Team *Enterprise*

“学以思”(STT) 成绩考评系统

Student Course Grade Assessment System

Software Feasibility Analysis Report

Report Status: [ ] Draft [*] Published [ ] Under Revision	Report ID:	Enterprise_STT_Feasibility
	Version No.:	1.0.0
	Author:	1809853G-I011-0014_张修博 1809853G-I011-0068_宋康琪 1809853J-I011-0065_贺宇杰 1809853E-I011-0016_王首鉴
	Date:	October 26, 2021

History of Versions

Version/Status	Author	Participant	Date Period	Comment
0.9.0	All team members	Not rated	9/9-9/13	first draft
1.0.0	贺宇杰 张修博 宋康琪 王首鉴	40% 20% 20% 20%	9/15-9/20	**Published**

CONTENTS

1 Introduction

1.1 Product Positioning

This product is a comprehensive software that provides a series of convenient services around the teacher group to evaluate students' class performance.

1.2 Product Functions and Features

1. Use the database to store student information and student course score information.
2. Can provide functions like MS Excel's ranking, statistics, and addition functions. Able to assist teachers to make analysis with diversiform charts. Can export these results in a variety of file formats, and have printing support.
3. Friendly graphical solution and interactive system.

1.3 Customers and Market Prospects

Our main customer group is teachers.

We carried out a brief questionnaire about what people would expect for such a system. Since a Course Estimation System's users are mostly teachers, the questionnaire is mostly sent to teachers.

We got 23 groups of data, which is a small amount, but we consider it to be enough.

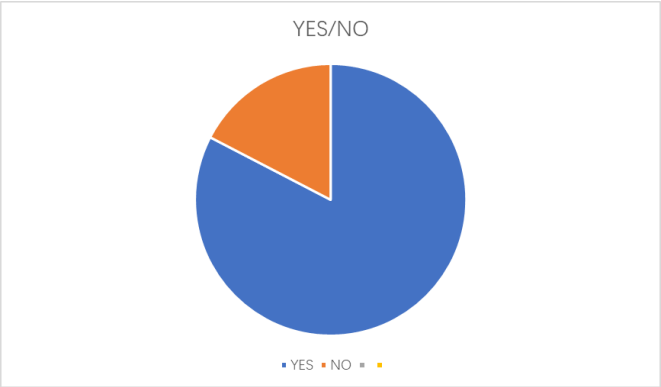

Figure 1 Result of 'Are you using such a system to grade student's study situation for the course that you teach?'

1.3.1 Demand for the System

The first question on the questionnaire is 'Are you using such a system to grade student's study situation for the course that you teach?'

19 people chose YES while 4 people chose NO. The pie chart is shown in Figure 1.

It's easy to see that more than 80% of them using such a kind of system.

Therefore, a powerful course assessment system is necessary for most of the teachers.

Figure 2 Result of needed functions

1.3.2 Demand for Some Functions

In the other questions that's raised, we tried to find out what functions we need to think about if we want to create a great assessment system for teachers.

We let them choose the level to answer how much they need for each of the functions we raised between 0 and 10. 10 means it's extremely important while 0 means it's totally useless.

The result is shown in Figure 2. All the functions we think about reach level 8, which is great because we do know what the users might need. Among all the functions, users' choices show that they really think it's important to have the system connected to the student database so that some mistakes could be prevented while using the system.

Also, we raised a question about how much the users may need a phone app to use this system. Compared to the data above, the level is lower for this question, which is only 7. A teacher also e-mailed us to discussed about this issue. He said making assessment for students is a serious work and using phone app is maybe unsafe, on which we all agree.

So, the development of a phone app will not be the majority of our work. We will focus on bringing out the functions that we think about on PC.

2 Feasibility Analysis

2.1 Technical Feasibility

2.1.1 Hardware Platform

Our software should run on major desktop platforms and may be further extended to mobile platforms, as it currently does not have dependency on special hardware other than a generic computing device. However, technology for developing for mobile platforms is different from that for desktop, so mobile platforms will not be our current focus. Our members are all equipped with at least one x86_64 (Intel/AMD) computer, which is eligible for developing this software, and is also the main target platform we develop the software for. Moreover, we have additional standby Mac laptops, so that after the main development process, we can do specific test to ensure our software also works on Mac.

2.1.2 Software Platform

2.1.2.1 Operating System

Major desktop OSes, running on the hardware platforms described above, include Microsoft Windows, macOS and variations of GNU/Linux. All of them have enough capabilities to support correct and efficient functioning of our planned software. But due to lack of some components for GUI in some operating system installations, functionalities will be limited in certain cases. See Section *2.1.3.2*.

