

Degree Plan/Audit Project

Team Members:

Rodolfo Gomez Arteaga

Zachary Young

Apar Singh

Allen Hernandez

Hummad Khan

Advisor:

Laurie Thompson

Table of Contents

Table of Contents	1
Abstract	2
Introduction	2
Project Description	2
Objectives of the project	3
Response to Similar Solutions	3
Our Solution	3
Timetable	4
Milestones	4
Phase I:	4
Phase II:	4
Phase III:	4
Phase IV:	5
Implementation Details	5
Code Documentation	5
Test Case Results	14
Impact And Security	17
Individual Assessment	18
Rodolfo Gomez	18
Allen Hernandez	18
Hummad Khan	19
Zachary Young	19
Apar Singh	20
Issues And Lessons Learned	21
Rodolfo Gomez	21
Allen Hernandez	22
Hummad Khan	23
Zachary Young	24
Apar Singh	25
Future Work	26
Ethics Discussion	27
Signatures Page	28

Abstract

What we are planning on accomplishing is building a Java application that aids Graduate Advisors and Degree Plan Evaluators (DPE) in producing degree plans and audit reports for graduate students. This tool makes the DPE's job more efficient by lessening the overall time they spend on generating degree plans and audit reports. The program will take in a student's transcript and read-in the information about their courses, grades, and any other necessary information. Then the program will show a preview of the degree plan where the DPE is allowed to make modifications to the data fields in the generated preview. The program will then produce the final documents for the audit report and degree plan. Lastly, a .txt file will also be generated that will contain data that can be read-in by our application to produce a degree plan and audit report without having the DPE start from scratch with the reading-in of a transcript. The development of this application is important because it lessens the strain put on DPEs by automating one of their most time-consuming and frequent tasks.

Introduction

Project Description

As the CS Department continues to expand, they are seeking ways to improve the advisors' processes. We aim to streamline the time-consuming task of creating and maintaining degree plans while also improving the process of completing degree audits. This project allows the advisors to better serve the students by providing them with more efficient and effective academic support. (UTDesign, 2023)

Objectives of the project

The objective of this project is to cut down the time that DPEs have to spend on generating an audit for a graduate student. If we had to list our objectives, they would be as follows:

- Develop a tool to assist DPEs in generating and printing Degree Plans quickly and accurately (UTDesign, 2023)
- Improve an existing tool that calculates GPA and required GPAs for degree audits (UTDesign, 2023)
- Generate PDF degree plan for printing with minimal data entry (UTDesign, 2023)
- System must be able to change course lists and degree plans (UTDesign, 2023)

Response to Similar Solutions

We plan to overhaul an early-stage, console-based program demonstrated by the advisors (a simple GPA calculator). We will improve upon the program's presentation and user-friendliness while adding many new and improved features such as:

- Degree Plan and Audit Generation
- Intermediate Editing
- PDF and/or Docx export

Our Solution

Our solution is, for the most part, unique in that our application will take in a student's PDF formatted transcript and read-in the information about their courses, grades, and any other necessary data. Then the program will show a visual representation of the degree plan where the DPE is allowed to make any necessary modifications. A .txt file will also be generated that will

contain data that can be read-in by our application to produce a degree plan and audit report without having the DPE start from scratch with the reading-in of a transcript. Lastly, the program will produce the final documents for the audit report and degree plan. DPE will be able to save the degree plan in the desired location (other generated documents will be saved to the original UTD Box location where the original graduate student transcript came from).

Timetable

Milestones

Phase I:

- ✓ Be able to get the file path of the desired PDF transcript file [Deadline: Relative to phase]
- ✓ Be able to read-in the PDF transcript file [Deadline: Relative to phase]

Phase II:

- ✓ Search through the content read-in from the transcript to retrieve the necessary data needed for the degree plan and audit report [Deadline: Within 1 week of starting Phase II]
- ✓ Create an object to store all the necessary data after parsing [Deadline: Within 1 week of starting Phase II]

Phase III:

- ✓ Use the data collected and stored in the object to check all the requirements for the degree plan [Deadline: Within 1 week of starting Phase III]
- ✓ Create a preview of the degree plan [Deadline: Within 2 weeks of starting Phase III]

- ✓ Allow the DPE to add/change/delete the content displayed in the preview of the degree plan [Deadline: Within 2 weeks of starting Phase III]

Phase IV:

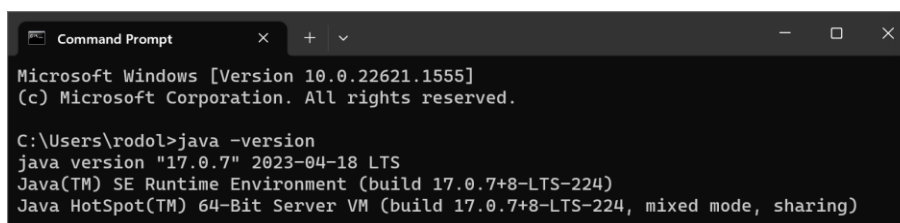
- ✓ Create final documents for the audit report and degree plan [Deadline: Within 2 weeks of starting Phase IV]
- ✓ Create a .txt file based on the object and the data stored within it [Deadline: Within 2 weeks of starting Phase IV]
- ✓ DPE being able to save the degree plan in the desired location (other generated documents will be saved to the original UTD Box location where the original graduate student transcript came from) [Deadline: Within 2 weeks of starting Phase IV]

All phases have been completed in some form or another, in accordance to the deadlines we set. Our application is ready to be demoed.

