Faculty of Computer Science and Business Information Systems

Computer Science (Bachelor of Engineering)

Exam Quality Control

5100240, Programming Project, 2023-2

Technical University of Applied Sciences Würzburg-Schweinfurt - Faculty of Computer Science and Business Information Systems, Sanderheinrichsleitenweg 20 97074 Würzburg.

KAAN ÖZER
Computer Science
9123043

Kemal Eren Öztürk
Computer Science
9123044

Müberra Şeyma USLU
Computer Science
9123059

Illia Rohalskyi
Computer Science
9123047

8. May 2023

Contents

1 Introduction

1.1 Context of the Work

The "Exam Quality Control" program aims to address the challenges faced in managing and evaluating exam schedules in educational institutions and organizations. Creating effective and fair exam schedules while following various rules and constraints can be difficult. Manual evaluation of exam plans is time-consuming and prone to mistakes. Therefore, it is essential to develop an automated software solution that can examine exam schedules based on predefined rules.

1.2 Motivation for This Work

The motivation behind developing the "Exam Quality Control" program is to evaluate the current exam schedule with utmost accuracy and efficiency. The main objective is to identify any conflicts between the evaluated exam plan and the specified rules and assist in creating the subsequent exam schedule that adheres as closely as possible to these rules. By automating this process, our institution can simplify their scheduling procedures, reduce conflicts, and improve the overall quality of exams. This software gives administrators and schedulers a reliable tool to follow regulations, allocate resources effectively, and provide the best possible exam experience for both students and examiners.

1.3 Objective for This Work

The primary objective of the "Exam Quality Control" program is to provide a comprehensive and automated solution for evaluating exam schedules based on predefined rules. The software will consider multiple rules during the evaluation process, including:

Big Exams Early: Big exams should be held early.

One Day Gap: There should be one day gap between each two exams.

One Exam Per Day: There should be only one exam per day for each student.

Special Dates: Examiners shouldn't come in specific dates.

Special Professors: The distance between the two exam shouldn't be so long if the examiner is same. Additionally, the number of exams that assigned to the examiners should be in a balance.

Room Capacity: The number of students and the capacity of the rooms should be in a balance.

Room Distances: If one exam is conducted in more than one room, the distance between them should be minimal.

By evaluating exam schedules based on these rules and potentially additional ones, the software will provide scores for each rule, enabling administrators to identify areas for improvement and make data-driven decisions to optimize the overall exam plan. The objective is to simplify the scheduling process, enhance the quality of exam plans, and improve the overall experience for both students and examiners.

2 Technical Foundations

2.1 Regression basics:

In the context of this work, *linear regression* is a technique that aims to find a line that best fits a given set of data points. It does so by minimizing the distance between the line and the data points. The formula of the line is $y = mx + b$, which represents a straight line.

Polynomial regression refers to a line that has N number of polynomials in its formula. The formula for polynomial regression is given by $y = m_n x^n + m_{n-1} x^{n-1} + \ldots + m_1 x + b$, where m is the slope and b is the intercept. Both the parameters m and b are approximated by regression. The degree N of the polynomial is determined heuristically. Using polynomials enables the line to be non-linear and have some curvature.

3 Evaluation of Project Results

3.1 Description of the current status:

The application is currently in an operational state, allowing users to input their exam plans for evaluation. The system utilizes algorithms and rule-based systems to process the exam plans based on the predefined criteria. It assigns weights to each criterion to reflect their importance in the overall assessment.

The rules are as follows:

- Big exams have to be early.
- Students mustn't have two exams on the same day.
- Students should have one day gap between exams.
- Rooms have to be neither too big nor too small for the exam.
- If two rooms are assigned for the exam, they have to be as close as possible.
- The professor is not available on some dates.
- The professor wants to come on a minimal amount of days.

The current version of the application provides a score indicating the overall quality of the exam plan. Additionally, it generates an HTML report file that includes visualizations, conflict dataframes, and scores for each individual criterion. This report serves as a valuable tool to help professors spot potential problems and gain insights into the strengths and weaknesses of the exam plan.

3.2 Functional requirements:

The application should assess the exam plans based on predefined criteria, evaluating each criterion individually. The perfect exam plan would satisfy both, professors and students. Thus, we should aim to provide a comprehensive solution that would take into consideration both perspectives.

