**TABLE OF CONTENTS**

1. **PROJECT METRICS………………………………………………………. 2**
2. **IMPLEMENTATION DETAILS………………………………………………………………..….. 3**
3. **IMPACT AND SECURITY.…………………………………………………………………. 5**
4. **INDIVIDUAL ASSESSMENT…………………………………………………………….. 6**
5. **ISSUES AND LESSONS LEARNED…………………………………………………………………. 12**
6. **FUTURE WORK…………………………………………………………………….. 14**
7. **Appendix………………………………………………………………… 15**
8. **SIGNATURES, NAMES and DATE……………….……………………………………………………… 16**
9. **Project Metrics**

This semester, our team was able to create a large part of the Degree Audit Program. Even though we were not able to deliver on every feature we used our combined knowledge to create a program we are proud of. No member of the team had ever worked on a system so complex before, and making it from scratch presented challenges none of us had faced. This project provided our team with a preview of what it is like to work with a team towards a common goal.

As a team, we were able to complete the following:

* Several GUI panes
* Transcript Selection
* Transcript Parsing
* Data storage as a Student object
* Editable options for the audit.
* Printing and saving the degree audit
* Printing saving the majority of the Degree Plan

Our team worked diligently to deliver the best project we could.

1. **Implementations Details**

Our team decided to use the IntelliJ IDE to code our project. With this, we were able to utilize several Java APIs such as PDFBox for our parsing, and Java Swing for our GUI components.

The design of our project consists of 8 classes:

1. DegreePlanApp - This class is the main class in our program and is in charge of calling the StartUp class which would initialize the GUI and prompt the user to select a file. After the transcript is selected it is passed onto the PDFParser class where it is tokenized and returned as a Student object. The user can then edit the options of the students such as prerequisites. From here, this class would take the Student and create a degree audit from the gathered information. You are able to save this degree audit to your computer and print it as a docx file. Then, the DegreePlanApp calls the GenerateEditagbleDegree class which creates the degree plan.
2. StartUp - The StartUp class would create the first GUI window that the user encounters. This window prompts the user for a Transcript or a Student file using a file selector from JSwing. The file information is passed back to the DegreePlanApp class.
3. PDFParser - The sole responsibility of this class is to extract the data from the PDF Transcript selected by the user. It would find and return the student information such as name, ID, and courses.
4. Student - The Student class was responsible for maintaining the data for each student that is input. Here, the student’s name, ID, courses, prerequisites, enrollment date, and more were kept. The courses were saved for each student using the Course class.
5. Course - This class kept track of the available courses for the program. Methods here are responsible for formatting the courses so they can be stored as an array of the Course object type. These arrays are used throughout the rest of the program.
6. GenerateEditableDegree - This class simply passes the Student information into the SpreadsheetUI class and instantiates it.
7. SpreadsheetUI - This class is responsible for generating the editable degree plan using a spreadsheet UI through the use of APIs. After the information is displayed it can then be edited to the users liking. This means adding and removing courses as well as editing student information. After the information is input the user is prompted to save the file as a PDF.
8. StudentSave - The goal of this class was to save the Student object as its own file type. This way the user could input the student file to retrieve their information. This was achieved by serializing the Student object and saving it in its own file to be deserialized later.
9. **Impact and Security**

**Impact:**

Improved Student Experience: The new degree planning or audit project has the potential to significantly enhance the student experience by providing a user-friendly platform for students to navigate, plan and edit their degree courses. The application’s graphical representation of the degree plan can help students visualize their academic progress and make informed decisions about their future course selection.

Improved Student Retention: By providing a tool that allows students to more easily track their academic progress, the new degree planning or audit project can help i,prove student retention rates. When students are better able to understand their academic standing and have the ability to make informed decisions about their course selection, they are more likely to remain enrolled and complete their degree on time.

Increased Efficiency: The degree planning or audit project can also contribute to increased efficiency in the academic planning and advising process. Advisors can quickly and easily access a student’s degree plan and provide targeted guidance and support. This can help reduce the time and effort required to manually track student progress and can free up resources for other important tasks.

**Security:**

In the current stage of our degree plan/audit project, we have not yet integrated any security measures. Nevertheless, it is crucial to acknowledge the significance of safeguarding sensitive information from potential threats. As we manage personal and academic data, it is our duty to prevent unauthorized access and misuse by unauthorized individuals. Implementing robust security features such as password-protected resources, data encryption, and access control mechanisms will provide our users with the assurance that their information remains confidential and secure. By expanding upon these protective measures, we can enhance the user experience while simultaneously maintaining the integrity of the data within our system. This proactive approach will ultimately contribute to building trust with our users and stakeholders, fostering long-term success for the project.