Implementation Details

Code Documentation

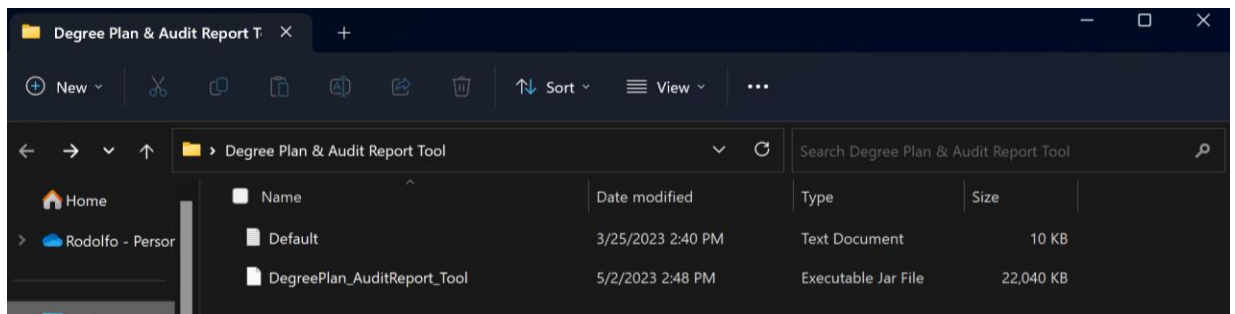
- 1.) It is imperative that the user's computer be using Java 17 or newer. If your computer uses Windows and you want to check which version of java your computer is using you can open the command prompt and run: `java -version`.



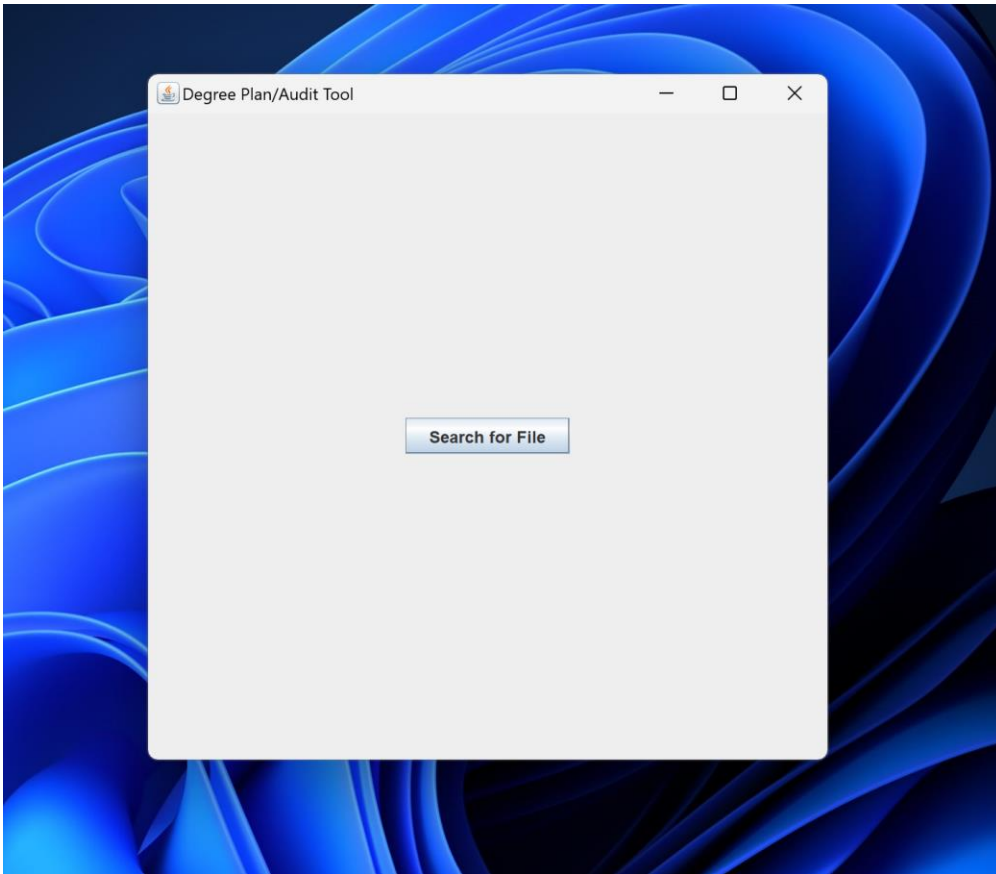
```
Microsoft Windows [Version 10.0.22621.1555]
(c) Microsoft Corporation. All rights reserved.

C:\Users\rodol>java -version
java version "17.0.7" 2023-04-18 LTS
Java(TM) SE Runtime Environment (build 17.0.7+8-LTS-224)
Java HotSpot(TM) 64-Bit Server VM (build 17.0.7+8-LTS-224, mixed mode, sharing)
```

