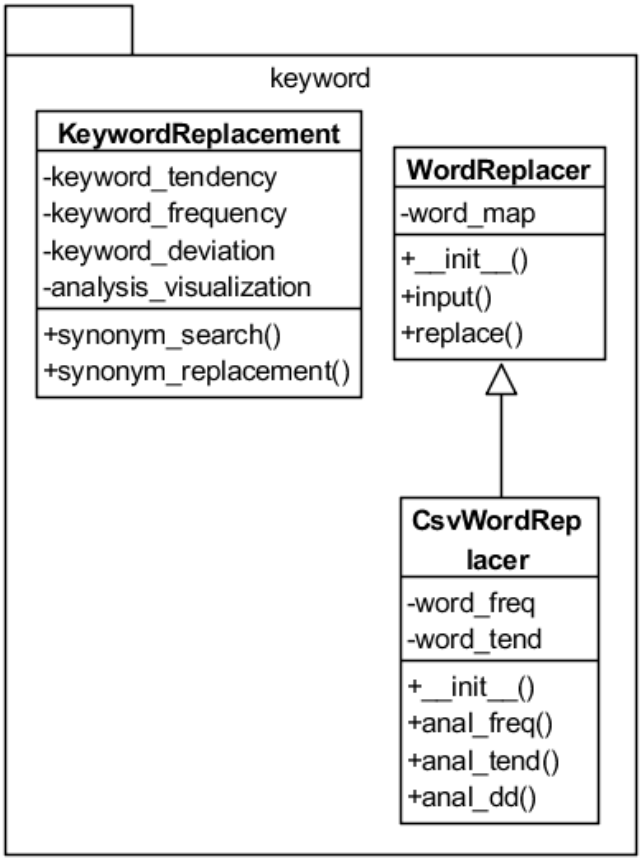



Fig. 7. Description of module *keyword_search*

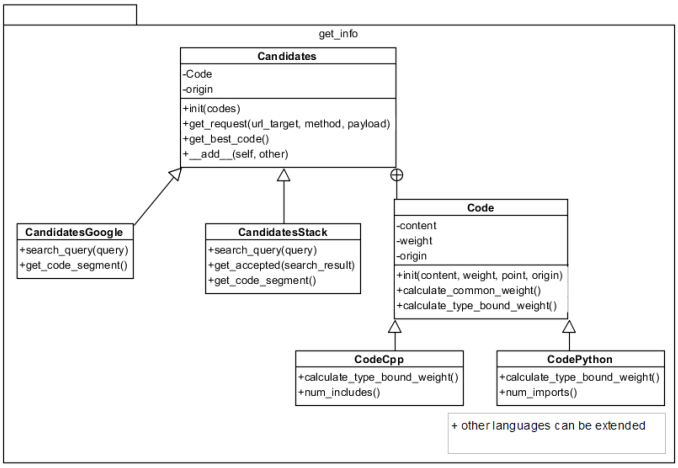



Fig. 8. Description of module *get_info*

### E. validation - Compling and Execution

This module is to take cpp file generated after parsing codes, then compile, and execute it. It imports *subprocess*, which makes python freely handle other programming language. It creates command with Popen function in subprocess, and then executes it with communicate function in subprocess.

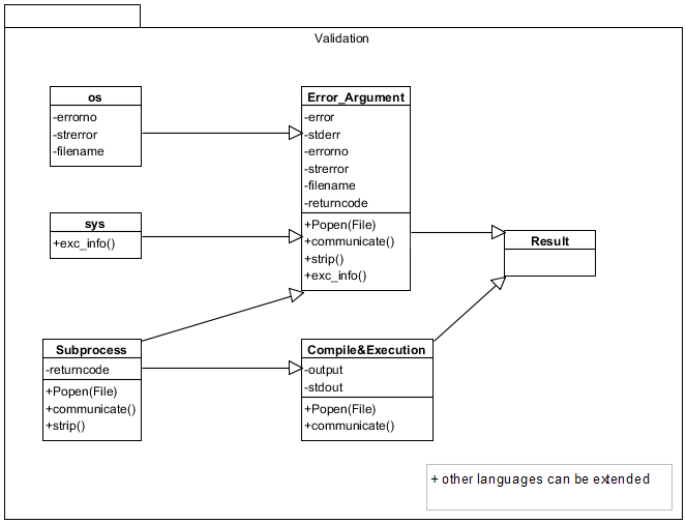



Fig. 9. Description of module *validation*

It compiles test.cpp using g++ and executes it using "./test" command Error handling is made by error exception method. At this point, it imports os, sys, and subprocess. Each of these catches error which they can handle. The os uses errorno, strerror, and filename. The sys uses exc_info() function, and the subprocess uses returncode and strip() function. Then Error_Argument prints out error caused within os, sys, and process.

see Figure 9

### F. GUI

GUI consists of 5 windows corresponds to each procedure. Firstly, the *main* window provides search. After clicking the search, the program executes *keyword_search* and then window moves to *search_result* window. *search_result* window has candidates from *get_info* module.