**How to run the program:**

1. Ensure that JDK 19 is installed on your system.
2. Download the Cap\_Proj zip folder and extract it.
3. Inside the Cap\_Proj navigate to the following file:

‘Cap\_Proj -> out -> artifacts .\_ Cap\_Proj.jar

1. Run the Cap\_Project executable jar file.

From here you can select a Transcript to begin the process.

1. **INDIVIDUAL ASSESSMENT**

Adam Kosicki (project lead) :

As the team leader of our senior project , I played an important role in leading our team towards a successful project completion. At the start of the project, I helped initiate the project by contributing to the design and planning stages. Throughout the project, I was responsible for managing deadlines, assigning tasks and roles, and making sure that everyone had the support they needed to complete their work.

I was also instrumental in key technical aspects of the project. For example, I implemented the saving and loading of student objects, which was a critical functionality for our project. I took part in starting and editing the GUI, which connects Java code from our different project versions together and enabled us to build on it. I also organized the team with instructions on what tasks were remaining and how we could complete them.

Moreover, I focused on communication among team members, by passing information from one team member to another when we were unable to meet and get the details out properly. For example, I helped troubleshoot issues when team members were struggling with running or compiling the project, and made sure that everyone could contribute. I also decided on and found all the APIs we used, which were critical for our project functionality.

Finally, I ensured that our project was easily accessible to users by exporting it in a jar file. Overall, I am proud of the work that our team has accomplished, and I am grateful to have worked with such a talented and dedicated group of individuals.

**Adrian Sanchez (Transcript and Audit):**My role was being in charge of the Audit and reading the students Transcripts in order to extract the relevant data. The first thing our group needed to do, to get up and running, was to extract all the relevant data from the PDF and be able to use that data in order to create the Audit and Degree Plans. As I had to learn how to use regex and how to correctly trim the data to separate course codes,names,hours,credits, and more, it took me a while to get this part done. I finished it on the week of March 6th, which gave us time during spring break to work on the Audit and Editable PDF. I had to hardcode the file path in order for it to work, up until my team members were able to create a GUI where I can select a file. Once the Transcripts were able to be read, I stored the information into a Course class and a Student Class, so that I could properly separate cores,electives,leveling courses, and calculate GPAs. Then with my createOutputDOCX method, I had to learn how to use the Apache API to properly print everything on a docx file. I have been working on the Audit report since then and the current deadline is 5/4. Currently I completed the task of printing ID, Name,track, GPAs, Cores, Electives, Leveling and Pre-reqs, and Outstanding Requirements classes. However, I do have a few pending tasks left. Even though I could print Leveling Courses and Pre-reqs I could not print the semester and year, or if it was required by the degree plan. I also had wrong formatting for outstanding requirements. Thus, for these things I will be pushing my deadline to 5/9 so that I can hopefully have something better to present in the Demo.

**Jonathan Vu (DegreePlan)**

As a member of the team, my primary responsibility was the DegreePlan component. This component accepts either a transcript PDF or a student data object, which determines the subsequent process. If a student data object is selected, the information for the degree plan is immediately generated. However, if a transcript PDF is chosen, the user must specify three components: the chosen track/degree plan, select prerequisites assigned from the list, and indicate whether a student is on the Fast Track or pursuing a thesis. Once this is done, the degree plan is generated similarly to the student data object. Creating the degree plan was the most challenging aspect of the project as we had to accept the student data and display all the relevant information accurately. However, after spending some time on it, we were able to manipulate the code to display the necessary information correctly and move on to generating the audit report. Additionally, I identified and fixed a minor issue at the beginning of the program where the program would still proceed to the degree plan component even if no file was selected. I resolved this issue by adding a section of code to throw an error message, which would prompt the user to select a file before proceeding. I am proud of my team's effort and grateful for their assistance in fixing and addressing my issues in the project. If given more time to work on the project, we would have gone back into the code to eliminate unnecessary coding, which we ended up scrapping for the overall program.

**Grant Reed (Application start and options field) -**

As a member of the project team, my primary role involved working on the application startup and the options window. I worked on how the program would perform when it is launched. I also helped write the code for the options field where you can select the student transcript as well as the prerequisites.