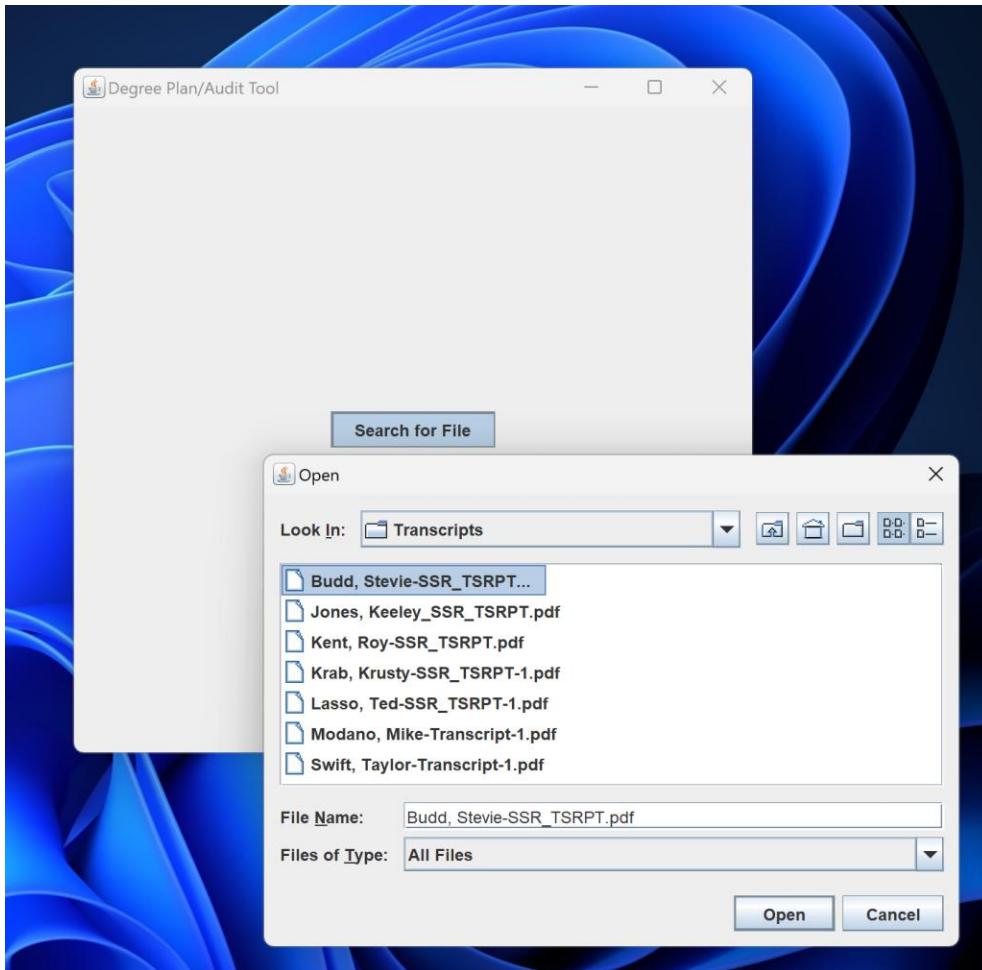
- 2.) The application and other necessary documents are within a folder called, *Degree Plan & Audit Report Tool*. You can have the folder stored anywhere on your computer, but we recommend having it on your computer's desktop.
- 3.) Open the *Degree Plan & Audit Report Tool* folder. You will see the following two items:



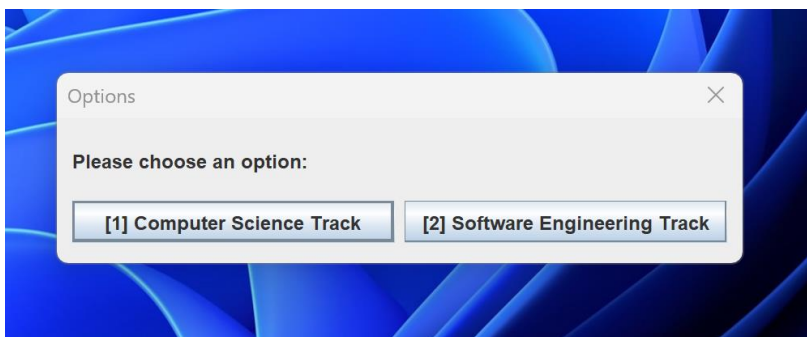
- The *Default* file is a TXT file that contains default data such as default tracks, default leveling courses/prerequisites, and more; this data is extremely important to the application.
 - The *DegreePlan_AuditReport_Tool* file is an executable Jar file that contains the necessary files and dependencies required to run our java application.
- 4.) Double click on the *DegreePlan_AuditReport_Tool* Jar file and the following window will appear:



After clicking the **Search for File** button, a file explorer window will appear and you can either navigate to open a student's transcript (in PDF format) or you can select a student specific TXT file. Further explanation as to what a student specific TXT file is can be found at step 8.

