

# Freshmen Pre-Packaging Software

Software Specification | Installation Instruction | User Manual



*Abhash Panta*

# TABLE OF CONTENTS

---

1	Introduction.....	1
1.1	Purpose .....	1
1.2	Definitions .....	1
1.3	User Description .....	1
1.4	User Functionality Requirement .....	1
1.5	Platform .....	1
1.6	Technical Requirements .....	2
1.7	Summary .....	2
2	Installation Instructions .....	3
3	User Manual.....	8
4	Design Documentation .....	14
4.1	Design Summary .....	14
4.2	Key Components/Aspects.....	15
4.2.1	Graphical User Interface (GUI).....	15
4.2.2	Web Scraping Four-Year Plans.....	15
4.2.3	I/O Manager.....	15
4.2.4	Scheduler .....	15
4.3	Description of Classes .....	16
4.3.1	Athlete .....	16
4.3.2	AthleteReader .....	16
4.3.3	AthletesExcelConstants.....	16
4.3.4	Course.....	17
4.3.5	CourseExcelConstants.....	17
4.3.6	CourseReader.....	17
4.3.7	CreditInfo.....	18
4.3.8	ExcelReader.....	18
4.3.9	FYP_Category .....	18
4.3.10	FYP_CourseElement.....	19
4.3.11	Interval.....	19
4.3.12	IOManager.....	20
4.3.13	Major .....	21
4.3.14	Prepackager .....	21
4.3.15	Scheduler .....	22
4.3.16	Sport .....	22

4.3.17	SportsTimeExcelConstants.....	23
4.3.18	SportsTimeReader.....	23
4.3.19	Student.....	23
4.3.20	StudentExcelConstants.....	24
4.3.21	StudentReader.....	24
4.3.22	TestingInfo.....	24
4.3.23	Webscraper.....	25
5	Test Plan and Test Results .....	26
5.1	Unit Testing Plan.....	26
5.2	Unit Testing Results .....	26
5.3	Integration Testing Plan .....	26
5.4	Integration Testing Results .....	26
6	Summary and Conclusions.....	27
7	Bibliography.....	28

# 1 INTRODUCTION

---

## 1.1 PURPOSE

To design and develop a desktop application that automates the process of prepackaging (scheduling courses) for college freshmen based on their characteristics, availability of courses and the respective major's four-year plan. The importance of this software lies in the tremendous amount of time academic advisors spend in order to prepackage each student manually. Therefore, the main purpose of the software is to save the time for advisors through automation.

## 1.2 DEFINITIONS

*Pre-packaging* - The process of finding the appropriate courses and schedule for each incoming freshman for their first semester based on their credentials/characteristics.

*Web-scraping* - The process of extracting information from webpages.

## 1.3 USER DESCRIPTION

The user of this software are the academic advisors who are responsible for prepackaging students. The software is expected to automate most of the work, it will still require supervision and rectification from the users based on the users' needs.

## 1.4 USER FUNCTIONALITY REQUIREMENT

- Users should be able to feed in several input files: Student List, Course List, Athletes list and Sports Practice Times.
- The software is expected to download and store the four-year plans for all the majors every academic year from the web.
- Users should receive a readable (excel or csv) file as an output that contains the list of all the prepackaged students along with the specific courses recommended by the software.
- A detailed user manual that demonstrates/explains the usage of the software must be provided along with the software.
- Guidelines for installation of the software is also required along with the User Manual.

## 1.5 PLATFORM

The software will be developed as a GUI desktop application that can be run on Microsoft Windows Operating System.

## 1.6 TECHNICAL REQUIREMENTS

- The software will need to be a GUI application. Therefore, GUI elements need to be carefully developed for better user experience.
- Web-scraping will be needed in order to obtain the four-year plans for all the majors from Ramapo College's Website.
- The process of taking the input from user (files/other information) must be user-friendly and the output should be clearly presented in a readable format (preferably excel/csv/txt).

## 1.7 SUMMARY

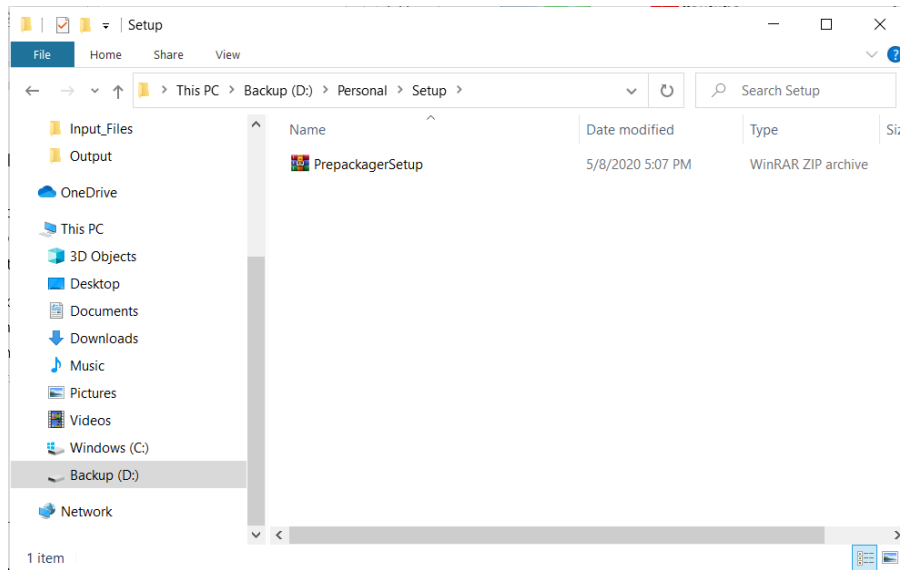
The prepackaging software contains a lot of small constraints and requirements both in terms of functionality and technicality. Although the process seems simple, certain aspects it contains exceptions and extraneous cases which need to be handled effectively. Overall, the goal of this software is to simplify the workload of the academic advisors by producing the list of courses that students will be taking in their first semester.

Having said that, the results obtained from the software are not meant to be final and absolute. In special circumstances, advisors might need to prepackage students manually. However, the chances of such special circumstances remain relatively low. Thus, this software will improve the productivity of advisors and efficiency of the prepackaging process in the long run.

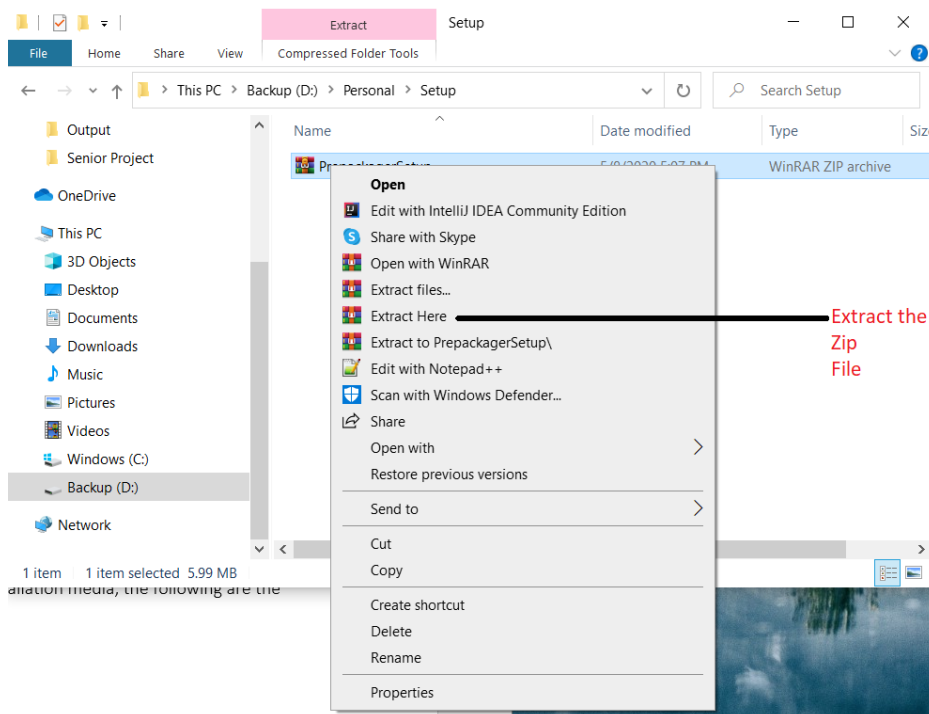
## 2 INSTALLATION INSTRUCTIONS

The prepackaging software can be installed in any Windows machine using the installation media. Assuming that the user already has the installation media, the following are the detailed steps for installation:

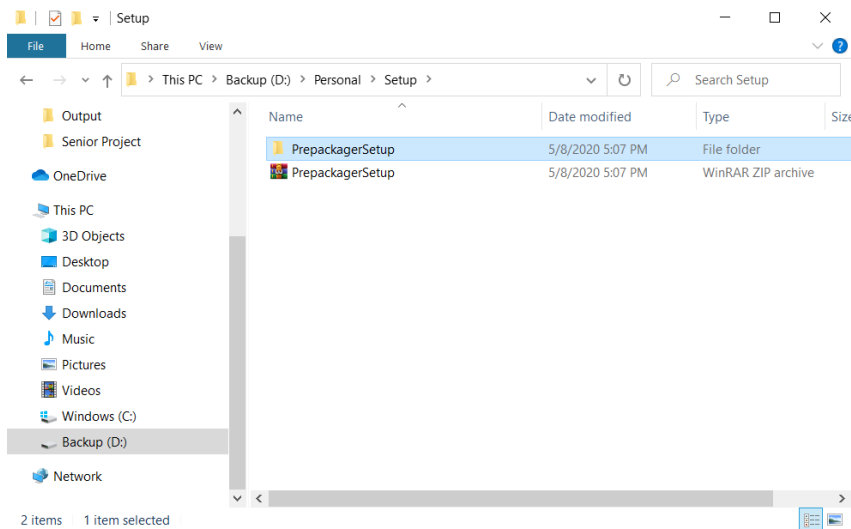
- I. You must have the zip file which contains the *PrepackagerSetup* files.