2.1.2.2 Programming Language & Runtime Environment

Concerning the possibility for our software to run on different OSes, it is desirable to have a programming platform that is portable to some extent. Also, it should have a reasonable number of features to assist us in modeling the problems and testing the software. An additional consideration is that our members all have some backgrounds in writing C programs, so it would be good if the main programming language is C-like.

Some candidate programming languages, along with their corresponding runtime environments (marked in square brackets), are listed below:

- *C++ (with standard C++11 or above) [Native, with C/C++ standard libraries]*
  It is preferred due to its similarities to C, and its efficient and compact runtime.
- *Java (SE 8 or above) [JVM]*
  It has good type systems for problem modeling, and a portable runtime, but has more differences from C than C++.
- *C# [.NET Framework/.NET Core/Mono]*
  It has good type systems and concepts, but portability may be a problem.
- *Python (3.x) [Python interpreter, such as CPython and PyPy]*
  It is a language with dynamic typing, which is far away from C; runtime efficiency for desktop applications is a drawback.

2.1.3 Investigation for Specific Area

2.1.3.1 Data Storage

It is a common requirement that information given by the user will be persisted through different sessions. This demands a database or similar system. Additionally, the system is preferred to be lightweight, as this will reduce the time to install our software for the user. It should be reliable, but reliability requirements for critical systems (such as a financial system) do not apply here. Moreover, it should have an easy interoperation with the chosen programming platform.

Some candidates are listed below:

- *SQLite*
  It is a free, easy-to-use, and lightweight DB with an official C/C++ API.
- *MySQL*
  It is a relatively large and open-source DB, popular in enterprise applications. A lot of materials can be found on the Internet.

2.1.3.2 Graphical User Interface

GUI is employed when our software needs to display some visual data to the user, or the user just needs an easier-to-use interface to interact with the software. This capability is available natively on Windows (Win32 API) and macOS (Cocoa API), and through X11/Wayland display systems on GNU/Linux[1].

Writing three different interfaces for the three families of OSes is not feasible, so we also need libraries/tools that can help us reuse most code for different GUI systems. The selected library/tool should have good interoperation with selected programming platform.

Some candidates are listed below:

- *Qt*
- *GTK*
- *JavaFX (generally only works with Java)*

[1] This means GUI functionality cannot work if the user does not correctly configure X11/Wayland systems on their machine. Although GNU/Linux in text mode can also simulate some GUI features, it is hardly feasible due the development time limit.

2.1.3.3 Web

Our software may also involve features on Web, that enable user experiences of remote accessing. Since these features should not change the basic behaviors of the software core, they could be designed as a wrapper of the core, which requires the Web development platform has good interoperation with the core platform. Depending on the chosen main programming platform, some candidate platforms for the "Web server" part are listed below:

- *Common Gateway Interface (CGI)*
  This works well with C++.
- *Python (with Django)*
  Python has a native interoperation mechanism with C. And some team members have experiences working with it.
- *Java (with Spring Framework)*
  It is good if we finally choose Java as the main programming language. Otherwise JNI may be required to do interoperation.
- *ASP.NET*
  It is good if we finally choose C# as the main programming language.

For the frontend part, current major browsers, including Chromium, Google Chrome, Microsoft Edge, Mozilla Firefox, and Opera, all support modern HTML/CSS/JavaScript standards well. There are also some third-party frameworks, like *Vue* and *React*, to accelerate our development and ensure the quality.

But there will be great difficulty in development if our user is still use older browsers like Microsoft Internet Explorer (especially versions 6, 7 and 8), despite possibility in theory, because of their huge difference from modern Web browsers. As a result, it is very likely that we must give up supporting such browsers, due to limit in time and labor.

2.1.3.4 Development Tools

In addition to the fundamental programming platform, some tools are available to boost the efficiency of the development process.

If C++ is used, then we need tools to maintain the building process across different operating systems. *CMake* is a tool catering to this need, which can produce platform-specific build configuration files (such as *Makefile*, *Ninja* and *MSBuild*) from a general and cross-platform *CMakeLists.txt*. There is also *vcpkg*, an open-source tool that can help manage C++ third-party library packages. Programming languages newer than C++ often come with more complete solutions for these kinds of management.

2.2 Operational Feasibility

2.2.1 Learning Cost

Learning cost is low, but some challenges still exist.