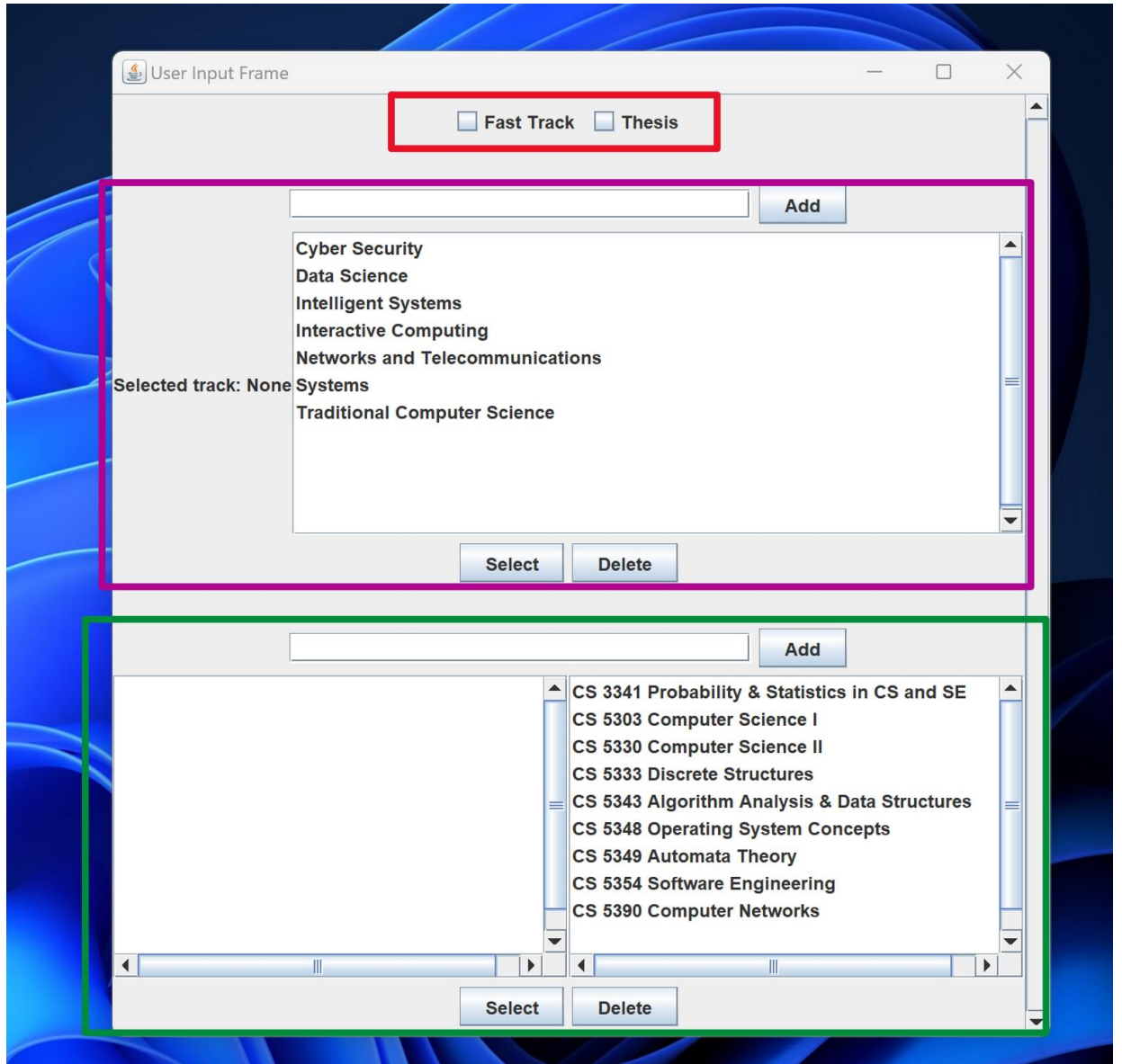


5.) After opening a student's transcript (in PDF format) or a student specific TXT file, the following window will appear:



Select the type of track the student is in or is interested in.

6.) Depending on the type of track you selected, the following window might appear a little different but in terms of capability there is no difference.



The section in **red** allows the user to select if the student is pursuing their master's degree via Fast Track or Thesis.

The section in **purple** allows the user to select the specific track the student will enter. To select the track, the user must click on the track from the list and then click on the **Select**

button. The user can also add tracks to the list by entering the track in the field next to the **Add** button, after entering the track the user must click the **Add** button. The user can also delete tracks by clicking on the track from the list and then clicking the **Delete** button.

The section in **green** allows the user to select which leveling courses/prerequisites the user wants to select. To select the leveling courses/prerequisites, the user must click on the track from the list and then click on the **Select** button. The selected leveling courses/prerequisites will appear on the left list. The user can also add leveling courses/prerequisites to the list by entering the leveling courses/prerequisites in the field next to the **Add** button, after entering the leveling courses/prerequisites the user must click the **Add** button. The user can also delete leveling courses/prerequisites from the list by clicking on the leveling courses/prerequisites and then clicking the **Delete** button.

To proceed, the user has to exit out of this window.

- 7.) The window that follows is the one that shows the preview of the degree plan. Here is an example:

Pre-view of Degree Plan

CORE COURSES (15 Credit Hours) 3.19 Grade Point Average Required

Course Title	Course Number	UTD Semester	Transfer	Grade
Natural Langua...	CS 6320	2022 Fall		B+
Design and An...	CS 6363	2021 Fall		B+
Artificial Intellig...	CS 6364	2023 Spring		
Machine Learning	CS 6375	2021 Fall		A

Add Row
Delete Row

One of the Following Courses

Course Title	Course Number	UTD Semester	Transfer	Grade
Database Design	CS 6360	2022 Spring		B+
Advanced Oper...	CS 6378			

Add Row
Delete Row

FIVE APPROVED 6000 LEVEL ELECTIVES (15 * Credit Hours) 3.0 Grade Point Average

Course Title	Course Number	UTD Semester	Transfer	Grade
SPECIAL TPCS...	CS 6301	2022 Fall		A
ALGORITHMIC...	CS 6385	2022 Fall		A
WEB PROGRA...	CS 6314	2021 Fall		A-
INFORMATION...	CS 6322	2022 Spring		A-
REAL-TIME SY...	CS 6396	2023 Spring		

Add Row
Delete Row

Additional Electives (3 Credit Hours Minimum)

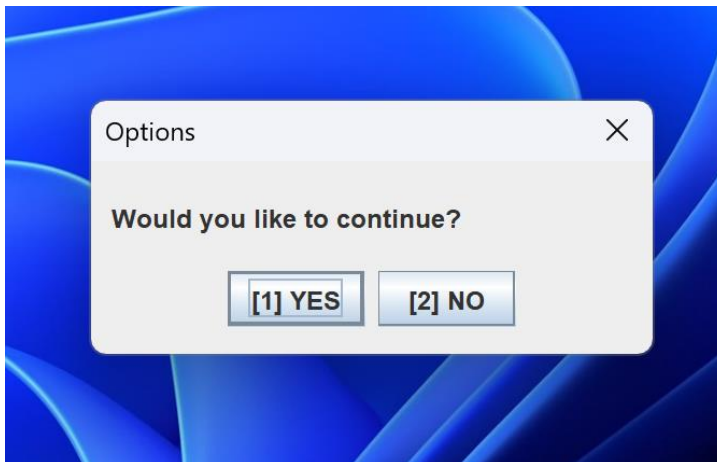
Course Title	Course Number	UTD Semester	Transfer	Grade
--------------	---------------	--------------	----------	-------

Each table has a label above it to tell the user what its contents relates to. The user can add rows simply by clicking on the appropriate **Add Row** button. To delete a row the user selects the row they wish to delete and then they click the **Delete Row** button.