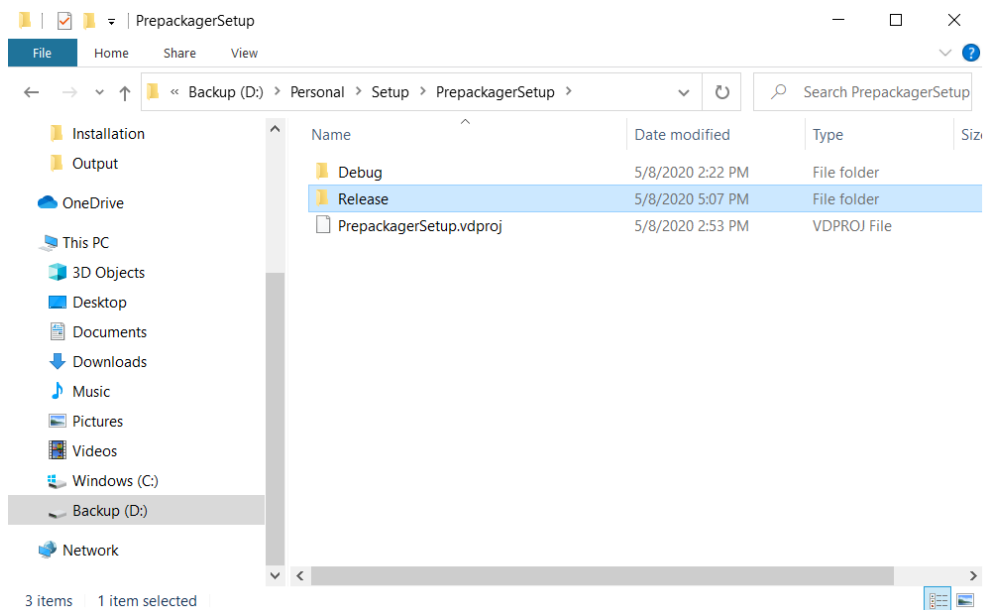
- II. Extract the contents of the zip file into a folder of your choice.



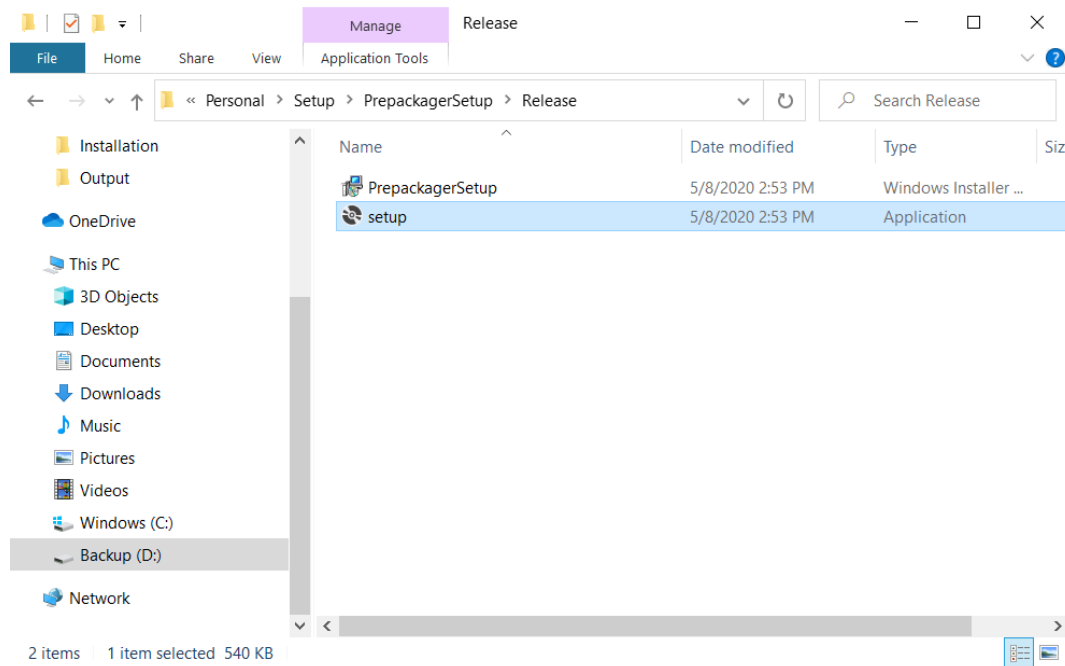
III. After extracting the zip file, you will get a new folder which contains the setup files.



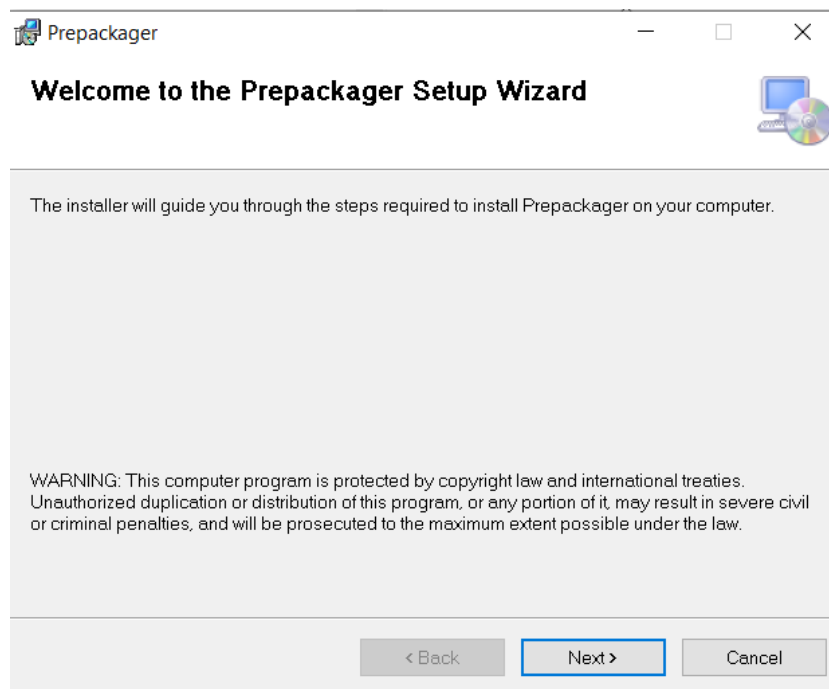
IV. Open the *PrepackagerSetup* folder. You will find two more folders inside it. Open the *Release* folder.