The application should calculate an overall score for each exam plan, indicating its quality. The scoring mechanism will be based on weighted averages, taking into account the relative importance of each criterion.

The application should provide detailed explanations for the assigned score, clearly stating the reasoning and factors that contributed to it. This feature helps professors understand the strengths and weaknesses of the exam plan and provides transparency in the evaluation process.

The application should generate an HTML report file, presenting visualizations, conflict dataframes, and scores for each individual criterion. The report should be easily accessible and designed in a clear and intuitive manner, allowing professors to review the assessment results effectively.

The application should have conflict detection capabilities, identifying any conflicts within the exam plan. It should highlight inconsistencies or contradictions between criteria in the report, enabling professors to address and resolve these conflicts during the exam planning process.

By fulfilling these functional requirements, the solution aims to comprehensively evaluate exam plans, calculate scores, provide explainability, generate informative HTML reports, and assist professors in identifying and resolving conflicts to enhance the quality of their assessments.

3.3 Non-functional requirements

The application should be reliable, ensuring accurate and consistent evaluation results. It should be able to handle errors and exceptions gracefully, with minimal impact on the overall functionality. On top of that, the application should be designed and developed with maintainability in mind, allowing for easy updates, bug fixes, and future enhancements

4 Description of the Solution

4.1 Description of the software architecture

Our software consists of three layers which are Input, Process, and Output layers. Data layer.

The input layer is responsible for the preprocessing of examination plan files, enabling their utilization within the software.

The process layer is responsible for analyzing the exam plan data to score, identifying conflicts, and visualizing data based on rules such as ensuring a one-day gap between exams.

The output data is responsible for representing the scores, conflicts, and charts of every rule in a HTML file.

