Elijah Anderson

Senior Project Proposal

*Avideom: A Desktop Media Player for Windows*

Section: Maria Schwartzman

February 14, 2019

Table of Contents

[Project Summary: 3](#_Toc461472358)

[Significance: 3](#_Toc461472359)

[Required Tools & Availability: 4](#_Toc461472360)

[Demonstration Plans: 4](#_Toc461472361)

[Qualifications: 5](#_Toc461472362)

[Project Specifications: 5](#_Toc461472363)

[Functional Specification: 5](#_Toc461472364)

[User Interface Specification: 6](#_Toc461472365)

[Technical Details: 6](#_Toc461472366)

[Timeline: 7](#_Toc461472367)

[Future Enhancements: 8](#_Toc461472368)

[Bibliography: 8](#_Toc461472369)

# Project Summary

This project consists of the implementation of an open-source desktop media player application. Avideom is able to play both audio and video media files (such as .mp3, .wav, .mp4, .mov, etc.) directly from the user’s files. It is designed as a better alternative to the Windows Media Player, the default media player application on Windows operating systems.

Once a user runs Avideom, he or she can select any audio or video file with a recognized file extension and it will play the corresponding media. Avideom features a clean and intuitive GUI and all standard media player functions (pause, play, skip, volume control, etc.), and potentially bonus features such as visualizations, an EQ tuner, or different playback speeds.

Avideom uses the Python programming language as well as several its libraries to accomplish the above tasks. It can be downloaded for free online.

# Significance

*Relate the project to an area or areas of Computer Science. This section is generally short - a few paragraphs - and is written in a way that will give the reader a general understanding of the scope, significance, and complexity of the project.*

Example: SmartDialer will solve the problem of forgetting or not knowing the area code for a phone number by allowing users to lookup and create a list of area codes. When moving to a new state or location, users may prefer to keep their old phone number. The issue arises that they can no longer dial seven-digit phone numbers to make local calls; they have to include the three-digit area code for their new location. It is often very easy to forget these three numbers, especially when people are new to the area. People who commute daily for work or have vacation homes also likely face the issue of remembering multiple area codes. The complexity of this application involves creating a local database, executing web requests, retrieving user locations, and designing an easy-to-understand graphical user interface. Most of the scope of this project does not relate to any of the material taught in DePauw’s computer science department, but rather information obtained through online courses and an internship. Parsing the HTML source code from the web requests requires an understanding of HTML tags and structuring which was taught in CSC 340: Web Programming and Cybersecurity.

# Required Tools & Availability

*This section should describe the software and hardware you will need in order to complete the project, and also explain how you will access that equipment. This includes the language or languages you will use to implement your project. Don't just say "C++" - instead explain whether you are using Borland C++ or Visual C++ on a PC, gnu C++ on a Linux machine, etc. If the equipment you need is publicly available at DePauw, say so - if not explain how you will access it. If the platform is not publicly available at DePauw, you must****clearly and unquestionably****have access to the required platform.*

This project will require access to a Windows computer, PyCharm, Github, and several Python libraries acquired via Internet. I will be using my personal laptop, a Dell Inspiron 15-7000, which runs Windows 10 and has all the necessary applications for project development. My machine has Python 3.6.1, the same version that my project is written in. I will use Github for version control as well as a place to back up all my code. To write the GUI, I will use tkinter , a simple and powerful Python library for creating GUIs. This project will also use piglet and winsound to assist with playing media files. Other Python libraries used in this project for varying tasks include datetime, threading, os, and sys.

# Demonstration Plans

*You will be required to demonstrate your project "live" in room 260 of the Julian building during our checkpoint meetings. Explain how you will do this - is the hardware/software you need available in that room/your laptop? If not, what equipment will you use and how will you provide it?*

All Avideom demonstrations will be done in Room 260. The day prior to any demonstrations, I will practice in Room 260. I will use Google Hangouts for screensharing so that adapters are unnecessary. If any technical difficulties occur, I will have backup adapters to use. Eventually, a demonstration may be done through a classmate’s device as well.

# Qualifications

*The proposal should clearly describe your qualifications for completing all work described in the proposal. In other words, you must demonstrate that you have the knowledge and ability to complete the project. This might include stating the relevant course, internship, or experiences you have had that will allow you to complete the project. You should clearly explain where you learned the languages and tools you will be using. For example, if you are using a specific graphics library through C++ you should discuss where you learned how to use that library. If you are using Python, you should explain where you learned Python, etc.*

The project is written in Python, a programming language that I have three years of experience in. I have taken two classes at DePauw dealing with Python (CSC 296 and CSC 396) as well as real life production experience in Python for two months during an internship at Cerner, the current leading company in healthcare software. Additionally, I’m well versed in object-oriented programming, the method by which Avideom is developed. I also have experience in computer graphics from taking CSC 350, which helps me understand how file formats are handled and provides a solid background for building a GUI. This project is reasonably ambitious for someone with my background and provides an excellent challenge to finish my college career with.

# Project Specifications

*This section describes****what****the project will do. In this section you should provide any required background information that motivates your project, and you should explain what the goals of your project are. Depending on the project, this section will likely include:*

## Functional Specifications

*Describe the components of the system and what they do, not how they do it. You will need to be detailed here, providing a specific list of functionality for the system. For example, if you are designing a database for CD's, exactly what operations will the user be able to perform on the database? Just saying "search for a CD" is not enough; you need to specify exactly what criteria the search will be based on.*

Example: The programming side will require using iOS’s Core Location framework to retrieve the user’s location and allow them to view it on a map. The application will take the zip code of the user’s location, load it into a URL, and execute a web request. The web request will retrieve the HTML source code of the page, which will then be parsed to retrieve the area codes. With this data, users will be presented a list of results (if there’s more than one), be able to select an area code from the list, and load it into the phone dialer. iOS’s Core Data framework will be used to allow users to save and create a list of area codes. From the list of saved area codes, users will be able to easily select one which will load it into the phone dialer.