For the application to start I worked with Adrian to prompt the user for a transcript or student within a GUI. This PDF file would then be parsed with our PDFParser to extract and store the data in a student class, which I also worked on. The program then needs to detect the type of file given and choose what to do next.

I also worked on the GUI that displays additional options after the transcript is uploaded. The prerequisites, fast track, and thesis masters all needed to be editable. Once that data was collected it needed to be stored. From there, the program generates the audit and displays the degree plan.

Overall, I am satisfied with the work I provided on the project and I feel I was a valuable member of our team. I tried to keep the team on track and motivated which I feel helped raise morale. Even if we are unable to deliver on all the features I will be happy knowing I did my best.

**Jason Vu (Student Class and GUI)**

As an integral part of the project team, I focused on developing the student class and refining the serialization process. My work involved creating a user-friendly class structure that allowed students to select their courses and majors easily through the GUI. I also incorporated a feature that simplifies finding courses and their respective prerequisites, streamlining the course selection process for students. Additionally, the student object is designed to calculate their GPA for each class and determine the necessary GPA to maintain their academic standing. To help with data management, I implemented serialization, which enhanced data readability and maintainability.

If I was done with my part, I would communicate with my team to make sure we are up to date on what else needs to be done or would help if there was a problem with a teammate’s issue and try my best to fix their problem. I also made sure to help out make the GUI functionality work better and quality of life to make it look nice.

Overall, this project took a lot to manage, but I am happy that me and my team did our best to make this project work to the best of our abilities and definitely learned a lot of new things I did not know before working on this project.

**Michael Wu(Degree Plan GUI):**I was largely responsible for designing and implementing the GUI for displaying the generated degree plan and making edits to it as needed. I also helped with managing and setting up the GitHub repository that we used to collaborate on the project and merge our individual pieces of code together.

My component grabs the course and student info collected by the other team members’ components and generates an editable degree plan using that info. This was done by having the GenerateDegreePlan class create a degree plan GUI object from my GUI class and have it call the necessary methods from the degree plan GUI class to pass it the student’s info as well as info about any relevant courses they’ve taken for their chosen masters’ track. The GUI then creates the degree plan table from that info and fills out the student’s name, id, track, and semester admitted fields using the info given to it by the GenerateDegreePlan component as well as generates the track specific core and leveling courses that the student needs to complete for their chosen degree plan. It also generates empty sections for the approved and additional electives that the student needs to complete. After that, it takes the student’s completed courses relevant to their track and fills out the info for the courses on the degree plan that the student has already completed. The GUI then allows the user to edit the degree plan by changing individual cells in the table. The user can then choose to save the degree plan table as a pdf which the user then stores by browsing and then selecting a location in their computer’s files.

1. **ISSUES AND LESSONS LEARNED**

Regarding the step-by-step instructions, we have included detailed guidelines in the appendix to assist users in installing, using, and potentially extending the features of our project. These instructions cover the setup of necessary dependencies, compilation and execution of the application, as well as explanations of available functionalities and how to utilize them effectively.

In terms of major concepts used from previous courses, we leveraged the knowledge of regular expressions acquired from CS 3377. This understanding greatly facilitated the process of extracting relevant information from the PDF data using regex patterns.

Throughout this term, we learned how to effectively use Maven as a build automation tool and how to implement a GUI using JFrame. These skills were crucial for streamlining the development process, managing dependencies efficiently, and enhancing the overall user experience of our application.

Reflecting on our experience with the senior design course, we believe there are a few areas that can be improved further. Firstly, incorporating more real-world, industry-oriented projects can help students gain practical skills that align with the demands of the job market. It would be beneficial to focus on projects that utilize popular frameworks, tools, and technologies commonly used in professional software development.

Additionally, providing explicit guidance and resources dedicated to learning and utilizing tools like Maven and GUI frameworks would be valuable. These tools are widely employed in industry projects, and having a solid understanding of them can greatly enhance students' readiness for future software development endeavors.

Moreover, fostering collaboration and teamwork within the senior design course can replicate a professional environment and allow students to develop crucial skills in communication, project management, and collaborative problem-solving. Encouraging interdisciplinary projects and facilitating effective team dynamics can help students gain a holistic perspective on software development.

By incorporating these enhancements, the senior design course can better equip students with the practical skills, knowledge, and experiences necessary for a successful transition into the professional world of software development.