Note that leveling courses/prerequisites that were selected in the prior window will have “>->” next to the course title.

To proceed, the user has to exit out of this window.

8.) A window asking, “Would you like to continue?” will appear:



If you select the **[1] YES** button then you can continue making other Degree Plans & Audit Reports. If you select the **[2] NO** button the application will terminate its run.

It is important to note that as soon as this window appears, the Degree Plan & Audit Report are saved to the same location from which you opened up the student's transcript or the student specific TXT file. The other thing that is saved to that same location is a TXT file which has the students name and “Default” as its name.

> Degree Plan & Audit Report Tool				
Name	Date modified	Type	Size	
Default	4/25/2023 6:08 PM	Text Document	10 KB	
DegreePlan_AuditReport_Tool	5/9/2023 7:37 PM	Executable Jar File	22,057 KB	
Stevie Budd's Audit Report	5/9/2023 8:47 PM	Microsoft Word Doc...	3 KB	
Stevie Budd's Default	5/9/2023 8:47 PM	Text Document	11 KB	
Stevie Budd's Degree Plan	5/9/2023 8:47 PM	Microsoft Word Doc...	4 KB	

There is the student specific TXT file being referred to in the past steps. This file is a combination of the student's transcript data and the data within the Default.txt file. It's meant to track specific modifications made to any of the data.

Test Case Results

1. Degree Plan and Audit Report for Elaine Benes

Degree Plan				
The University of Texas at Dallas				
Masters in Data Science				
Name of Student: Elaine Benes		Fast Track: N		
Student ID Number: 2021511587		Thesis: N		
Semester Admitted to program: 2021 Fall				
Anticipated Graduation: 2023 Fall				
Course Title	Course Number	UTD Semester	Transfer	Grade
CORE COURSES				
Statistical Methods for Data Sciences	6313	2021 Fall		A-
Big Data Management and Analytics	6350	2022 Fall		A
Design and Analysis and Computer Algorithms	6363	2022 Fall		B
Machine Learning	6375	2022 Spring		A
One of the following Course				
Social Network Analytics	6301	2023 Spring		
Natural Language Processing	6320			
Video Analytics	6327			
Statistics for Machin Learning	6347			
Database Design	6360	2022 Spring		A
5 APPROVED 6000 LEVEL ELECTIVES				
ALGORITHMIC ASPECTS/TELECOM NT	6385	2023 Spring		
OBJECT-ORIENTED ANALYS & DSGN	6359	2022 Spring		B-
COMPUTER VISION	6384	2023 Spring		
WEB PROGRAMMING LANGUAGES	6314	2022 Fall		A-
Other Requirements				

Additional Electives				
>>ALGORITHM ANALYSIS&DATA STRUCTURES	5343	2021 Fall		B
Admission Prerequisites				
Computer Science I	5303			
Computer Science II	5330			
>>DISCRETE STRUCTURES	5333	2021 Fall		B-
Algorithm Analysis & Data Structures	5343			
Operating System Concepts	5348			
Probability & Statistics in CS	3341			
>>Automata Theory	5349		Not Required by plan or electives	
>>Software Engineering	5354		Not Required by plan or electives	
>>Computer Networks	5390		Not Required by plan or electives	
* May include any 6000 or 7000 level CS course without prior permission *				
Academic Advisor _____ Date Submitted _____				

Audit Report

Name: Elaine Benes	ID: 2021511587
Plan: Master	Major: Computer Science Major
	Track: Data Science

Core GPA: 3.60
 Elective GPA: 3.17
 Combined GPA: 3.43

Core Courses: CS 6313 CS 6350 CS 6363 CS 6375
 Elective Courses: CS 6385 CS 6359 CS 6384 CS 6314 CS 5343
 Leveling Courses and Pre-requisites from Admission Letter:
 CS 5333 - Completed
 CS >>Automata Theory - Not Required by plan or electives
 CS >>Software Engineering - Not Required by plan or electives
 CS >>Computer Networks - Not Required by plan or electives
 CS 5343 - Completed

Outstanding Requirements
 Core Complete
 To maintain a 3.00 Elective GPA:
 The student must pass 6385 6384
 To maintain a 3.00 Overall GPA:
 The student must pass 6301 5343 6385 6384

2. Degree Plan and Audit Report for Kalinda Sharma

Degree Plan				
The University of Texas at Dallas				
Masters in Intelligent Systems				
Name of Student: Kalinda Sharma			Fast Track: N	
Student ID Number: 2021541957			Thesis: N	
Semester Admitted to program: 2021 Fall				
Anticipated Graduation: 2023 Fall				
Course Title	Course Number	UTD Semester	Transfer	Grade
CORE COURSES				
Natural Language Processing	6320	2023 Spring		
Design and Analysis of Computer Algorithms	6363	2022 Spring		B+
Artificial Intelligence	6364	2022 Fall		A
Machine Learning	6375	2021 Fall		B+
One of the following Course				
Database Design	6360	2021 Fall		A
Advanced Operating Systems	6378			
5 APPROVED 6000 LEVEL ELECTIVES				
SPECIAL TOPICS: COMPUTER SCIENCE	6301	2022 Fall		A
BIG DATA MGMT AND ANALYTICS	6350	2022 Spring		A
WEB PROGRAMMING LANGUAGES	6314	2022 Fall		A-
INFORMATION RETRIEVAL	6322	2023 Spring		
OBJECT-ORIENTED ANALYSIS & DESIGN	6359	2021 Fall		B+
Other Requirements				
Additional Electives				
ALGORITHM ANALYSIS & DATA STRUCTURES	5343	2022 Spring		B+

