Feasibility Study

Project Title: Scholarly

scholarly

Fueling Dreams, Shaping Futures

Client/Organization:
MSU Computer Science Department

Team Name:
S.E.E.S

Authors:
Angel Badillo and Khethulwazi Kunene

Date: March 18, 2024

Table of Contents

1. Introduction

1.1 Overview of the Project

The Midwestern State University (MSU) Department of Computer Science has a scholarship committee that is looking to make the process of dealing with scholarship applications more efficient, automated, and less time consuming. Currently, the Chair of the scholarship committee, Dr. Lopa has to do everything by hand, which is very time consuming and involves a lot of repetitive tasks. She looks through all the students in the Computer Science program, determines whether an applicant is eligible for a certain scholarship or not, for every student. She also has to write scholarship acceptance letters for each recipient of a scholarship, one by one as well. Furthermore, the process of selecting candidates for each category of the Outstanding Student Awards, then creating a poll for them to share with the other committee members is also very tedious. This is where Scholarly comes in, a Windows 11 GUI application that aims to streamline and automate these processes for the Chair of the Scholarship Committee.

1.2 Objectives of the Project

The objectives of Scholarly are to automate as much of the processes of handling the scholarship application review and acceptance processes, and the processes selecting and polling of candidates for the Outstanding Student Awards. Thus, Scholarly will aim to sort and filter student records based on the criteria for a scholarship, generate scholarship acceptance letters, send emails to recipients, and create a poll for the “best” candidates for the Outstanding Student Awards.

1.3 The Need for the Project

The current system for handling these processes involving scholarships and the Outstanding Student Awards is time consuming, and involves a lot of repetitive, manual human input and processing. The Chair and the scholarship committee are seeking a software system to automate and streamline the processes to minimize the need for human input and repetitive manual labor. As it stands currently, these processes take a substantial amount of time, and having a software system that can speed up these processes would greatly reduce the amount of time it takes to complete these processes.

1.4 Overview of Existing Systems and Technologies

There are numerous existing systems that offer similar functionality or are tailored to similar use cases that Scholarly is aimed to provide. For example, SmarterSelect and SurveyMonkey Apply. Both are application management systems, which can be used for scholarships or other financial awards. However, they differ in the sense that they focus on the process of applicants submitting applications, but they require a lot of configuration, they charge high annual and monthly fees, and lock features behind different plans. Furthermore, SmarterSelect and SurveyMonkey Apply require that the reviewer / evaluator review and approve each application one by one.

1.5 Scope of the Project

Main actors of this system

- The Chair of MSU Computer Science Scholarship Committee

Main use cases associated:

1. The Chair of MSU Computer Science Scholarship Committee can search and filter through student data to see what students are eligible for a scholarship.
2. The Chair of MSU Computer Science Scholarship Committee can generate scholarship acceptance letters for recipients of a scholarship.
3. Chair of MSU Computer Science Scholarship Committee can send generated emails to students meeting the requirements for a scholarship.
4. The Chair of MSU Computer Science Scholarship Committee can create a Google Form or some type of poll with students meeting the requirements for the Outstanding Student Awards.

1.6 Deliverables

The deliverables for this project will be:

- The Scholarly application designed for devices running Windows 11 or above.
- The user manual for Scholarly.
- The documentation for Scholarly.

2. Feasibility Study

2.1. Financial Feasibility

The development of the Scholarly application is very feasible, from the financial aspect. All software used to develop the Scholarly have licenses that make them free for personal and commercial use, as well as open-source. The softwares to be used are:

- Python 3.12.2
- SQLite 3
- PyQt 6
- Gmail API
- Google Forms API

All of these software are free to use, and allow for commercial use. Therefore, there is no extra development cost due to the softwares being utilized to develop the Scholarly application, making the project very feasible from the financial side.