V. You will see two files inside the *Release* folder. Double-click on *setup* file to start the installation process.

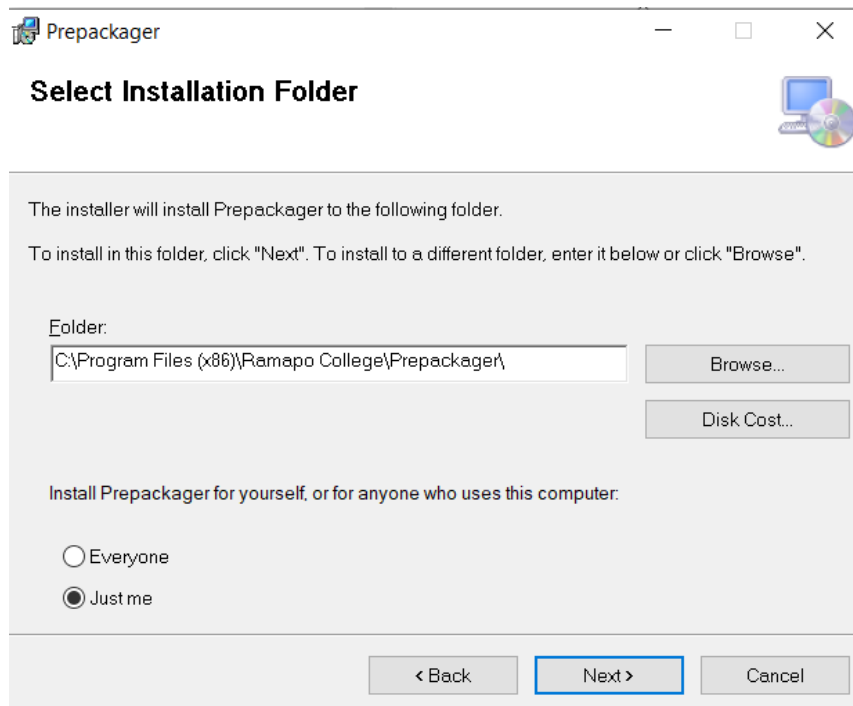


VI. In the first pop-up window, click on Next button.

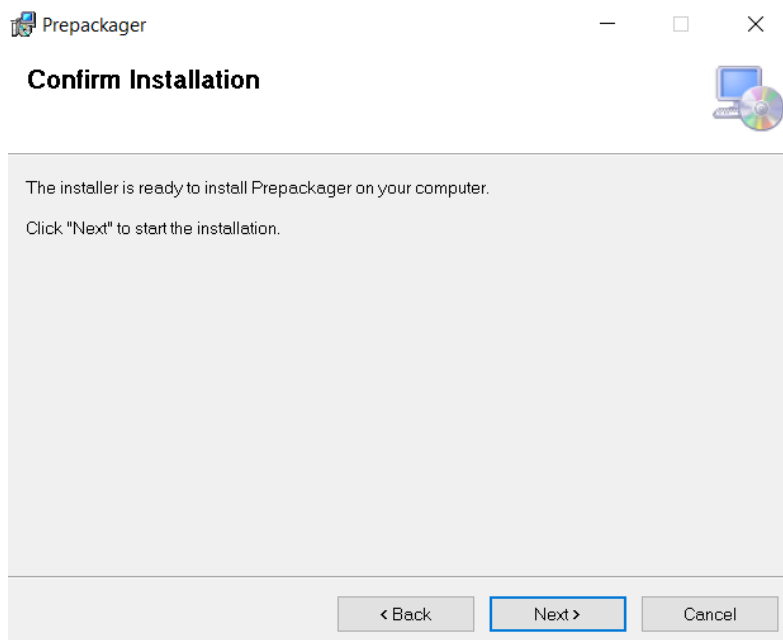


VII. Select the destination folder (where you want the software to be installed). Click on Next button.

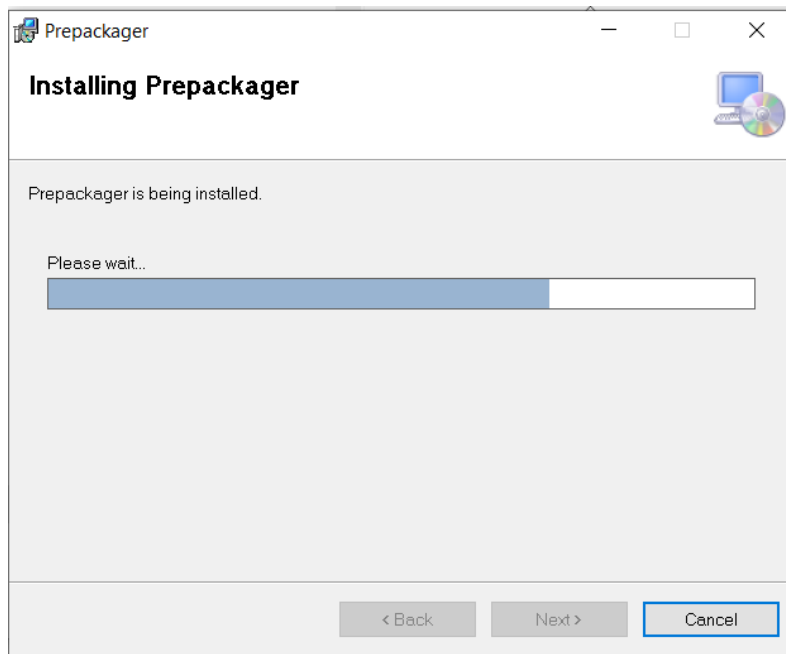




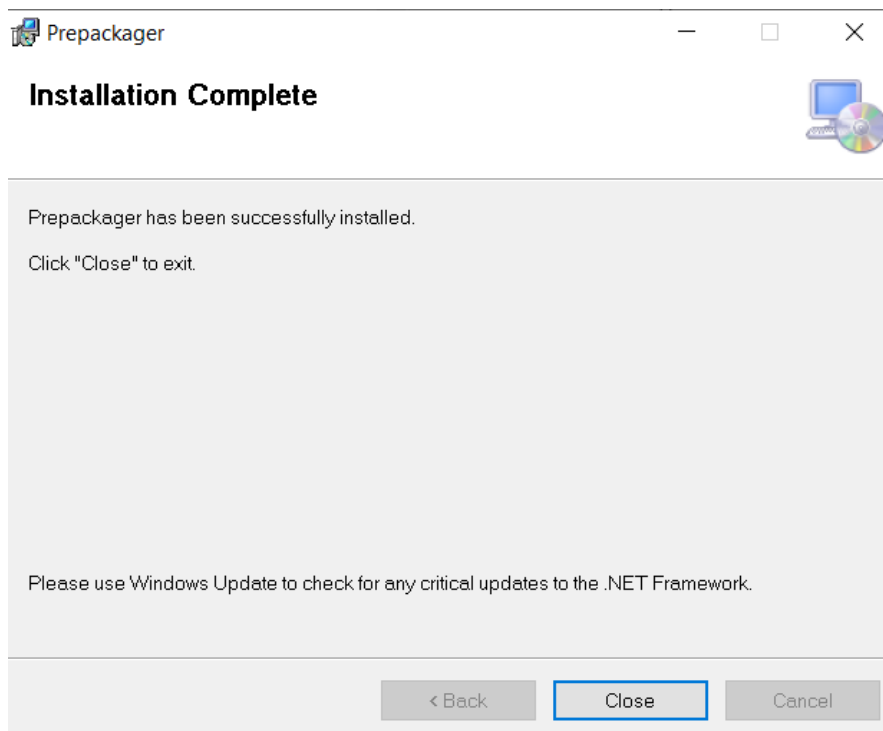
VIII. In the next window, click on Next button.



IX. Wait for the installation to complete.



- X. Finally, your installation is complete. You will find the installed files in your destination folder. A shortcut of the executable file will be added to your desktop.



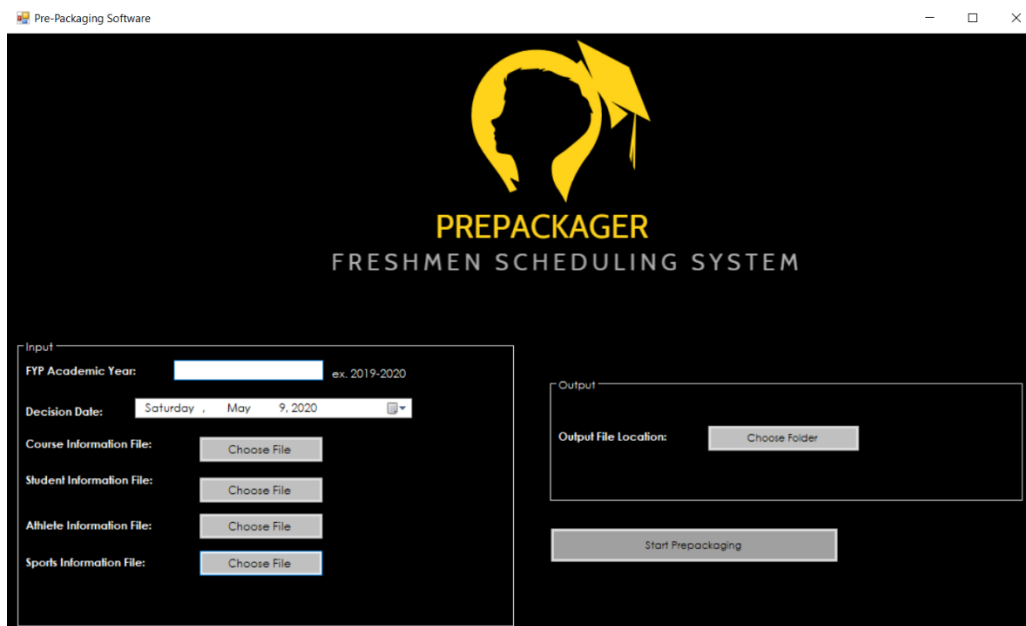
The installation process is now complete. For more instructions on using the actual software, please refer to [User Manual](#).

-----End of Installation Instructions-----

### 3 USER MANUAL

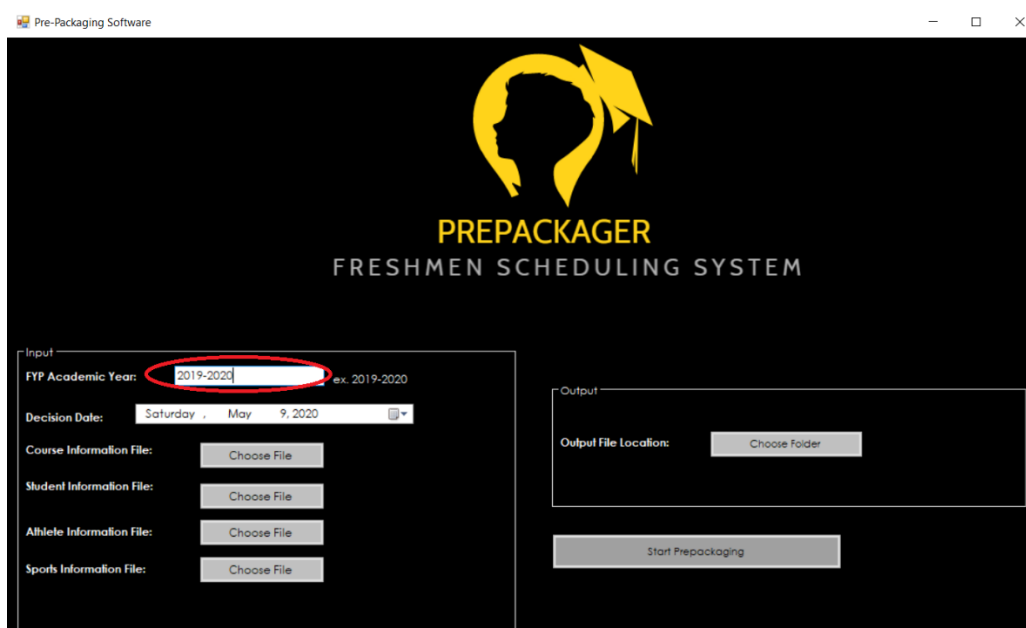
This document explains the usage instruction of the prepackaging software. It explains all the requirements, inputs and output of the prepackaging software.