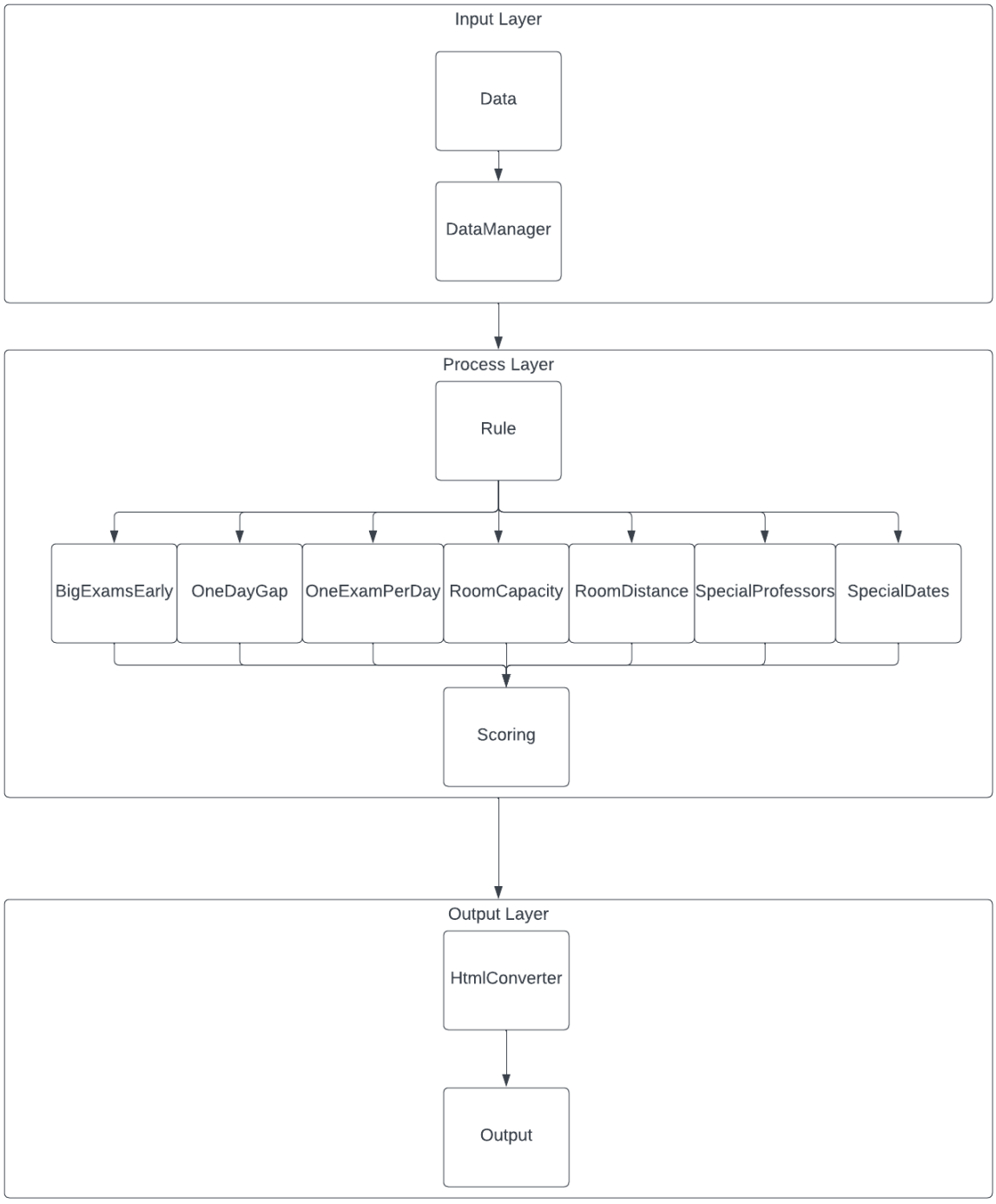

Figure 1: Architecture Design

4.2 Overview and interaction of the components

4.2.1 Main Class

- This class is responsible from the execution of the program and saving the various html results based on the demand.

4.2.2 Scoring Class

- Scoring class interacts with the following rule classes: BigExamsEarly, OneExamPerDay, RoomCapacity, RoomDistance, SpecialProfessors, OneDayGap, SpecialDates.
- This class is a wrapper for all of the rule classes. It is made to make it easy to access each of the components under one class.

4.2.3 Data Class

- The Data class is responsible for importing and organizing the input data from various files.
- It can load different types of data such as CSV, JSON, and Excel. The following are the files that the Data class interacts with: `FIW_Exams_2022ws.xlsx`, `Pruefungsanmeldungen_anonmous.csv` `room_distance_matrix.xlsx`, `capacity.json`, `special_dates.csv`, `specific_professors.xlsx`.
- It also splits and extracts relevant information from the loaded data and stores them as attributes and methods for further processing.
- All of the rule classes interact with Data. It is a prerequisite for them to work.

4.2.4 Rule Class

Rule is the superclass of all scoring classes (such as BigExamsEarly, OneExamPerDay, etc.).

The class has the following attributes:

- score.
- conflicts.
- plot array.

It has a method called compute() which returns the score, conflicts, and plot array.

4.2.5 Data Manager Class

DataManager is a class to instantiate Data objects. The class provides methods to create and retrieve instances of the Data class. It ensures that only a single instance of the Data class is created and reused throughout the program.

4.2.6 BigExamsEarly Class

- Big exams should be held early in the exam period, and this class calculates the score and evaluates the exam plan based on the number of big exams and their scheduling time.

4.2.7 OneExamPerDay Class

- Verifies whether each student has only one exam scheduled per day.
- Gives a penalty score while calculating the score of the exam plan based on the number of students who has multiple exams in a day.

4.2.8 OneDayGap Class

- This class calculates the score based on the absence of a one-day gap between exams for the same student.
- basically identifies cases where there is no one-day gap between exams for the same student and calculates a general score for this rule by giving penalty based on the number of conflicts and also draws a plot showing the relationship between the number of conflicts and the score.

4.2.9 RoomCapacity Class

- Ensures that the assigned rooms can accommodate the number of students registered for each exam.
- Computes the score by comparing the number of students with the room capacities for the each exam.

4.2.10 RoomDistance Class

- It search the rooms for each exam and if there is more than one room. It calculates the sub-scores of the distance between the relevant rooms.
- This class has only score as its output.

4.2.11 SpecialDates Class

- This class is responsible for calculating the score for examiners, which are not able to come for the exam on a specific day.
- Checks for conflicts between examiners' names and special dates in the `special_dates_df`.
- Import data class to reach `special_dates_df` .
- if there is a match, it adds the conflict information to the 'conflicts' list and makes a binary decision.
- It returns a percentage score based on whether there is a conflict or not.