Admission Prerequisites				
Computer Science I	5303			
Computer Science II	5330			
Discrete Structures	5333			
Algorithm Analysis & Data Structures	5343			
>>Probability & Statistics in CS and SE	3341		Not required by plan or electives	
>>Operating System Concepts	5348		Waived: 21F	
* May include any 6000 or 7000 level CS course without prior permission *				
Academic Advisor _____ Date Submitted _____				

Audit Report

Name: Kalinda Sharma	ID: 2021541957
Plan: Master	Major: Computer Science Major
	Track: Intelligent Systems

Core GPA: 3.56

Elective GPA: 3.61

Combined GPA: 3.67

Core Courses: CS 6320 CS 6363 CS 6364 CS 6375

Elective Courses: CS 6301 CS 6350 CS 6314 CS 6322 CS 6359 CS 5343

Leveling Courses and Pre-requisites from Admission Letter:

CS >>Probability & Statistics in CS and SE - Not required by plan or electives

CS >>Operating System Concepts - Waived: 21F

Outstanding Requirements

To maintain a 3.19 Core GPA:

The student must pass 6320

To maintain a 3.00 Elective GPA:

The student must pass 6322

To maintain a 3.00 Overall GPA:

The student must pass 5343 6322 6320 |

3. Degree Plan and Audit Report for Rebecca Welton

Degree Plan				
The University of Texas at Dallas				
Masters in Software Engineering				
Name of Student: Rebecca Welton			Fast Track: N	
Student ID Number: 2021612458			Thesis: N	
Semester Admitted to program: 2021 Fall				
Anticipated Graduation: 2023 Fall				
Course Title	Course Number	UTD Semester	Transfer	Grade
CORE COURSES				
Object Oriented Software Engineering	6329	2022 Fall		B+
Advanced Requirements <u>Engng</u>	6361			
Adv. <u>Softwr Architect</u> & Design	6362	2022 Fall		B
Software Testing, Validation, Verification	6367	2022 Spring		B+
Advanced Software Engineering Project	6387	2023 Spring		
One of the following Course				
STATISTICAL MTHDS DATA SCIENCE	6313	2022 Fall		A
ARTIFICIAL INTELLIGENCE	6364	2022 Fall		A-
ADVANCED REQUIREMENTS ENGRNG	6361	2022 Spring		B+
DATABASE DESIGN	6360	2023 Spring		
MACHINE LEARNING	6375	2022 Spring		A-
5 APPROVED 6000 LEVEL ELECTIVES				
>>DISCRETE STRUCTURES	5333	2021 Fall		A
DESIGN & ANALYS-COMP ALGORITHM	6363	2022 Spring		B+
SOFTWARE ENGINEERING	5354	2021 Fall		A
Other Requirements				

Additional Electives				
Admission Prerequisites				
Computer Science I	5303			
Computer Science II	5330			
Discrete Structures	5333			
Algorithm Analysis & Data Structures	5343			
Operating System Concepts	5348			
>->Automata Theory	5349		Not required by plan or electives	
>->Software Engineering	5354		Completed: 21F: A	
* May include any 6000 or 7000 level CS course without prior permission *				
Academic Advisor _____ Date Submitted _____				

Audit Report

Name: Rebecca Welton	ID: 2021612458
Plan: Master	Major: Software Engineering Major
	Track: Software Engineering

Core GPA: 3.22

Elective GPA: 3.78

Combined GPA: 3.49

Core Courses: SE 6329 SE 6361 SE 6362 SE 6367 SE 6387

Elective Courses: CS 5333 CS 6363 SE 5354

Leveling Courses and Pre-requisites from Admission Letter:
None

Outstanding Requirements

To maintain a 3.19 Core GPA:

The student must pass 6361 6387

Elective Complete

To maintain a 3.00 Overall GPA:

The student must pass 5333 6360 6387

Impact And Security

The impact this project will have on the graduation audits and degree plans for advisors and their graduate students by making the whole process more efficient. Thus, by making the process faster, the staff will be able to complete more audit requests or be able to work on other tasks such as their own classes. For the students, they will have an ease of mind when receiving their audit request in a much timelier manner. Lastly, hopefully alleviating this process will allow for advisors to be freer to answer their corresponding students. Fortunately, our program and its development process does not pose a substantial, if at all, security threat. Any and all sample data given to our team by the advisor for development purposes had all identifying information changed, and none of this data is carried into the finished product. Our program merely processes data and does not persist it, so our program does not pose a security risk. Storage locations of input and output files are entirely determined by the user. Although we were provided with altered transcripts during development, once the program is shipped, it will be handling sensitive information. Only approved UTD Faculty may be allowed access to the program and all related documents. This includes input transcripts and text files, generated degree plans, generated audit reports, and generated text files. All of these files must be stored in password-protected locations on faculty machines. The program is not responsible for any misused or misplaced data since all generated files are output to the same location the input files were stored.

Individual Assessment

Rodolfo Gomez

In our project, my primary role was focused on researching essential libraries, writing and modifying source code, and designing the user interface. I contributed to the project by researching and implementing the APACHE POI and APACHE PDFBOX libraries, which enabled us to read and manipulate PDF files, as well as Word documents. Additionally, I wrote the source code for reading in PDF files and TXT files. I also played a crucial role in designing the data parsing logic. Moreover, I delved into JAVA Swing to create an intuitive GUI for user input, modify it for the degree plan preview, and update it based on project requirements. I also modified existing data parsing code to account for specific cases and to ensure the application's smooth functioning. To further enhance the application, I developed an algorithm to update default files which contain data that would otherwise be hard-coded in our application. Please refer to the Milestones section to see that I have completed all my tasks, but also to take note that my team has also fulfilled their responsibilities and tasks.