- I. Once you run the executable file, you will be greeted with the following screen.



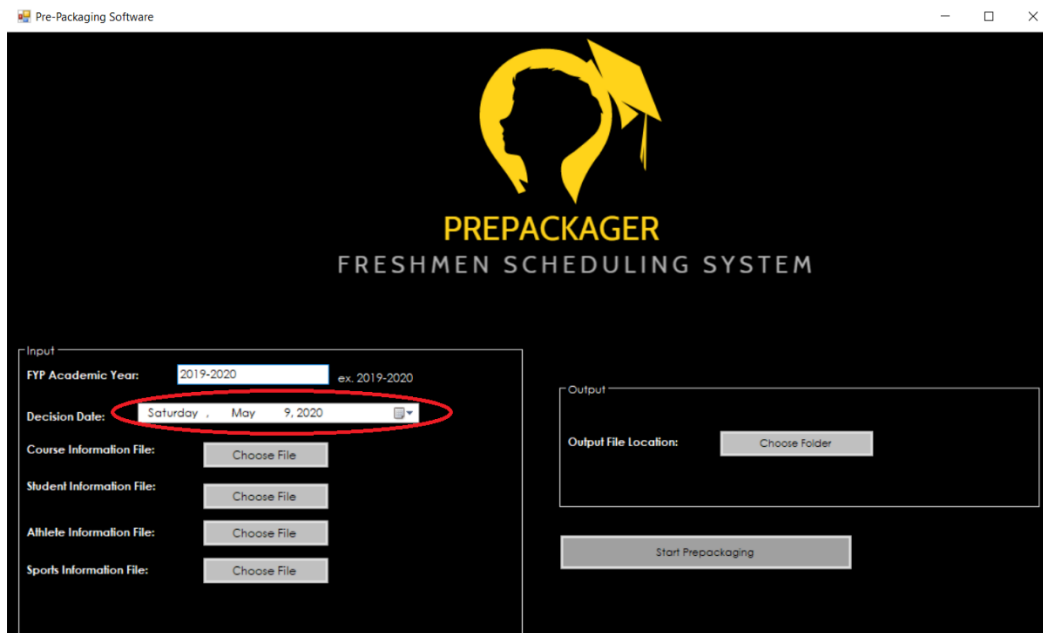
The screenshot shows the main interface of the PREPACKAGER FRESHMEN SCHEDULING SYSTEM. The window title is "Pre-Packaging Software". The interface features a logo at the top center consisting of a yellow silhouette of a person's head with a graduation cap. Below the logo, the text "PREPACKAGER" is displayed in yellow, and "FRESHMEN SCHEDULING SYSTEM" is displayed in white. The interface is divided into two main sections: "Input" on the left and "Output" on the right. The "Input" section contains the following fields and buttons: "FYP Academic Year:" with a text input field containing "2019-2020" and a placeholder "ex. 2019-2020"; "Decision Date:" with a date picker showing "Saturday, May 9, 2020"; "Course Information File:" with a "Choose File" button; "Student Information File:" with a "Choose File" button; "Athlete Information File:" with a "Choose File" button; and "Sports Information File:" with a "Choose File" button. The "Output" section contains an "Output File Location:" label with a "Choose Folder" button. At the bottom center, there is a "Start Prepackaging" button.

- II. You can see that there are two columns for specific user input. In the input column, the item is the FYP Academic Year. This is the academic year related to the Four-Year Plans.

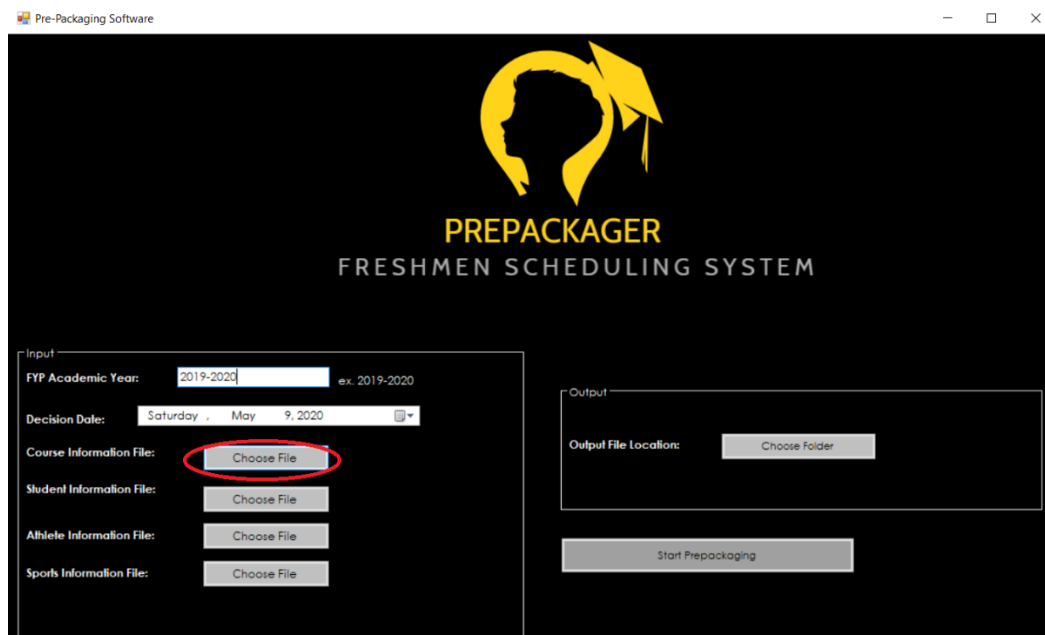


This screenshot is identical to the one above, showing the main interface of the PREPACKAGER FRESHMEN SCHEDULING SYSTEM. The "FYP Academic Year:" text input field, which contains "2019-2020", is highlighted with a red circle to draw attention to it.

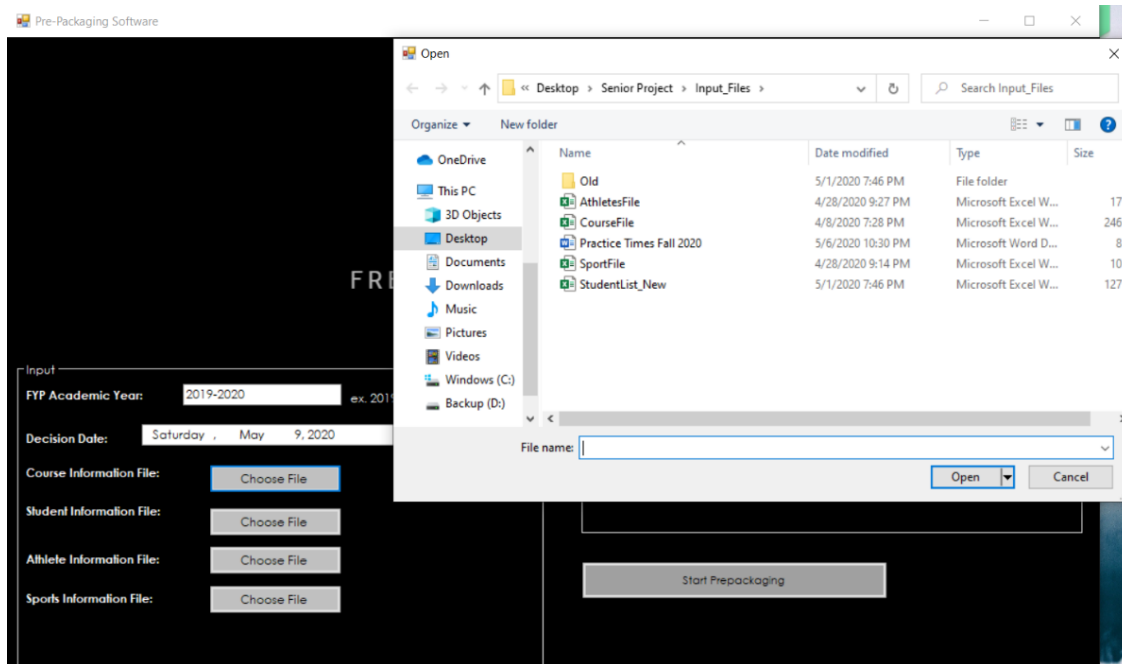
- III. The next element, Decision Date is the date of enrollment decision based on which the students are going to be filtered.