## User Interface Specification

*This section will be detailed or not depending on whether the user interface is critical to or the focus of the project. Include details about how the UI will look, what it will include, and could also include how (or if) you will test user feedback about your UI.*

The user interface will feature the standard media player functions: a play button, pause button, stop, fast forward, reverse, and skip. It will also include a volume slider and a shuffle button. Avideom will have a tab bar in the upper left hand corner. This tab bar will have a File tab to allow the user to browse through their files and select music, a Settings tab to allow the user to change more advanced application settings, and another tab with other miscellaneous functions, such as creating a playlist.

# Technical Details

*The content of this section will vary from project to project. In general the goal of this section is to explain****how****you will implement your system. For example, if your project involves the design and implementation of a database, then the proposal should clearly explain: (a) the tables and relationships that will drive the database using a formal model such as an ER diagram; and (b) a plan for populating the database with real data before the project is completed. For other projects it may be important to describe the major data types, data structures, file structures, and algorithms associated with the project. In all cases, justify your choices and mention any tradeoffs used in making your decisions.*

Example: SmartDialer will be using iOS’s Core Data framework to create a local database on a user’s device. There will be an “Area Code” database that has many area code objects. Each area code object will have a “Value” field that will hold the value of the area code. To retrieve the area code information, SmartDialer will utilize iOS’s Core Location framework to retrieve information about the user’s current location. SmartDialer will create a web request in the background with information about the current user’s location and parse the resulting page’s source code. The app will search for a specific element of the page that will contain the area codes. While this may seem like an excessive amount of work to achieve results, the tradeoffs are that it is free/does not require subscription to a service and that there are no area code APIs for Swift.

# Timeline

*The proposal should include a project timeline. The time line should contain clear and well-defined descriptions of the work that must be completed before each of four project checkpoints (see syllabus) and before the final project demonstration (see syllabus). One should be able to use the timeline to understand the implementation process and also to know the deliverables you will present at each checkpoint. You should spend significant time developing the timeline, if for no other reason than to try to determine if the project can be completed in the greater part of a semester.*

*As you work on the timeline, assume you will spend at least 10-15 hours working on your project each week. Thus the amount you can accomplish between checkpoint 1 and checkpoint 2, for example, should require 20 - 30 hours to complete. You must phrase your checkpoint items in terms of tangible deliverables that can be demonstrated. The best way to do this is make a list for each checkpoint of things you will****show/demonstrate****. Consider the following:*

* ***I will have started my data structure design.****NO GOOD. There is nothing tangible to show, so there is no way your instructor will be able to determine if you have completed this task.*
* ***I will demonstrate the insertion of a CD consisting of a title, artist, and year into the database, then I will demonstrate querying the database by searching for all CD's with a specified artist to show that the new CD is found.****GOOD!*
* ***Feedback from user will be obtained.****NO GOOD. No deliverable.*
* ***I will show typed summaries of the notes I took while meeting with five prospective users to let them experiment with my system and critique the user interface.*** *GOOD!*

Example:

## Checkpoint #1 (February 28th)

* I will demonstrate basic audio player functionality: playing, pausing, stopping, fast forwarding, and reversing a .mp3 audio file.
* I will show the basic structure of Avideom’s GUI: the buttons, volume slider, and file navigation for file selection.

## Checkpoint #2 (March 14th)

* I will demonstrate basic video player functionality: playing, pausing, stopping, fast forwarding, and reversing a .wmv video file.
* I will show a more fully fleshed-out GUI, with a better design and a Settings tab for the user to edit more advanced settings.

## Checkpoint #3 (April 11th)

* I will demonstrate the ability to make and save playlists.
* I will demonstrate Avideom’s Smart Shuffle, an algorithm for shuffling songs in the user’s selected playlist.
* I will demonstrate Avideom’s capability to stream radio from the Internet.

## Checkpoint #4 (April 25th):

* I will demonstrate the ability to download Avideom from the Internet as an executable file.
* I will show a complete and fully-functional GUI with several more undetermined graphical features for a better user experience.

# Future Enhancements

*This section lists, in bullet form, at least three enhancements that could apply to your project. These enhancements could either be elements that you might add to your project (time permitting) or enhancements that you would not be able to implement due to time or ability constraints.*

Example:

* Currently, SmartDialer is only a Windows desktop application, but in the future, I would like to extend its platform to Linux or Android.
* In the future, I would like to add the ability to play more obscure file formats for both audio and video.
* Avideom may eventually be hosted on a web server rather than just a downloadable application.

# Bibliography

*The proposal should contain a bibliography that lists all sources. Any material paraphrased or quoted should be cited with an in-text reference. Use standard MLA format.*

Example:

Bisht, Suraj. "Create Music Player App Using Python and Pyglet.” *Bitforestinfo*. Bitforestinfo, 06 Jan. 2017. <http://www.bitforestinfo.com/2017/01/how-to-create-music-player-using-python.html>

"Python 3 Tutorial." *Python Tkinter Course*. Python Tkinter Course, 2016. < https://www.python-course.eu/tkinter\_buttons.php>.

"music-player." *Github*. Sticken88, 14 Apr. 2018. < https://github.com/sticken88/music-player>.