2.2 Technical Feasibility

From the technical point of view, this project is feasible because the existing technologies in current times allow for the application to be built and implemented. As previously mentioned, the softwares used to implement this application are the Python programming language, the SQLite 3 library, the PyQt 6 library, and the Gmail and Google Forms APIs. The PyQt 6 library is a Python library that has bindings to Qt version 6, a C++ library for creating graphical user interface (GUI) applications. SQLite 3 is a library written in the C-language that implements a fast, self-contained, serverless database engine. SQLite 3 has a Python library that interfaces the SQLite 3 database engine, and allows for its use in Python code. The student data imported from CSV files needs to be stored in a database during the application's execution for fast lookups, and SQLite 3 fulfills this. The application needs the user to send emails to the scholarship recipients, and since the email addresses provided by MSU to the students and faculty are Google Gmail addresses, the Gmail API can be used. As for creating a poll with Google Forms, this can be done with the Google Forms API, and this requires a Google account, or a Gmail address, which all staff at MSU have. Thus, taking this all into consideration, this project is feasible from the technical standpoint.

2.3 Resource and Time Feasibility

There are enough resources for the development of the application. Money-wise, there are no additional costs, or recurring payments that need to be made to any service or software that will be utilized for the development of Scholarly. Labor-wise, there are 2 software engineers in S.E.E.S that are working on the development of the application. While a 2 member team is rather small, it is enough for the project at hand, with the timeline and deadlines for the project. The usage of the Python programming language typically enables programmers to write programs at a much faster pace, as it is a language with automatic garbage collection and it is highly dynamic, meaning that the programmer does not have to manually manage and keep track of allocated memory.

2.4 Risk Feasibility

a) **Risk associated with size**
Since the application is to be written in the Python programming language, there is a chance the size of the application may take up relatively more space in storage than if it were to be written in another programming language such as C or C++. However, if good programming practices are followed, and there is minimal or no duplicate code, and the application is packaged without extra unused libraries, then the size of the application should be acceptable for the minimal requirements of a device running the Windows 11 operating system.

b) **Users of the product**
As it was discussed and revealed during user interviews and user stories, there is only one class of user. The main user for this product is the Chair of the MSU Department of Computer Science Scholarship Committee.

c) **Number of projected changes to the requirements for the product? Before the delivery? After the delivery?**
Before the delivery of the product, it is expected that there will be some changes to the requirements, as the client and users may provide feedback that dictates some changes to be made to the requirements. It is expected that possibly 2-5 requirements be changed before the delivery, as it is possible security requirements, performance requirements, and user interface requirements are subject to change as the client demos the application during testing phases. After the delivery, it is expected that requirements may be added, such as maintainability requirements such as specifying the Mean Time To Repair (MTTR), or the minimum level of services (LoS).

d) **Amount of Reused Software**
While the code will be written from “scratch”, per se, there will be reused software in the form of Python libraries and packages, as well as the use of Google APIs to interact with Google web services such as Gmail and Google Forms. The functionality for creating a shareable poll and sending emails will not be newly coded, the application will have integration with the Gmail and Google Forms API to fulfill these requirements.

e) **Business impact risks**
   i. **Effect of this product on company revenue**
   It is expected that this product will not directly generate profit itself, as the main user of the product will be the current client, the Chair of the Scholarship Committee. However, the product’s main purpose is to reduce the amount of time spent on processing scholarship applications and selecting candidates for the Outstanding Student Awards, which in turn, will provide the Chair of the Scholarship Committee, and the rest of the Committee more time to tend to other responsibilities corresponding to

their roles at Midwestern State University. Indirectly, the time saved by this product will generate more revenue, as it might reduce the required resources expended for other responsibilities the Committee have to tend to.

ii. **Reasonableness of delivery deadlines:**
Considering the fact the S.E.E.S team has prior experience under their belt with Python programming, and one of the team members, Angel Badillo has a solid background on mobile application, desktop application, and web application development, a lot of the implementation for the Scholarly application will be similar to previous experiences faced by S.E.E.S. Thus, the first deadline that involves providing a demo of the application, being March 18th, 2024, is reasonable as the time between the deadline for the demo and the time between the previous iteration was roughly a month in time. The final product has to be delivered by the beginning of May 2024, but by March 18th, 40% of the functionality should be delivered. Meaning, that slightly less than 2 months time, the final product must be delivered, which is doable if the development of the application keeps the same pace.

iii. **Number of customers who will use this product and the consistency of their needs relative to the product**
The number of customers who will use this product is relatively narrow, as the main user of this application will be the Chair of the MSU Department of Computer Science Scholarship Committee. The main user is also the client, and the requirements for the system correspond directly with the needs and wish list of the client. Therefore, the consistency of the customer's needs relative to the product will be one to one, as it is tailored to fulfill the customer's requirements.