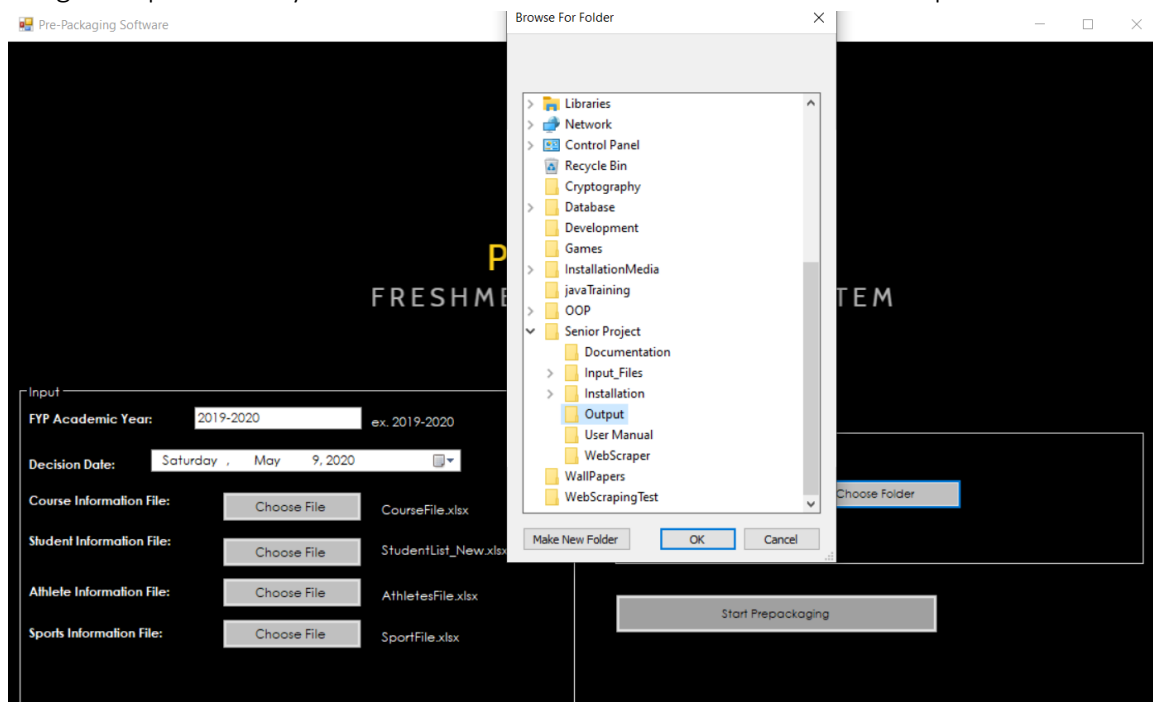
IV. For course information file, click on the choose file button.



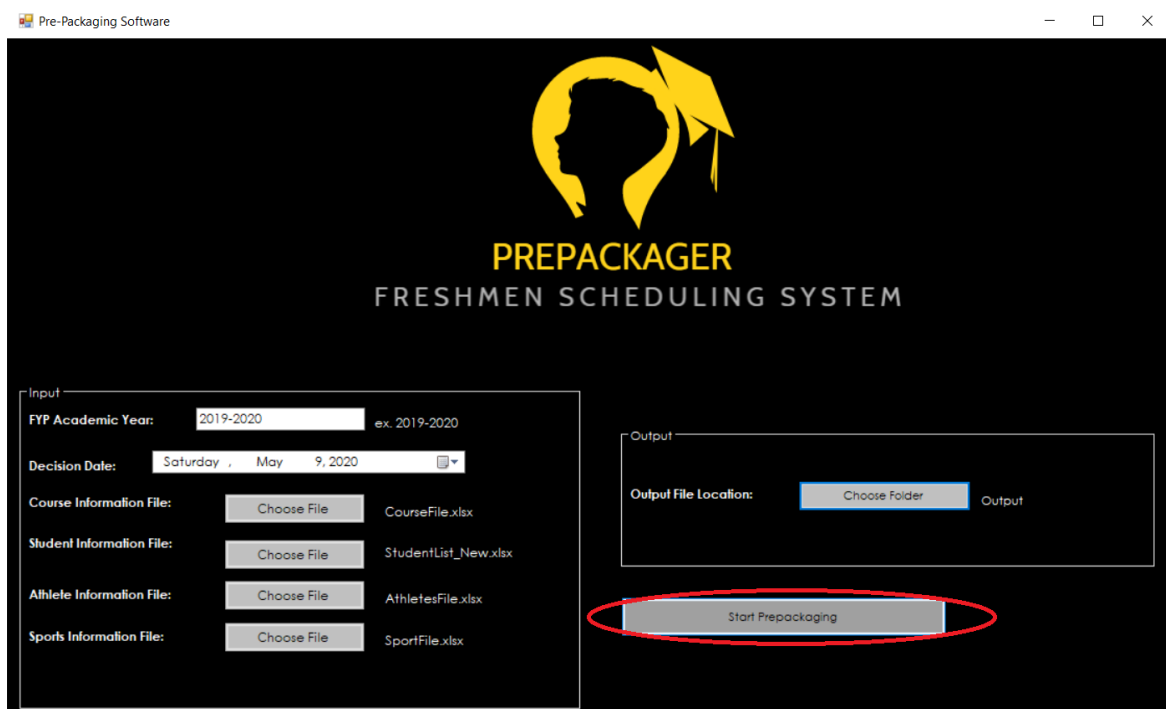
V. You will be prompted with an open file dialog box from which you will be able to choose the respective file. Choose respective files for Student Information File, Athlete information File and Sports Information File similarly.



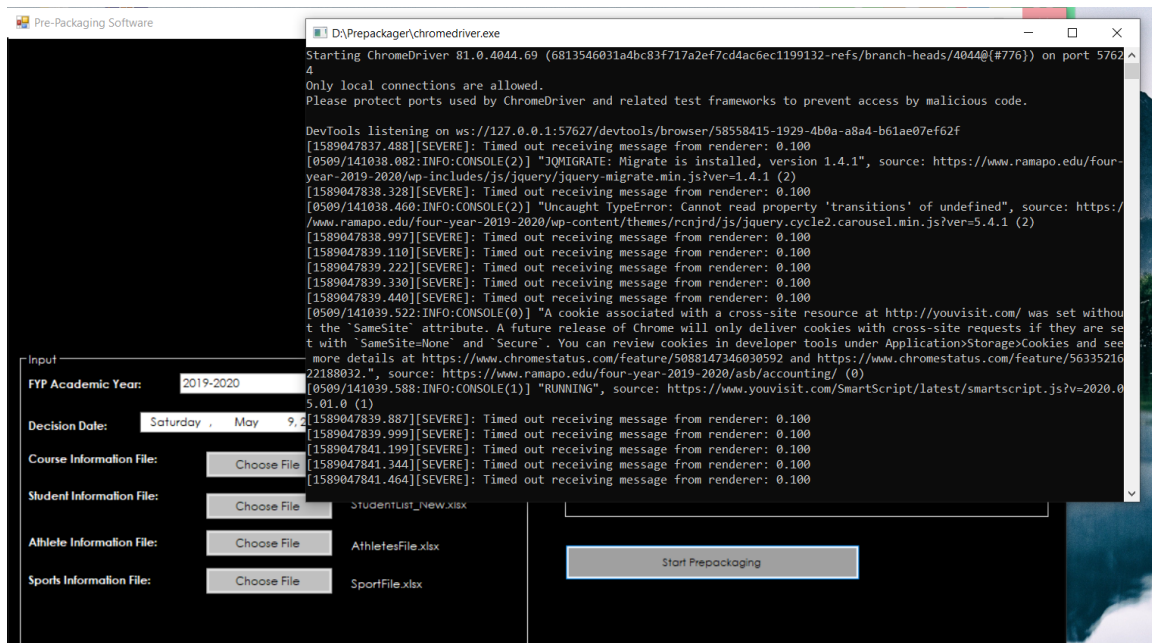
- VI. In the output column, choose the output folder by clicking on the choose folder button. An open folder dialog will open where you will be able to choose the location for the output file.



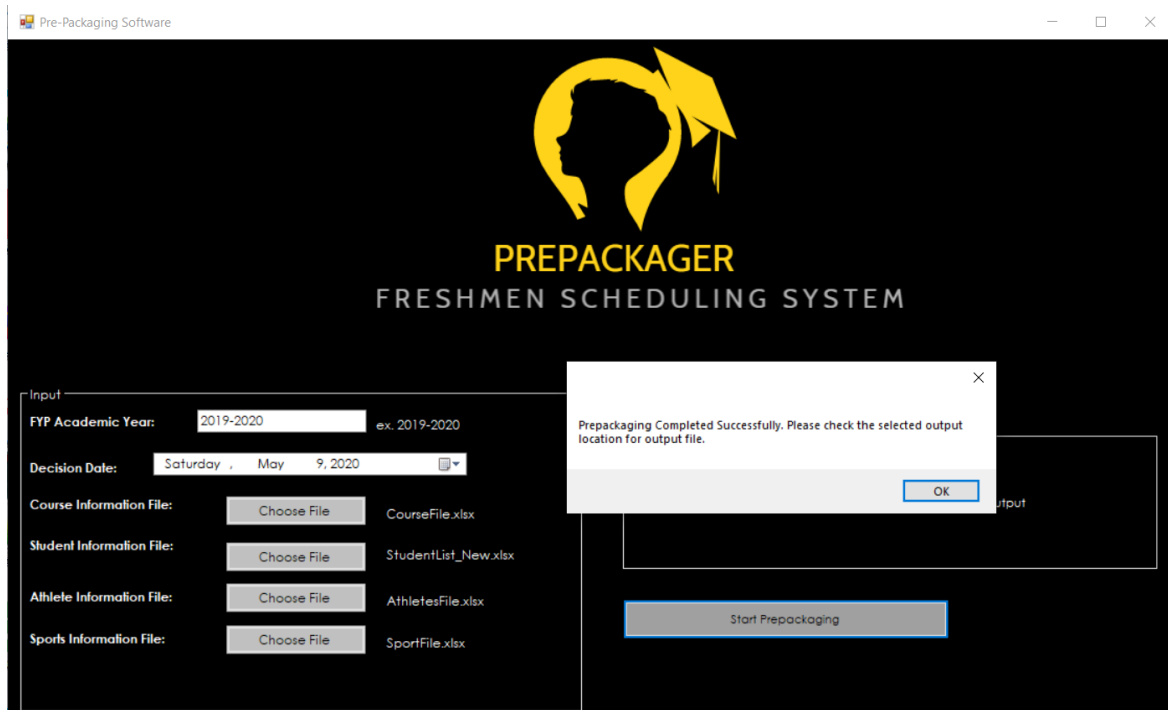
VII. After all the necessary input is complete, click on the Start Prepackaging button.