4.2.12 SpecialProfessors Class

- This class calculates special professors score based on the number of exams and the distance between those exam days that same professors supervise.

4.2.13 Output Class

- Handles the generation and saving of output files.
- Provides methods for saving the results as HTML files. These methods offers two different option either you can get a single result of one rule class or a summary list that includes all results.
- Utilizes the HtmlConverter class to convert data into HTML format and print the output.

4.2.14 HtmlConverter Class

- Converts the processed data into HTML format for output representation.
- Provides methods for creating HTML pages, and printing the HTML output.

4.3 Quality Assurance

4.3.1 Introduction

This document presents the Quality Assurance strategy that has been employed for the evaluation of our existing examination scheduling system. We utilized Python's unittest framework to construct and perform unit tests for each rule class in the system.

Each rule class is thoroughly tested under both the best-case and worst-case scenarios. These scenarios are represented using a combination of mock data, created within the Test methods for each test case, and artificial data derived from external files. The objective is to mimic a variety of real-world scenarios.

These test cases are organized within a dedicated Test class, containing distinct test methods for each case. For instance, the test methods `'test_big_exams_early_best'`, and `'test_big_exams_early_worst'` correspond to the best and worst scenarios of the 'big exams early' rule respectively.

Our goal with Quality Assurance is to thoroughly test the exam schedule evaluation system under different scenarios. This helps to make sure our system is strong, trustworthy, and can accurately check the quality of any exam schedule. We aim to highlight both what's working well and what could be better in the schedule.

4.3.2 Module: OneExamPerDay

Test1: One Exam Per Day - Best Scenario

Test Description: This tests uses a dataset where students take a maximum of one exam per day. It tests the rule under optimal conditions and expects a high score.

Input: Mock data where all students in exam plan take maximum one exam per day.

Expected Result: Score is approximately 100 out of 100. This indicates that the scheduling system is functioning properly for the best-case scenario.

Test2: One Exam Per Day - Worst Scenario

Test Description: This tests uses a dataset where students take more than one exam per day. It tests the rule under optimal conditions and expects a low score.

Input: Mock data where all students in exam plan take more than one exam per day.

Expected Result: Score is approximately 0 out of 100. This indicates that the scheduling system is functioning properly for the worst-case scenario.

4.3.3 Module: SpecialDates

Test1: Special Dates - Best scenario

Test Description: This test uses a dataset where examiners don't have to supervise exams in his/her special specific dates.

Input: Mock data where all examiners don't supervise exams in specific dates.

Expected Result: Score is approximately 100 out of 100. This indicates that the scheduling system is functioning properly for the best-case scenario.

Test2: Special Dates - Worst scenario

Test Description: This test uses a dataset where examiners have to supervise exams in his/her special specific dates.

Input: Mock data where all examiners supervise exams in specific dates.

Expected Result: Score is approximately 0 out of 100. This indicates that the scheduling system is functioning properly for the worst-case scenario.

4.3.4 Module: BigExamsEarly

Test 1: Big Exams Early - Best Scenario

Test Description: This test uses a dataset where the bigger exams are scheduled early. It tests the rule under optimal conditions and expects a high score.

Input: Mock data where larger exams are scheduled earlier in the exam schedule.

Expected Result: Score is greater or equal to 70. This indicates that the scheduling system is functioning properly for the best-case scenario and bigger exams are indeed scheduled earlier.

Test 2: Big Exams Early - Worst Scenario

Test Description: This test uses a dataset where the bigger exams are scheduled late. It tests the rule under the worst conditions and expects a low score.

Input: Mock data where larger exams are scheduled later in the exam schedule.

Expected Result: Score is less or equal to 5. This indicates that the scheduling system properly penalizes scenarios where bigger exams occur late in the schedule.

4.3.5 Module: RoomDistance

Test 1: Room Distance - Best Scenario

Test Description: This test uses a dataset where students are allocated in rooms close to each other. It tests the rule under optimal conditions and expects a high score.

Input: A combination of mock data that is either directly created as a dataframe or drawn from artificial datasets like Excel files. Both types of data are later assigned to the attributes of the Mock class. This dataset simulates a situation where students' allocated exam rooms are in close proximity, thus representing an optimal scenario for the room distance rule.