Clicking *compile* button on the *search_result* window will execute compile. Compile is done by *validation* module. After compiling, the program shows either *success* or *fail* window

see Figure 10

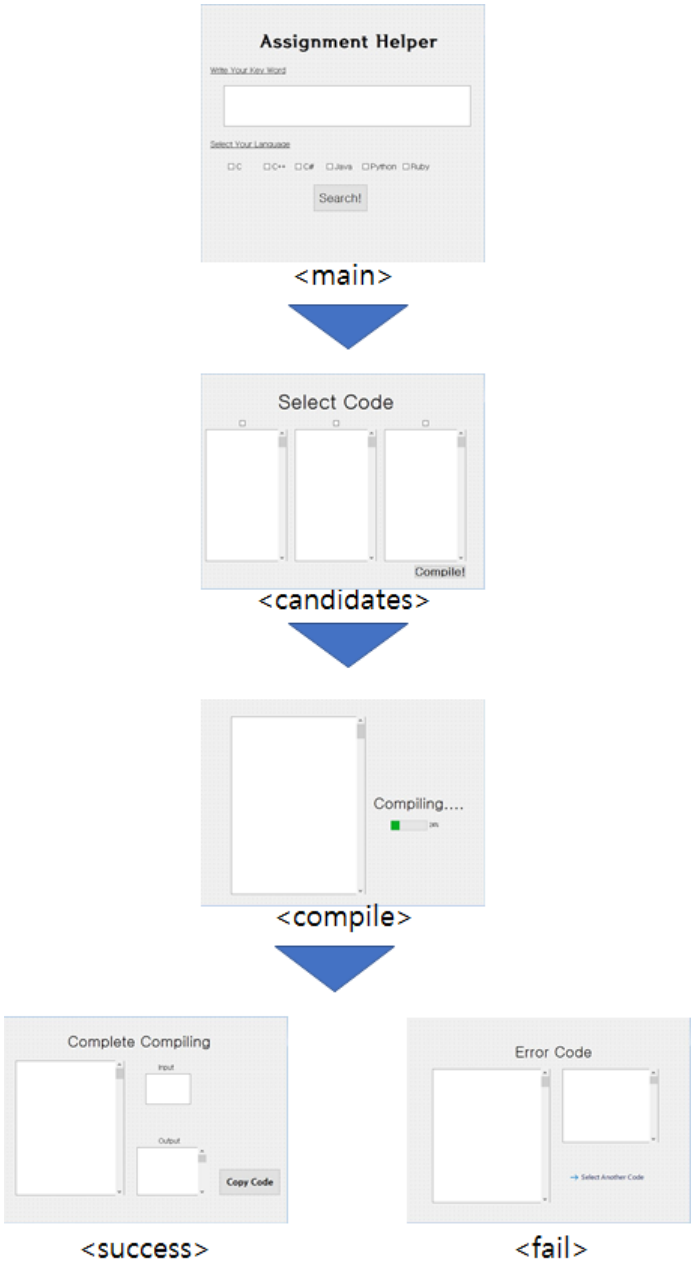



Fig. 10. Description of module *gui*

## VI. Use Case

### A. install

This use case specifies the way user install the program

- download the file from the link
- auto extractor will unzip the program
- show pragram installed finished display

### B. first run the program

This use case presents how the program will act when the program is run very first time

- initial running the program
- show up the configuration window(conf.exe)
- after configuration done, do the tutorial

### C. do the tutorial

This use case specifies how the program will act when user do the tutorial

- clicking the tutorial button on main window or initial setting
- play the tutorial movie(tut.mov)

- after playing the tutorial, close the tut.mov

*D. open configuration window*

This use case specifies how the configuration window looks like when the user opens the configuration window

- initial running the program or double clicking on conf.exe
- open a checkbox gui
- the checkbox gui has languages to be installed

*E. check the languages and confirm installation*

This use case specifies how the program should act in the configuration window. In the configuration window, the user can easily install language packs from server.

- check language packs to install / uncheck languages to uninstall
- click install button to install and uninstall selected ones

*F. open readme*

This use case describes about opening the readme file.

- open a readme file which gives basic installation

*G. run the program(main.exe)*

This use case specifies of initializing the program when the user executes the program.

- running the program
- open the main window(search window)

*H. tooltips*

This use case specifies about how the user enables the tooltips

- hanging mouse on the buttons for 3 seconds
- show the tooltips
- tooltip disappear 1 seconds after mouse hovering finished

*I. view search history*

This use case specifies about viewing search history. User can view the search history and replicates the search result using search history.

- clicking search history button
- show search history pane

*J. enter a query*

This use case sepecfies how users can search their questions. The users first select the language and then search on window

- enter a query on the search window
- while entering a query, keep run auto completion
- show the auto completion result
- select language to search
- uninstalled languages are greyed out

*K. submit the query*

This use case specifies the procedures when submitting the query

- click 'search' button
- run, query extracting module
- run, *get_code* module based on the result
- show the candidates window

*L. select code*

This use case specifies about how user can interact with *candidates* window. This is the main interaction with the program.

- view each code to compile
- the user evaluates codes
- click code that the user things the best suitable.