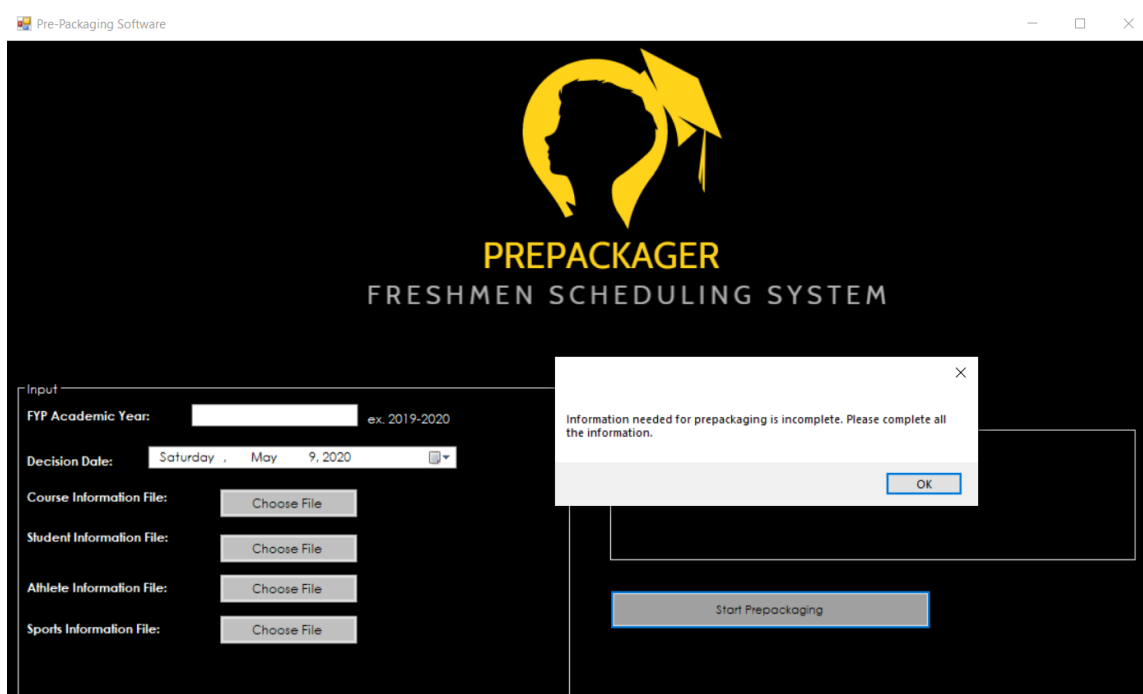
VIII. If the Software is being run for the first time in a new academic year, it will parse the Four-year plans from the webpages. This process will take about 5-6 minutes. The following window will appear if the web-scraping starts.



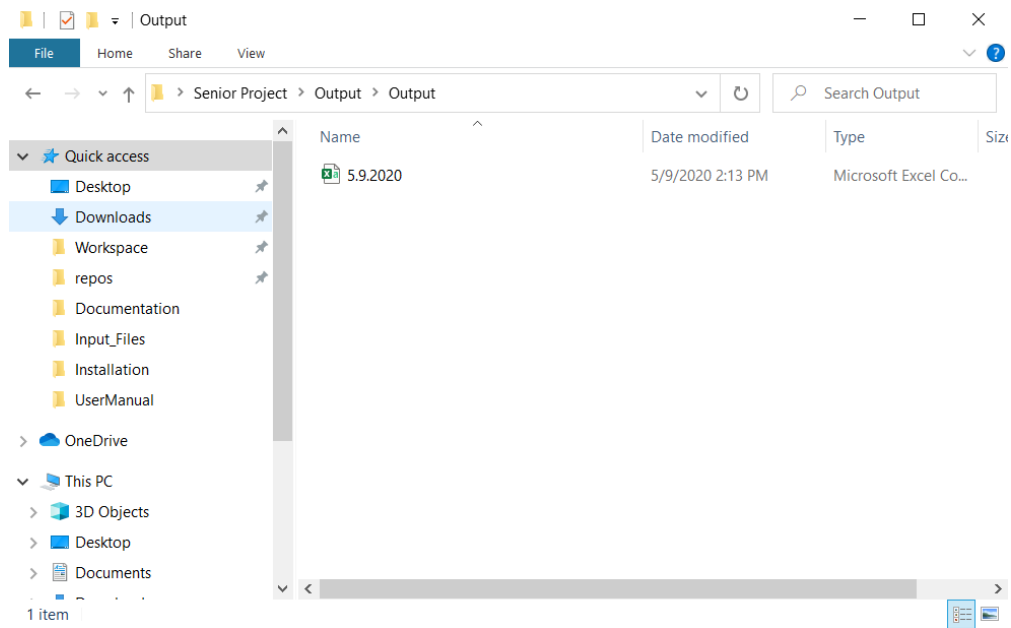
IX. Otherwise, the program will take a few minutes to complete. The following message box will appear after the successful completion of prepackaging process.



- X. If all the inputs are not provided, the following message box will appear. You will need to make sure all input elements are completed and correct before your start prepackaging.



- XI. Once the Prepackaging process is successfully completed, the output file will be located in the output destination that you chose. Notice the name of the output file is the enrollment decision date.



- XII. Finally, the output file will be in a .csv file format. You might want to save the .csv output file as an excel workbook.

----- End of User Manual -----



## 4 DESIGN DOCUMENTATION

---

### 4.1 DESIGN SUMMARY

The design of the software was heavily depended on the functional requirements and the target audience of the software. The audience/users of the software are the academic advisors at Ramapo College who presumably don't have much expertise in programming. Therefore, it was very important to provide the users with a simple graphical user interface rather than a command line interface.

In terms of functional requirements, one of the very important elements of the scheduling process were the four-year plans for each major, which needed to be read from Ramapo College's Four-Year Plan's webpages. A simple lightweight package called HTML Agility Pack along with selenium driver have been used to parse Four-year plan components from the web. In essence, web-scraping was one of the most challenging and critical aspects of the project.

Similarly, reading excel input files was another challenging element of the project. Although there were many different libraries and packages for C# that provided the functionality to read from excel files, choosing an efficient library was important. There were some libraries which were not very efficient in reading large excel files. *ExcelDataReader* was a lightweight and easy to use library which accomplished the task for this project. Moreover, developing an appropriate framework to improve readability, writability and maintainability of the source code was a challenging design decision.

Furthermore, designing the scheduler was one of the most difficult tasks in designing the software. The scheduler needed to consider a multitude of caveats, exceptions and criteria for scheduling students. Implementing such business logic was very tricky and required frequent consultation with the users (academic advisors). A lot of the scheduling algorithms and time conflict finding algorithms had to be developed from scratch keeping in mind the users' requirements as well as efficiency of the scheduling process.

Finally, all of these elements were carefully integrated into classes keeping in mind the principle of Object-Oriented Programming. As the software is designed to provide its functionality for many years, one of the key objectives while writing the code has been to focus on better readability and better maintainability. A lot of the functional requirements of the software are subject to change over the years. Therefore, the source code has designed to be easily modifiable and adaptive.

Additionally, an installer for the software has also been provided for better abstraction and better shareability of the software. The software can be installed in any computer running Windows operating system. On a client machine, only the binary (executable) will be installed so they don't have direct access to the program's source code. Overall, the design of the software has been hugely influenced by the ability to change, modify and maintain the software in future years. All the components are independent and later combined through a specific interface. This ultimately gives the maintainer of the source code flexibility in terms of changing independent components without having to modify other components.

## 4.2 KEY COMPONENTS/ASPECTS

The following section describes some of the key aspects of the Prepackager Software, their importance, and the issues that arose while tackling each aspect.

### 4.2.1 Graphical User Interface (GUI)

The GUI is one of the most significant aspects of the project as it serves as the gateway between the users and the software. I chose Windows Form Application for the GUI component of the software because of its simplicity and ease of use. Within the form app, I needed some key information (inputs) from the users like the academic year, date, input files, etc. Since the output was to be produced into a CSV file, the design of the form app is not event-driven. The backend processing is automatically triggered once the user inputs all the necessary information and clicks the button to start prepackaging process. Therefore, the GUI design of the application is very simple as most of the complex processes are carried out in the backend.

### 4.2.2 Web Scraping Four-Year Plans

Web scraping Four-Year Plans was a very difficult task as it presented a unique set of problems. HTML Agility Pack, a package for C# which helps to parse the contents of a webpage, has been used in order to extract relevant information from the Ramapo College's Four-Year Plan webpages. One of the problems that I faced when designing the web scraper was to understand the structure of the webpages (integration of HTML elements and contents) and to identify inconsistencies in the webpages. In case of any inconsistencies, I had to either formulate a method (while coding) to avoid them or communicate with Center of Student Success to rectify them in the webpages.

Similarly, some of the elements in the HTML source code were hidden and would only appear in the source code if a certain HTML element was clicked. Therefore, I had to use another powerful package called Selenium WebDriver, which is mostly used for testing Web Applications. Using Selenium, all the clickable HTML elements could be clicked before obtaining the pages' source code so that all the hidden elements would appear in the source code, and ultimately could be parsed.