iv. **Number of other products/systems with which this product must be interoperable**
The other products or systems that Scholarly will be interoperable with Gmail and Google Forms. The interoperability is one way; Scholarly interacts with the Gmail API and Google Forms API.

v. **Sophistication of end users**
There is only one class or type of user, which will be the Chair of the scholarship committee, Dr. Lopa, and other individuals that take the position in the future. Therefore, the end user will be an advanced level or expert level user, as they will have experience retrieving student records, and be familiar with the procedure of processing student scholarship applications and selecting candidates for the Outstanding Student Awards. The application will not differ greatly from the general procedures followed

by the existing system in place, but it will automate the process by minimizing the need of repetitive human interaction and labor.

vi. **Amount and quality of product documentation that must be produced and delivered to the customer:**
There will be a substantial amount of product documentation, and the quality of it will be high. The S.E.E.S team will follow the Google Python Style Guide, which involves clear, readable, and precise code documentation that enables easier maintenance and comprehension of the code. In addition to code documentation, the repository for the project will also have clear documentation in the form of a README file, explaining the configurations used in the code development environment used while developing the software. A user manual will also be delivered to the customer, which will be available online and will clearly show steps, provide details, and show pictures that will help the user learn how to use the application.

f) **Costs Associated with delivery:**
It is anticipated that the delivery of the software product will not incur any additional costs, as it is expected that the software application will be packaged in either an executable file packaged dependencies, or as an installer executable file. The client will be given access to the product online, and there will be no fee incurred as the delivery of the product will not require someone to physically deliver and install the product.

g) **Customers related risk:**
The risks related to the customers are rather sparse and few, as there is only one customer / client, and the client has worked closely with the S.E.E.S team to derive the requirements, constraints, and provide other necessary assistance regarding information regarding the requirements.

h) **Development Environment Risks:**

i. **Is a software project management tool available?**
Yes, there are numerous tools that will be used to manage the software project. For management of code changes and version control history, Git will be used. For keeping track of the timeline and deadlines of the project, there are softwares such as Trello Board, Monday.com, and Slack, which are used for team communication and project management.

ii. **Are tools for analysis and design available?**
The tools used for analysis and design will involve the usage of Visual Studio Code, and plug-ins that analyze the source code for errors and design issues. The types of plug-ins that do this are linters, code formatters, debuggers, test explorers, and code navigators. Such plugins are the Python, Pylance, and Black Formatter extensions.

iii. **Are compilers or code generators available and appropriate for the product to be built**
Yes, there are appropriate code generators and interpreters for the product to be built. Python comes packaged with its own interpreter when installed, so there is no need for external compilers or interpreters. Code generators that could be employed for the development of the GUI of the program, such as Qt Designer, a GUI design software that allows for drag-and-drop visual programming of the GUI.

iv. **Are testing tools available and appropriate for the product to be built?**
There are multiple testing tools out there for Python code, however Python comes packaged with libraries for performing tests on code already, such as the doctest library, which performs tests written within the documentation to ensure that functions and methods with the code work properly.

v. **Are software configuration management tools available?**
The Visual Studio Code code editor will be used to write the code for the application, and within Visual Studio Code, there are many configuration management tools for the workspace.

vi. **Does the environment make use of a database or repository?**
Yes, the development will make use of a repository for the purpose of version control and tracking changes. As for a database, there will be a database used, but it will be a part of the application, and it will be packaged with the application.

vii. **Are all the software tools integrated with one another?**
The software tools that will be used to develop the product are integrated with each other, as Visual Studio Code provides plug-ins for Python development, and Python has libraries and packages that will be used to develop the application and communicate with Google APIs.

i) **Process issue risks:**
As it stands now, there are no observable issues or risks associated with the processes to be performed by the system along with the system.

j) **Technical issue risk:**

I. **Are specific conventions for code documentation defined and used?**
Yes, as previously mentioned, the Google Python Style Guide will be used as a guideline for creating the documentation.

II. **Do you use a specific method for case design?**
Currently, for test case design, the doctest library will be used for writing test cases for each module, class, method, and function in the source

code. However, as the project development moves forward, other methods of case design may be employed.