1. **FUTURE WORK**

If given more time, we would suggest implementing more error handling and validation checks to make the program more robust and user-friendly. This would help prevent crashes and ensure that the program is giving accurate results. Additionally, we would explore the possibility of integrating machine learning algorithms to enhance the accuracy of the program and reduce the risk of errors. Furthermore, we would suggest improving the user interface by adding more interactive features, such as drag-and-drop functionality and graphical representations of data. This would help users visualize their degree plan and make it easier to make changes as needed. In terms of updates, we would also explore the possibility of integrating with other educational resources such as online courses, textbooks, and academic journals. This would help provide more comprehensive information to users and help them make more informed decisions about their degree plans. Overall, we believe that by implementing these suggestions, the program would become more efficient, accurate, and user-friendly, providing a better experience for users.

1. **Appendix**

To extend the appendix of our project, we have compiled a list of references that we found particularly useful during our development process. These resources provided valuable insights, guidance, and documentation that aided us in various aspects of the project. The references include:

"Apache PDFBox Documentation" - Official documentation for Apache PDFBox API, which helped us understand the functionalities and usage of PDFBox for reading and manipulating PDF files.

"Java Regex Tutorial" - An online tutorial that explained the concepts and usage of regular expressions in Java, which were crucial for extracting information from the jumbled PDF data.

"Apache POI - HSSF and XSSF API Documentation" - The official documentation for Apache POI library, which guided us in using POI for generating docx files and manipulating their contents.

"Maven: The Complete Reference" - A comprehensive guide to Maven, which provided step-by-step instructions on configuring project dependencies, managing build lifecycles, and utilizing plugins effectively.

"Java Swing Tutorial" - A tutorial series that covered the basics of Java Swing and JFrame, helping us understand how to create a graphical user interface for our application.

1. **TIMETABLE**

**Phase I**: due **3/10/2023**: Read the PDF and extract information to make the degree audit.

* Phase 1a: Writing code to read all the information off the pdf
* Phase 1b:Formatting the information read off the pdf
* Phase 1c: Write logic to get specific needed portions of information and store them into the appropriate data structures
* Phase 1d: Write code to output specific information needed to generate degree audit

**Phase II:** Get user input and use input to generate editable degree plan

* Phase 2a: Get pdf transcript from user
* Phase 2b: Get student’s selected track from user
* Phase 2c: Get leveling courses/pre-reqs from user
* Phase 2d: Get whether student is on fast track as well as whether student is pursuing a thesis or non-thesis masters from user
* Phase 2e: Use information extracted from transcript PDF as well as information given by user to generate degree plan

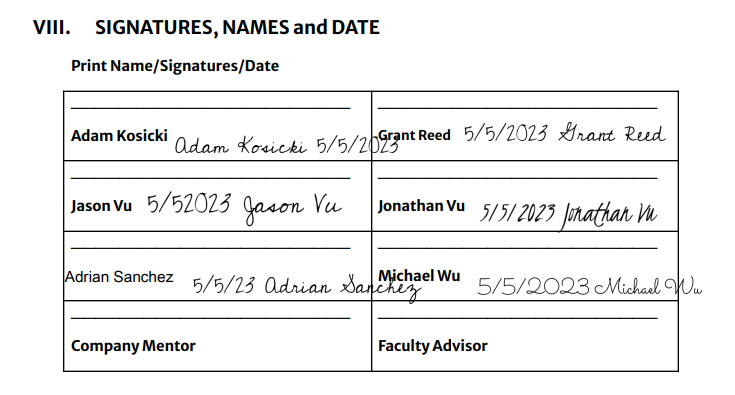
**Phase III**: due **4/10/2023**: GUI implementation for our program.

* Phase 3a: Display completed core, elective, and leveling courses
* Phase 3b: Have editable fields in generated audit report that allows user to add, delete, and move courses and pre-reqs by dragging and dropping fields
* Phase 3c: Allow for user to save and print degree plan in user-chosen location
* Phase 3d: Ask user for confirmation for continuing audit report

**Phase IV**: due **4/28/2023**: Combining both phases I, II, and III into the final program and complete code to generate audit report.

* Phase 4a: Display core, elective, and overall GPAs on generated audit report
* Phase 4b: Ask user if student will be taking additional courses and if so, ask user for number of additional courses
* Phase 4c: Ask user to choose disposition of all uncompleted pre-reqs
* Phase 4d: Allow user to print and save final audit report in user-chosen location

**Link to Github: https://github.com/MichaelDW851/CS-Project---Team-8**

****