### 4.2.3 I/O Manager

Handling input and output operations is another very important part of the software. The software requires the data stored in several excel files as input in order to process them and prepackage students. Therefore, reading excel files was another big challenge of the project. ExcelDataReader, a lightweight package that reads excel files, has been used in order to read excel files. Considering the maintainability of the project, for each excel reader class I created a class that holds the column constants so that the source code is easily modifiable and maintainable. In terms of the output, the IOManager simply writes the prepackaged students into a CSV file.

### 4.2.4 Scheduler

Scheduler is the core element of the Prepackager, responsible for scheduling each student individually based on their attributes. However, there were several constraints and requirements that the scheduler

needed to abide by in order to package each student correctly. In order to prepackage athletes, athlete information and sports time information are needed so that athletes are not scheduled into classes that conflicts with their practice times.

Similarly, honors students needed to be prepackaged into honors classes only when applicable. Presidential scholars and Deans scholars receive priority (get scheduled first). Moreover, commuter need to be scheduled into classes that are in close proximity and their classes shouldn't be spread over more than four days a week. Additionally, there were several other caveats related to course selection which needed to be considered during the Prepackaging process. With all these caveats, it was still crucial to find an appropriate set of courses for each student that don't conflict with each other. Therefore, integrating all these decision points into the scheduling algorithm was a challenging process.

## 4.3 DESCRIPTION OF CLASSES

### 4.3.1 Athlete

This class represents an athlete read from the athlete excel file. The purpose of this class is to hold Ramapo ID and Sport for each athlete. We can match Ramapo ID of the athlete with Students in the StudentList. If we find a match, we update the student's sport attribute.

The following are the functions of the class and their purpose.

Function Name	Purpose
public Athlete (string a_ramapold, string a_sport)	Constructor of Athlete Object. Takes the two attributes as parameters and initializes them.

### 4.3.2 AthleteReader

The AthleteReader class that inherits from ExcelReader class and contains functionality for reading athletes information excel files. The purpose of this class is to parse the excel file, row by row, reading each column and creating respective Athlete objects.

The following are the functions of the class and their purpose.

Function Name	Purpose
public List<Athlete> ReadAthletesList(string a_filename)	This function will take the athletes information file name, parse the file, create Athlete objects for each row and return a list of athletes.

### 4.3.3 AthletesExcelConstants

This static class holds constants representing the column values for Athletes Information file. It holds an enum with values corresponding to the column values in the input excel file.

Any changes to the order of columns in the course excel file can be easily reflected in the source code by changing the values in the enum.

This class doesn't have any member functions and only holds a public enum.

#### 4.3.4 Course

This class represents a unique class offered at Ramapo College (Each instance has a unique CRN). This class holds critical information about the course that is required for prepackaging.

The following are the functions of the class and their purpose.

Function Name	Purpose
public string GetDaysString()	This function computes the day string related to the course. Since days are stored as a dictionary, we need this function to easily obtain the day string. Example of daysString: mr, mwr, tf. etc.

#### 4.3.5 CourseExcelConstants

This static class holds constants representing the column values for Course Information File. It holds an enum with values corresponding to the column values in the input excel file.

Any changes to the order of columns in the course excel file can be easily reflected in the source code by changing the values in the enum.

This class doesn't have any public member functions and only holds a public enum.

#### 4.3.6 CourseReader

The CourseReader class that inherits from ExcelReader class and contains functionality for reading course information excel files. The purpose of this class is to parse the excel file, row by row, reading each column and creating respective Course objects based on the excel file.

The following are the functions of the class and their purpose.

Function Name	Purpose
public List<Course> ReadCourseList(string a_filename)	This function will take the course information file name, parse the file, create Course objects for each row and return a list of courses.

#### 4.3.7 CreditInfo

This class represents the credit information related to a particular student. It holds all the credits related data for a particular student. It has been used to encapsulate similar attributes (Credits Data) for better readability.

This class doesn't have public member functions.

#### 4.3.8 ExcelReader

This is an abstract class that provides the functionality for parsing excel spreadsheets. It renders general functionalities for parsing excel spreadsheet using the Excel DataReader package.

This class doesn't have public member functions but rather contains protected member functions which will be inherited by its child classes.

#### 4.3.9 FYP\_Category

This class represents a category of courses. The category could be a representative of a single course. Ex. Critical Reading and Writing II or it could be a representative of multiple courses. Ex. Gen Ed. Social Science Inquiry.

It provides a generalized object to represent each category in the four-year plan. It contains the list of course elements, which are actual courses (Ex. MATH 101). For a single course (like. MATH 101), the list will have only one element.

For a category of courses (like Gen Ed. courses), the list will have multiple courses.

The following are the functions of the class and their purpose.

Function Name	Purpose
public FYP_Category()	This is the default constructor for FYP_Category class. Initializes the member variables to null.
public FYP_Category(string a_category, List<FYP_CourseElement> a_courseList)	Parametrized constructor for FYP_Category class.
public static FYP_Category operator+ (FYP_Category a_cat1, FYP_Category a_cat2)	Overloading + operator for FYP_Category class. Helps in combining the courses in two different categories.
public bool IsNull()	Determines if the object is a null object or not.
public bool IsGenEd()	Determines if the FYP_Category is a Gen Ed category or not.
public override string ToString()	Converts the object into a string for serialization purposes.
public bool IsFYS()	Determines if the FYP_Category is a FYS (First Year Seminar) category or not.

	This is important because we don't want to prepackage students into FYS classes. They have the option to pick their FYS classes themselves.
public bool IsGlobalAwareness()	<p>Determines if the FYP_Category is a Global Awareness category or not.</p> <p>This is important because we don't want to prepackage students into Global Awareness classes.</p> <p>This is because Global Awareness classes often require language testing which the student might not have necessarily tested into.</p>

#### 4.3.10 FYP\_CourseElement

This class represents an individual course element (Ex. CMPS 364, PHIL 101, MATH 101, etc.). The class holds the title (ex. MATH) and id (ex. 101) for each course element.

The FYP\_CourseElement object will be contained in the FYP\_Category and will ultimately be a part of the Major Class.

The following are the functions of the class and their purpose.

Function Name	Purpose
public FYP_CourseElement()	Default Constructor for FYP_CourseElement class. Sets all attributes to null.
public FYP_CourseElement(string a_title, string a_id)	Parametrized constructor for FYP_CourseElement class.
public override string ToString()	Converts the object into a string for serialization purposes.

#### 4.3.11 Interval

This class represents a time interval of a particular course/sport. It holds the startTime, endTime and days for a particular course/sport. This will be used to determine time conflicts by the scheduler.

The following are the functions of the class and their purpose.

Function Name	Purpose
public Interval()	Default Constructor for Interval class. Sets times to -1 and days to empty string.
public Interval(int a_startTime, int a_endTime, string a_days)	Parametrized constructor for Interval. Takes all the attributes as parameters and sets them.

#### 4.3.12 IOManager

The class handles all input and output operations related to prepackaging process. The purpose of this class is to get all the inputs from respective sources and create the output file based on the results of the scheduler.

The following are the functions of the class and their purpose.

Function Name	Purpose
public IOManager(string a_year, string a_courseFilePath, string a_studentFilePath, string a_outputDirPath, string a_athleteFilePath, string a_sportFilePath, DateTime a_decnDate)	Parametrized constructor for IOManager Class.
public List<Major> GetFYPMajors()	Obtains the list of Major class objects, which represent the four-year plan for each major in Ramapo.
public List<Course> GetCourses()	Obtains the list of all courses (classes) being offered at Ramapo for the particular semester. This will be obtained from the Course Information File.
public List<Student> GetStudents()	Obtains the list of all incoming freshmen students based on the student information excel file.
public List<Athlete> GetAthletes()	Obtains the list of all athletes. This will be obtained from the Athletes information excel file.
public List<Sport> GetSports()	Obtains the list of all the sport objects using SportsTimeReader. This will be obtained from the sports information file.

public void WriteStudentList(List<Student> a_prepackagedStudents)	Writes the output - prepackaged students and their details to a csv file.  The file is saved in the output folder which has been taken as a user input.
---	---

#### 4.3.13 Major

This class represents the Four-Year Plan for a single major. It represents the FYP for a major by holding the Course Categories and the Course elements recommended for the major.

The following are the functions of the class and their purpose.

Function Name	Purpose
public Major()	Default constructor of Major class. Sets all the member variables to null.
public Major(string a_name, string a_id, List<FYP_Category> a_courseCategories)	Parametrized constructor for Major class.
public override string ToString()	Converts the object into a string for serialization purposes.
public bool IsNull()	Determines if the major object is null or not. If both name and courseCategories are null, the major is considered to be null.
public List<FYP_Category> GetGenEds()	Obtain all the general education categories from the CourseCategories.

#### 4.3.14 Prepackager

This class represents the prepackager which provides an interface to initiate aspects of prepackaging process. It serves as the link between the front-end user interface and back-end business logic.

The following are the functions of the class and their purpose.

Function Name	Purpose
public Prepackager(string a_year, string a_courseFilePath, string a_studentFilePath, DateTime a_decisionDate, string a_outputDirPath, string a_athletesFilePath, string a_sportsFilePath)	Parametrized Constructor of Prepackager Class. Obtains attribute values as parameters and sets respective attributes.
public void StartPrepackaging()	Handles all elements of the prepackaging process.  Gets all necessary inputs from the IOManager class.