Allen Hernandez

My role in this project was mainly in creating and formatting the two final word documents- the final audit and the degree plan. I first began by researching new technology. The Apache POI library. Additionally, I played a role in utilizing GitHub to its full potential in order to collaborate with team members to split tasks without interfering with each other's code. My specific contributions were the creation of the header, footer, and collaborating to create the automated table cells for the degree plan. Furthermore, I also created multiple functions that

further processed the parsed data. In carrying out this additional processing, these functions made the data readable to the student and graduate advisors. I worked in conjunction with team members to create the formatting and manipulation of data pertaining to the students' courses.

Hummad Khan

My role in this assignment was to create the degree plan table, fill it with the necessary information, and work with my teammates on the DegreePlan and AuditReport classes. We discussed several methods to achieve this task and we decided on using Apache POI to make the Word document. After researching the tool, I replicated the basic structure of the degree plan and managed to recreate the degree plan table by utilizing the Apache POI tool. After the creation of the table, I researched further into cell structures and matched the corresponding cell layout to the degree plan document. Since our team was divided into two groups, I assisted with filling in the necessary info for the degree plan using the parsing algorithm created by my teammates. Furthermore, I designed several of the functions for the degree plan and the audit report. I contributed to the team effort by providing individual support to my fellow teammates whenever they ran into issues with their code. Specifically, I was always ready to help troubleshoot errors, offer advice, and provide guidance wherever necessary.

Zachary Young

My Contribution to the Degree Plan project mostly consisted of formatting the Degree Plan and Audits reports and populating them with user data. The most difficult part of development was learning how to use the Apache POI library. This library allowed us to properly format all of the data into the output documents. Since Apache POI is a very powerful tool with countless capabilities, it came with a steep learning curve. Early versions of the Document generation tools

took a very hard coded approach, not leaving much room for variance in output. I turned our code into a more iterative approach that can generate the degree plan dynamically from a list of courses and titles with varying length. As a team we further developed the code structure by Implementing the Graduate Student and Section classes, allowing the student object to hold a Degree Plan and Audit Report object for easy generation. The Section object was a way to store and order user data and dynamically allocate these sections into the final output. These changes improved the readability and encapsulation of our program and proved quite useful when linking the Document tools to the GUI. I feel that our team worked well together and we helped one another problem-solve through all stages of development

Apar Singh

Throughout our project, my main contribution to the source code was in designing and developing a class that would display all the courses of the student into their particular tables in the GUI. The class had to display the courses which are default to the degree plan as well as the courses read-in from the student's transcript. I developed many methods to iterate through the passed-in data and display it in the user interface. I also created an algorithm to populate the missing data of the default courses with data from the read-in courses. I also allowed the user to interact with and edit the table of courses to create more functionality for the user. The current source code we have is also able to deal with differentiating the special courses and populating the specific table. You can refer to the TimeTable to check that all the tasks have been completed. I communicated with my team members about any issues I was dealing with and also tried to help them with their problems. I provided my insight on developing and improving the application throughout the process and tried to get clarification whenever I did not understand, which could have allowed other members to also better understand certain topics.

Issues And Lessons Learned

Rodolfo Gomez

Throughout the duration of the senior design course, I faced various challenges but from these challenges I learned valuable lessons. My primary responsibilities included researching various libraries, writing and modifying source code, and designing the user interface for the application. One of the major challenges I encountered was familiarizing myself with the APACHE POI and APACHE PDFBOX libraries. These libraries were essential for our project, as they allowed us to read and manipulate PDF files. To overcome this challenge, I spent a significant amount of time researching these libraries and understanding their functionalities. This effort enabled me to write the source code for reading PDF files and extracting the necessary information. Additionally, I implemented these libraries using Maven. Here are the steps to add library dependencies using Maven:

1. Identify the required library dependencies for your project.
2. Open your project's pom.xml file, which is the configuration file for Maven.
3. Add the dependency information for each required library to the dependencies section of the pom.xml file, including the group ID, artifact ID, and version.
4. Save the pom.xml file.
5. Run the Maven build command, which will download the required libraries from the central Maven repository and add them to your project's classpath.

Another issue I faced was designing the GUI using JAVA Swing. I had to ensure that the user interface was intuitive and user-friendly. To achieve this, I researched JAVA Swing and collaborated with my team members in designing an interface that met the project requirements.

Throughout the project, I also had to modify existing code and create algorithms to handle specific scenarios and edge cases. This involved debugging and testing the code, as well as considering the different ways users might interact with the application.

In terms of concepts from previous courses that helped me, my experience in Object-Oriented Programming was particularly helpful in structuring and organizing the code. This foundation allowed me to create efficient and maintainable code.

A major lesson I learned during this term was the importance of effective communication and collaboration within a team. Working together, we were able to brainstorm, troubleshoot, and develop solutions more effectively than if we had worked individually.

To further improve the senior design course, I suggest providing more guidance on the project; possibly treating it like an internship. This would allow students to get better hands-on help and it would allow the students to better understand the work environment they will be entering into after they graduate.

Allen Hernandez

One of the issues I faced during this semester was the learning curve using the APACHE POI library. The documentation did not provide examples of the uses of the library's methods and classes. This lack of documentation led me to find the information I needed in other places, some examples being conducting searches on third party websites, and continuous testing of different methods to note the interactions between them. Another issue I faced was the testing of the methods I developed without actual varying data. The solution that I used was to get the data from one transcript, since that method wouldn't waste time trying to develop the parsing method that the other team was already developing. One of the most useful skills that I was able to learn

more about in the class was the use of Git commands in the terminal. This semester, I worked on three semester long projects with teammates of varying experience in coding and using git/GitHub. Thus, the consistent practice of git commands in the array of different projects I encountered throughout the semester allowed me to increase my knowledge and confidence in using git. Thus, I was able to bring this growing knowledge of the intricacies of git to this project, as well as my experience in previous semesters working on these bigger collaborative projects to smoothly interact with the project and collaborate in a capacity stronger than I would have before all of that growth in my git knowledge and comfortability. My ideas to improve the senior design course would be to provide a lab time with a technical advisor that would come in during lab time and talk over with the team about any questions, concerns, or simply provide overall guidance.