III. **Are configuration management software tools used to control and track change activity throughout the software process?**
Yes, Git and Github are used to control and track changes through the development process. Visual Studio Code plugins such as version control and Git Lens can also be used to track changes in a tree or graph view as well.

k) **Technology risks**

I. **Is the technology to be built new?**
For the most part, no, the technology to be built is not new. A lot of the source code and functionality will utilize existing software, libraries, and APIs.

II. **Do the system requirements demand the creation of new algorithms, input, or output technologies?**
A lot of the input and output technologies already exist, the schema for the input and output will mostly come in the form of comma separated value (CSV) files, Word Document (docx) files, emails, Google Forms, and possibly JavaScript Object Notation (JSON) files. However, for the generation of scholarship acceptance letters, there is a need to design a new algorithm in such a way that there can be specific placeholder values in a template letter that can be replaced with values corresponding to student records and the sender of the letter.

2.5 Social/Legal Feasibility

The product will not necessarily violate any ethics of any kind, as it will not release student information in an unauthorized manner. From the legal standpoint, all software tools used to develop the product are free for personal and commercial use and are open source. Therefore, it is feasible for the product to be developed as it does not violate any human ethics or local, state, or federal laws.

3.Considerations

3.1 Performance:

The application must be able to read in and process a large amount of student records, as well as scholarship criteria. There are approximately 6,000 students enrolled at MSU, divided amongst 49 undergraduate and 27 graduate programs. Although there are not that many students in the computer science program, it is expected that the application should be able to handle at least 10,000 student records with a maximum wait time of 20 seconds or less. Similarly, the application should be able to sort and filter student records based on student criteria in a short amount of time so the user does not have to spend a significant amount of time waiting.

3.2 Security:

The student information is very sensitive and should not be viewed by unauthorized personnel. Therefore, Scholarly should not transmit student data in an unsecure fashion, or an unauthorized fashion. Any communication protocols that need to be used to transmit student data over the Internet must be secured protocols, such as HTTPS. Furthermore, student data must not be transmitted without the explicit knowledge and consent of the authorized user, the Chair.

3.3 Usability and ease of use:

The application needs to have menus and be intuitive to the point it takes less than 2 hours of training and reading the manual to understand how to operate it. All major menus and menu actions should have good names that properly describe their functionality. Any slightly ambiguous or complex GUI components must provide tooltips to the user when the user hovers over them with the mouse for ease of use. Furthermore, tied in with the performance, the user should not spend a significant amount of time waiting for output for standard amounts of student records.

3.4 Capacity and scalability:

The product is tailored specifically toward the MSU Department of Computer Science and the client's needs, thus scalability is not a major concern. The student enrollment for the Computer Science program remains relatively the same each semester, and the capacity of students on campus as a whole is limited by several factors, such as classrooms capacity, housing capacity, and more. Therefore, the application just needs to be able to process slightly more student records than the total enrollment on campus with a short processing time to enhance usability. The capacity of student data will not change much due to the aforementioned factors, thus the application just needs to be at a good scale for at minimum, the Computer Science Department, and at maximum, the university as a whole.

3.5 Availability:

The application will be designed specifically for desktop computers with the Windows 11 operating system, which should soon be installed on all computers on campus.

However, it is possible for the application to run on Windows 10, as long as the device meets the Windows 11 hardware requirements. Therefore, the application has the capability of being available across the entire university. At minimum, the application should just be available for the Computer Science Department, but in the future, it could be expanded for other departments or the entire university.

3.6 Maintainability

The source code of the product should be well documented, in a well formatted style so that S.E.E.S or other future software development teams in charge of maintaining the software can do so with minimal issues reading and navigating the source code. Furthermore, the application should be designed with Object-Oriented-Programming (OOP) in mind so that functionality can be encapsulated in classes, with their corresponding attributes and methods to minimize coupling of portions of the code and enhance maintainability.

4. References

- https://www.slideshare.net/PasinduTennage/sample-software-engineering-feasibility-study-report
- https://pypi.org/project/PyQt6/
- https://sqlite.org/
- https://docs.python.org/3.12/faq/general.html
- https://developers.google.com/gmail/api/reference/quota
- https://developers.google.com/forms/api/limits
- https://google.github.io/styleguide/pyguide.html
- https://doc.qt.io/qtforpython-6/overviews/qtdesigner-manual.html