It is very probable that basic functions are implemented in C++ or C language. Members of our team can independently write C++ or C language code.

For the GUI part, it is required to mobilize certain GUI libraries like Qt (with a C++ interface), which may increase learning cost to some extent.

Also, difficulties may exist in integrating with the database, and designing the Web frontend.

All above will bring some challenge, but they are affordable.

2.2.2 Time Feasibility

The basic functions are not difficult to achieve, and can be adjusted at any time according to needs. It is completely feasible to complete the basic functions before the mid-term exam. Each part can be carried out in a very rhythmic and orderly manner.

And we have half of semester to expand our functions like completing GUI of this project. We can spend enough time on testing.

We are confident to do every part completely within the stipulated time.

2.3 Economic Feasibility

2.3.1 Labor Cost

The software development is mainly carried out by the team members on the PC, and there is no labor cost.

2.3.2 Development Costs

The software mainly uses the free or educational version (will not be used for commercial use).

2.3.3 Market Research and Publicity Costs

A small number of samples are surveyed in the form of questionnaire surveys, and the cost is very low. Currently, there is no commercial release plan, so no marketing or advertising activities are carried out.

2.4 Legal Feasibility

2.4.1 Licenses of STT and Modules of it.

2.4.1.1 Licenses of STT

This software was originally used for classroom learning and teaching purposes only, but the commercial rights are reserved.

Copyright(c) Team Enterprise.

Even if the software is commercially available, we are happy to adopt LGPL for some codes or functions.

If you use the form of dynamic link, then you can publish your application in any form, commercial, non-commercial, open source, non-open source, whatever you want. If for some reason you must statically link a library released under the LGPL license (hereafter referred to as LGPL library), then you are obligated to do the following:

1. You must state in your document that the LGPL library is used in your program, and that this library is released based on the LGPL;

2. You must include a copy of the LGPL agreement in your application release, which is usually the text file;

3. You must open all codes that use LGPL library codes, such as some wrappers. However, other codes that use these wrappers do not need to be open;

4. You must include the object files for the rest of your application (usually what we call .o etc.), or other equivalent files. The source code is not required.

2.4.1.2 Copyright of SQLite

SQLite is in public domain.

All of the code and documentation in SQLite has been dedicated to the public domain by the authors. All code authors, and representatives of the companies they work for, have signed affidavits dedicating their contributions to the public domain and originals of those signed affidavits are stored in a firesafe at the main offices of Hwaci. All contributors are citizens of countries that allow creative works to be dedicated into the public domain. Anyone is free to copy, modify, publish, use, compile, sell, or distribute the original SQLite code, either in source code form or as a compiled binary, for any purpose, commercial or non-commercial, and by any means.

The previous paragraph applies to the deliverable code and documentation in SQLite - those parts of the SQLite library that you actually bundle and ship with a larger application. Some scripts used as part of the build process (for example the "configure" scripts generated by autoconf) might fall under other open-source licenses. Nothing from these build scripts ever reaches the final deliverable SQLite library, however, and so the licenses associated with those scripts should not be a factor in assessing your rights to copy and use the SQLite library.

All of the deliverable code in SQLite has been written from scratch. No code has been taken from other projects or from the open internet. Every line of code can be traced back to its original author, and all of those authors have public domain dedications on file. So the SQLite code base is clean and is uncontaminated with licensed code from other projects.

2.4.1.3 Copyright of Qt

Qt is available under different licensing options designed to accommodate the needs of our various users:

Qt licensed under commercial licenses is appropriate for development of proprietary/commercial software where you do not want to share any source code with third parties or otherwise cannot comply with the terms of the GNU LGPL version 3.

Qt licensed under the GNU Lesser General Public License (LGPL) version 3 is appropriate for the development of Qt applications provided you can comply with the terms and conditions of the GNU LGPL version 3 (or GNU GPL version 3).

Qt components licensed under the Qt Marketplace License Agreement are appropriate for the development of Qt applications commonly with Qt software components licensed under the commercial or GNU LGPL version 3 (or GNU GPL version 3) terms and conditions.

Some Qt libraries are related to third parties and we will try to avoid using them. If used, it will be noted or handled with special care**.**

2.4.2 Censorship and Related Laws

Since this product is likely to be used in the People's Republic of China, if the product will be commercialized. We must establish a complete text censorship system.

If network services are provided to the public, they must be filed with relevant departments in accordance with relevant laws and strictly abide by relevant laws.