Hummad Khan

My biggest challenges this semester was caused by GitHub and VSCode. Before this project, the only IDE I had experience with was IntelliJ and my experience with GitHub was only uploading my project onto the site. This changed this semester because we agreed as a group that we would all use the same IDE so that we wouldn't run into different errors due to differing IDEs and we would be able to help each other out with any issues or errors that occur. My problem started after opening the initial source code from GitHub into VSCode. I noticed that VSCode wasn't able to compile the project for me but it was working perfectly for my teammates. I've tried multiple suggestions but I learned that the program is riddled with errors when I attempt to click "Run Code" but compiles and runs just fine when I click "Run Java" instead. I never solved this issue since it was unnecessary when the program runs just fine using another method, but it only

left me more confused with VSCode. I had trouble pulling from and pushing into GitHub and had to learn to move my changes to another document and manage the branches accordingly before attempting either. Merging branches also caused a lot of issues. Merge would be successful on GitHub but my program on VSCode was malfunctioning. The solution I found was to create a new project and pull the new source code again. Learning Apache POI was a separate issue that I had to study and research the library in order to understand how to manipulate the table and cell data. A lot of the information was found through multiple sites since it was confusing to understand if I had used only the library.

Zachary Young

There were a number of challenges throughout our development process. First and foremost, this was my first experience doing a project of this size in purely java, which caused some unexpected issues. Learning how different JDKs and build tools interact with one another was essential to a smooth development process. I also learned to carefully monitor my GitHub pulls and pushes to ensure that our group's development as a whole was moving as one, and not overlapping or erasing each other. Learning how to properly set up a GitHub connection on VSCode was very helpful in the development process. Another major roadblock was the Apache POI library, a very powerful and useful library that proved essential to our development process. It was one thing to simply add text to a word document, but to properly format the output documents in a way that we were happy with required a tenuous grasp on how the library functioned. This involved a number of Document, Table, Paragraph, and Run objects all nested inside of one another and countless hours combing through Apache documentation in order to understand.

Apar Singh

One of the issues I faced while developing my part of the source code was figuring out how to populate each table with their specific courses because when the tables are created, each table is its own instance, so there would be 5 or 6 different instances depending on the degree plan. The problem would occur when the user edits any of the tables, and for that table's instances, the change would be saved, but no change would be seen in the other table's instances. As a result, the overall course data would not be saved equally for the other members to use. I worked with my group member to figure out a solution which was first to save the original data that was created with all the courses of the student in the calling class. Then, the new data would be replaced in the calling class when any changes occurred. It would be passed on to the original class. This solution allowed the data to be the same throughout all the instances. Another issue I faced was connecting to GitHub through Eclipse. I could not push my code to the branch because the IDE kept asking me to sign into my GitHub account and displayed an error every time. However, I was able to pull the code of the other members. I also tried to change the IDE to VSCode, but a different issue occurred where the project could not run. I ended up just uploading the class files that I had edited to GitHub. Some major concept I utilized from my previous classes was the OOP principle of polymorphism and inheritance. These concepts allowed me to derive methods and properties from another class and implement their interface differently for our project. I learned about the SWING and APACHE POI libraries, which helped me succeed in this project. I believe this course can be improved by implementing more time to meet with the advisors to discuss issues or topics about the project.

Future Work

If we had more time to improve the project outcome, we would suggest implementing the following features and updates:

- Improved GUI design: The creation of a degree plan table could be implemented only once to allow for the other classes to better populate the data. Overall, fix any of the edge cases the user could deal with and improve the flow and design of the application.
- Integrate a database to store and retrieve default courses and saved degree plans. This would allow users to easily access and modify their saved degree plans.
- Displaying the number of hours, the student completed and the number of hours they still need to complete.

- Ex: EARNED: 31 Hours NEEDS: 64 Hours

- It would be simpler to understand what the student still needs to accomplish if the hours needed were separated between core courses, electives, prerequisites, etc.

- A legend at the bottom of the page which would allow us to use characters and symbols to briefly explain certain characteristics about the course.

- Ex: *LEGEND*

IP = Course is in progress

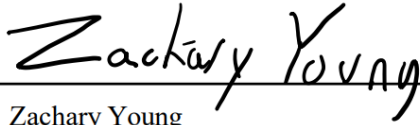
>> = Leveling Course

Ethics Discussion

Before we started any programming, we discussed as a team how we would create the auditing tool and decided on utilizing Apache POI for this project since it was the most straightforward method, we could find for using java to write into a Microsoft Word document. From there, we managed to find a simple program from [geeksforgeeks.org](https://www.geeksforgeeks.org) that was able to implement tables onto a word document. We did end up using some of the code but we didn't hesitate or believe it was against our ethical principles because the program gave us an initial and basic understanding of how we could utilize Apache POI since none of us had any former experience with the tool. The program from geeksforgeeks was also only a snippet of what we managed to accomplish since we had a far larger goal of the kind of table we needed and of what we needed to write into the table. We iteratively pulled information from a PDF and pushed the data onto the table but the program only showed us how to manually write into the table itself.

The portion we used was a basic code structure we used to better grasp the concept of ApachePOI. That said, the small snippet of code was transformed and modified to the point that it is unrecognizable by the time we reached the final product. For this reason, we believe the final product is 100% fully original work that we coded ourselves without the use of code generators or external libraries. We're estimating that the code in our final product that came from open-source libraries would be approximately 5%, accounting for the ApachePOI library.

Signatures Page



Zachary Young



Apar Singh



Allen Hernandez



Faculty Advisor, Laurie Thompson



Rodolfo Gomez Arteaga



Hummad Khan