*M. modify code*

This use case specifies about modifiying the code if the user wants to.

- the user decide whether the codes needs to be refined
- click on the code
- editing the code
- press enter to confirm

*N. compile code*

This use case specifies about compiling the code, when user clicks copile button.

- after select and modify, user press compile button
- validation module executes the compile process
- show the result of compiling either *success* or *fail*

*O. pop up fail window*

This use case specifies when the compile fails, shows error result.

- failure of the compiling triggers the fail window
- show the fail window
- fail window shows errors

*P. pop up success window*

This use case specifies when the compile succeed, shows copy code window.

- pops up the window
- the success window provides to test input and output
- the success window also provides copying code

*Q. test input*

This use case specifies about testing more input using *validation* module.

- put input on the input textbox
- click on the test button
- outcome will show the result with the program

*R. copy code*

This use case specifies when the user wants to save result, this provides the way to save the result.

- press copy code button
- code will be copied to the clipboard

*S. click x button*

This use case specifies how the program act when x buttons are clicked

- click x button
- pops prompt window that asks about the cancellation
- cancel all procedure and go back to the first search window

*T. program exit*

This use case specifies how to close the program.

- click x button on main(search) window
- program terminates

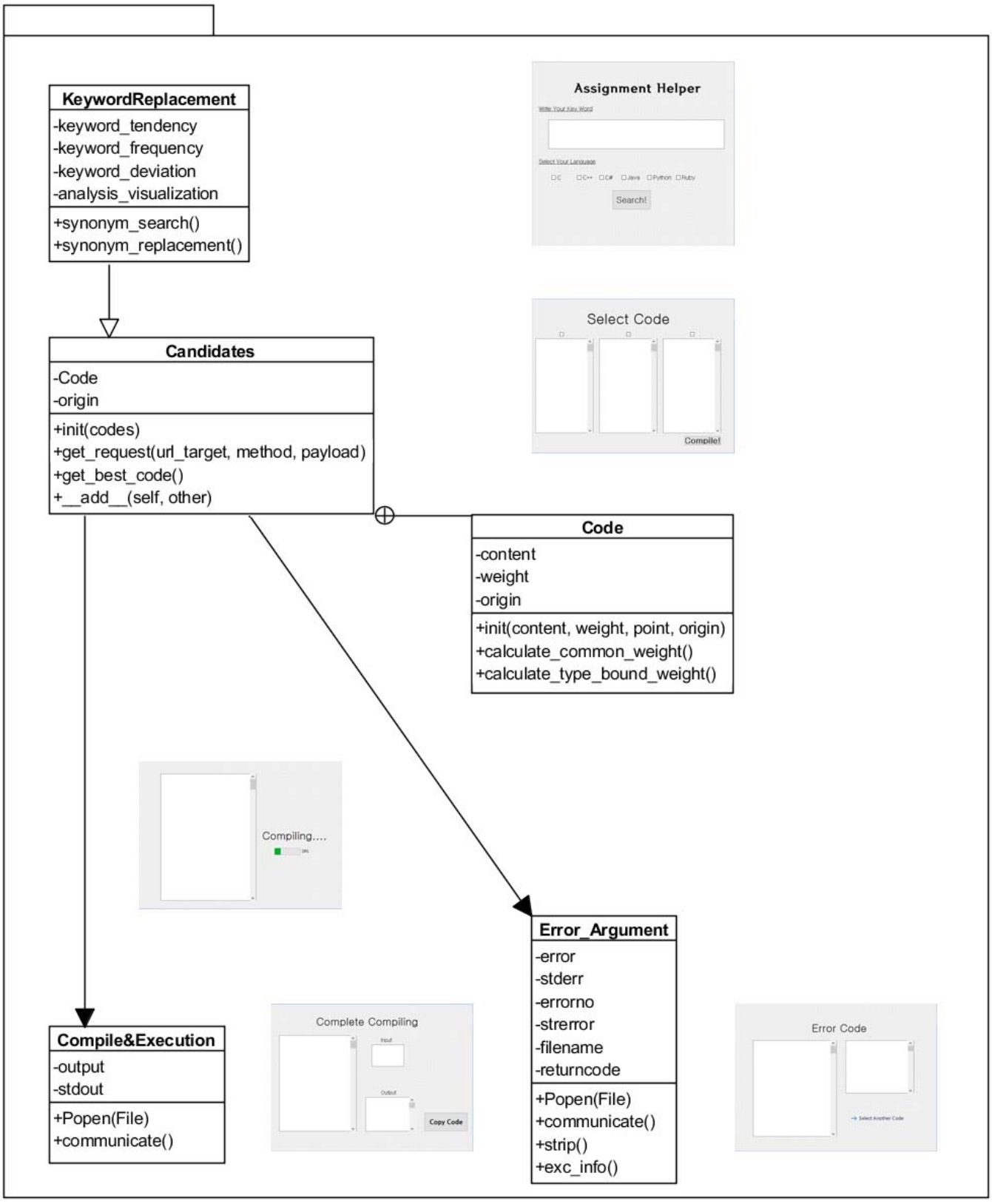



Fig. 11. Description of Overall Architecture