	<p>Schedules the students using Scheduler class.</p> <p>Writes the output to a csv file using IOManager class.</p>
--	--

#### 4.3.15 Scheduler

This class represents the scheduler which is responsible to schedule each student into appropriate classes based on their attributes.

This class contains the critical scheduling logic needed to consider the criteria and requirements for scheduling students.

This class schedules students based on the scheduling requirements outlined by the Center of Student Success.

The following are the functions of the class and their purpose.

Function Name	Purpose
public Scheduler(List<Course> a_courseList, List<Student> a_studentList, List<Major> a_majorList, DateTime a_decisionDate, List<Athlete> a_athleteList, List<Sport> a_sportList)	Parametrized constructor for scheduler object. Takes required attributes as parameters and sets those attributes.
public List<Student> StartScheduling()	This function starts the scheduling process, by scheduling qualified students based on priority characteristics and returns a list of prepackaged students.

#### 4.3.16 Sport

This class represents a sport read from the sports information file. It holds the sport name and the practice times as intervals.

The following are the functions of the class and their purpose.

Function Name	Purpose
public Sport()	Default constructor of the sport class. Sets name to empty string and intervals to null.
public Sport(string a_name, List<Interval> a_intervalList)	Parametrized constructor for Sport class. It obtains attribute values as parameters and sets each attribute respectively.

public bool IsNull()	Determines if the Sport is a null sport (no value). If the name is empty string and intervals is null, it is a null sport.
----------------------	---

#### 4.3.17 SportsTimeExcelConstants

This static class holds constants representing the column values for Sports Information File. It holds an enum with values corresponding to the column values in the input excel file.

Any changes to the order of columns in the course excel file can be easily reflected in the source code by changing the values in the enum.

This class doesn't have any public member functions and only contains a public enum.

#### 4.3.18 SportsTimeReader

This class inherits from ExcelReader class and contains functionality for reading sports information excel files. The purpose of the class is to parse the excel file, row by row, reading each column and creating respective Sport objects.

The following are the functions of the class and their purpose.

Function Name	Purpose
public List<Sport> GetSportsList(string a_filename)	This function will take the sports information file name, parse the file, create sports objects for each row and return a list of sports.

#### 4.3.19 Student

This class represents each incoming freshman student who needs to be prepackaged. It holds all the information about the student which are required for prepackaging purposes.

The following are the functions of the class and their purpose.

Function Name	Purpose
public bool IsPrepackaged()	This function determines if a student has been previously prepackaged and registered into courses.
public bool HasRequiredTests()	This function determines if a student has taken the tests that are required.
public List<string> GetCourseCredits()	This function returns a list of strings, each of which represent a course credit that they have acquired.

<code>public bool IsPresidentialScholar()</code>	This function determines is a student is presidential scholar or not.
<code>public bool IsDeanScholar()</code>	This function determines is a student is dean scholar or not.
<code>public override string ToString()</code>	This function returns a string, which represents the serialized student object with partial details. The details include the prepackaged courses and their details (Name, CRN, Day/Time).
<code>public static string GetHeader()</code>	This function contains the header string for the output csv file. The header values must each match with the respective student data included in the ToString() method.

#### 4.3.20 StudentExcelConstants

This static class holds constants representing the column values for Student Information File. It holds an enum with values corresponding to the column values in the input excel file. Any changes to the order of columns in the Student excel file can be easily reflected in the source code by changing the values in the enum.

This class doesn't have any public member functions and only contains a public enum.

#### 4.3.21 StudentReader

This class inherits from ExcelReader class and contains functionality for reading student information from excel file. The purpose of the class is to parse the excel file, row by row, reading each column and creating respective Student object based on the excel file.

The following are the functions of the class and their purpose.

Function Name	Purpose
<code>public List&lt;Student&gt; ReadStudentList(string a_filename)</code>	This function will take the student information file name, parse the file, create Student object for each row and return a list of Students.

#### 4.3.22 TestingInfo

This class represents the test information related to a particular student. It holds all the test related data for a particular student. It is used to encapsulate similar attributes (Test Data) for better readability.

This class doesn't contain any public member functions.

#### 4.3.23 Webscraper

This class is responsible for parsing Four-Year Plans from the Ramapo four-year plan webpage. Its main objective is to read four-year-plan from the web for each major in Ramapo College and create a useable format of the FYP for each major.

The following are the functions of the class and their purpose.

Function Name	Purpose
public Webscraper()	Default Constructor for Webscraper Class. Sets m_year to 2019-2020 by default. The url is based on the year.
public Webscraper(string a_year)	Parametrized constructor for Webscraper class. Sets m_year based on the parameter. Sets url based on the academic year.
public List<Major> GetFourYearPlans()	Obtain the FYP representation for each major offered at Ramapo from the four-year plan webpage.

## 5 TEST PLAN AND TEST RESULTS

---

### 5.1 UNIT TESTING PLAN

Initially, in the design phase, the software was broken down into primary components, each of which were developed independently and later integrated with other functionalities. Components like Webscraper, Excel File Reader, and GUI design were developed independently. Therefore, these components could be tested independently.

### 5.2 UNIT TESTING RESULTS

Before working on the scheduler and integrating all the major functionalities together, unit testing was carried out to test each component. For each component, the framework to test and verify the correctness was developed. For example, the functionality of Web scraper was continuously tested and modified until the correct format of Four-Year Plans was obtained.

Similarly, the functionalities for reading excel files were tested individually for each type of excel file that were required for this project. Once all the independent components started working correctly, they were integrated together to complete the remaining part of the project.

### 5.3 INTEGRATION TESTING PLAN

The design of the scheduler required all other components to be working correctly. Therefore, after unit testing of all other components, these components were integrated together to test their interoperability and compatibility. Any inconsistencies and errors that were found during the integration testing were corrected. Finally, after the completion of all essential components of the software, the results of the software were tested against the expected outcomes specified by Center of Student Success.

### 5.4 INTEGRATION TESTING RESULTS

The Center for Student Success provided all the necessary input data (Student Lists, Course Lists, Athletes Information and Sports Practice times). Thus, in order to test all the functionalities of the software, I had all the required data although the data was from the last academic year. Similarly, the four-year plans extracted from the webpages were also from 2019-2020 academic year instead of 2020-2021 academic year. Finally, the working of the software and its components were compared against the results and requirements expected by the academic advisors. Any inconsistencies were solved according to the requirement document designed in consultation with the Center for Student Success.

As the latest data for the next academic year started coming in, the results (excel file) were tested in respect to the requirement document. In addition, the correctness and consistency of the results were verified through an in-depth discussion with the academic advisors. Finally, the design of the software was consolidated after all the academic advisors agreed and understood the output produced by the software.

## 6 SUMMARY AND CONCLUSIONS

---

The process of planning, designing, developing, testing and deploying the software in a real-life work environment has been a challenging but fulfilling process. Apart from software design skills required to write the software, this project demanded a lot of other intrapersonal and technical skills.

As the project was being developed for a real-life work environment, I had to communicate with the Center for Student Success and the academic advisors on a weekly basis to discuss the nitty-gritty of the project. It was important to develop the software with respect to the requirements of the users and the business process. Without having enough domain knowledge of the prepackaging process, I had to fully understand the requirements, processes and caveats involved in the process so that the business logic could be converted into appropriate software design and code.

Moreover, designing a project which will be used for many more years with many different input parameters and dynamic requirements meant that the design of the software needed to be flexible, easily comprehensible and modifiable. Thus, the entire process of planning the design of the software required tremendous amount of critical thinking skills. As mentioned earlier, a lot of the components used in the software have been developed and tested independently making these components much more portable and flexible. The changes that might be needed in the future due to the change in functional requirements will mostly likely very few changes to the overall source code.

Similarly, each aspect of the project presented a multitude of previously unseen or unprecedented problems which needed to be tackled pragmatically. Oftentimes, one little but unique problem halted the development of the software for weeks. However, each problem provided a lot of insight about the concepts being used and significantly improved my problem-solving skills. For instance, I have learnt a lot about methods and tricks of parsing webpages because of my involvement in this project. Similarly, reading excel files and processing the information read from those excel files was a new aspect of programming for me. In terms of automating the actual scheduling itself, developing your own unique algorithms to schedule students and making sure the algorithms were efficient in itself was a big achievement for me personally. Overall, the whole process certainly yielded a whole lot of unique problems and challenges. But solving those problems and achieving the development objectives that I had set personally has definitely made me a better problem solver and researcher.

To conclude, the Prepackaging Software is a tailor-made software targeted to automate a specific process for a specific set of users. The software is expected to save almost a month of the time taken by academic advisors when the process is carried out manually. Therefore, it has a lot of significant in terms of being instantly applicable to a real-life business scenario. Having said that, like any other software, the prepackaging software will have to be properly maintained over the years to make sure that its core functionality always remains consistent and effective. After the completion of the project, Ramapo College's ITS Department will be handed over the source code of the software and they will be responsible for the maintenance and modification of the software to comply with the changes in the software's requirements.

## 7 BIBLIOGRAPHY

---

- I. Combinations from n arrays picking one element from each array. (2019, April 3). Retrieved from <https://www.geeksforgeeks.org/combinations-from-n-arrays-picking-one-element-from-each-array/>
- II. Given n appointments, find all conflicting appointments. (2020, February 14). Retrieved from <https://www.geeksforgeeks.org/given-n-appointments-find-conflicting-appointments/>
- III. Html Agility Pack HAP - Documentations. (n.d.). Retrieved from <https://html-agility-pack.net/documentation>