Expected Result: Score is approximately equal to 100. This indicates that the scheduling system is functioning properly for the best-case scenario and the exams are scheduled in rooms close to each other.

Test 2: Room Distance - Worst Scenario

Test Description: This test uses a dataset where students are allocated in rooms far from each other. It tests the rule under the worst conditions and expects a low score.

Input: A combination of mock data that is either directly created as a dataframe or drawn from artificial datasets like Excel files. Both types of data are later assigned to the attributes of the Mock class. This dataset simulates a situation where students' allocated exam rooms are far apart, thus representing the worst-case scenario for the room distance rule.

Expected Result: Score is approximately equal to 0. This indicates that the scheduling system properly penalizes scenarios where exams are scheduled in rooms far from each other.

4.3.6 Module: OneDayGap

Test 1: One Day Gap - Best Scenario

Test Description: This test uses a dataset where students have at least one day gap between their exams. It tests the rule under optimal conditions and expects a score close to 100, indicating no conflicts.

Input: Mock data (coming from artificial datasets like Excel files, which are passed to the Data class as parameters), where all students have at least one day gap between their exams.

Expected Result: Score is approximately equal to 0, indicating there are no back-to-back exams for students, which is the ideal scenario.

Test 2: One Day Gap - Worst Scenario

Test Description: This test uses a dataset where students do not have a one day gap between their exams. It tests the rule under the worst conditions and expects a score close to 0, indicating high conflicts.

Input: Mock data where students have exams without a day gap.

Expected Result: Score is approximately equal to 0. The score is expected to be low, indicating that the system correctly identifies the situation where students have exams without a one day gap as a problematic scheduling scenario.

4.3.7 Module: SpecialProfessors

Test 1: Special Professors - Best Scenario

Test Description: This test uses a dataset where special professors have all their exams scheduled on the same day. It tests the rule under optimal conditions and expects a score close to 100, indicating no conflicts.

Input: Mock data where all special professors are assigned to multiple examinations on the same day (for each professor individually), rather than having their exams spread out over separate days.

Expected Result: Score is approximately equal to 100, indicating that all special professors' exams are scheduled on the same day. This means they aren't required to come to campus on multiple, separate days, which is the ideal scenario.

Test 2: Special Professors - Worst Scenario

Test Description: This test uses a dataset where special professors are assigned to supervise examinations on different days, implying they have to be present on campus on multiple separate days. This is considered the worst-case scenario for this rule and expects a score close to 0, indicating a large number of conflicts.

Input: Mock data where special professors are assigned to supervise different examinations, each on a separate day. This is achieved by intentionally designing the mock data such that each special professor supervises one examination per day, but the exams are scheduled on different days.

Expected Result: Score is approximately equal to 0, indicating that special professors' exams are distributed over several days. This is the least ideal scenario, as it means these special professors have to come to campus on multiple, separate days, creating a high level of inconvenience.

4.3.8 Module: RoomCapacity

Test 1: Room Capacity - Best Scenario

Test Description: This test uses mock data where the number of students and the capacity of exam rooms are nearly identical. It tests the room capacity rule under optimal conditions and expects a high score, indicating the rooms' capacity is efficiently utilized.

Input: Mock data, directly created within the test function, representing a scenario where the number of students in each course nearly matches the capacity of the assigned exam rooms.

Expected Result: Score is greater than or equal to 80, which indicates that room capacities are efficiently used and there is a balance between the number of students and room capacities. This implies a proper utilization of resources and an optimal exam scheduling.

Test 2: Room Capacity - Worst Scenario

Test Description: This test uses mock data where the number of students is significantly lower than the capacity of the assigned exam rooms. It tests the room capacity rule under worst-case conditions and expects a low score, indicating inefficient usage of room capacities.

Input: Mock data, directly created within the test function, representing a scenario where the number of students for each course is significantly lower than the capacity of the assigned exam rooms.

Expected Result: Score is less than or equal to 10, indicating that the room capacities are not efficiently used and there is a significant imbalance between the number of students and room capacities. This implies an inefficient utilization of resources and sub-optimal exam scheduling.

4.3.9 Test Results Interpretation

If all the tests pass, it indicates that the scoring system for the module is working as expected under both the best and worst scenarios.

- If any of the tests fail, it suggests a problem with the scoring system in handling that specific scenario. The console output and failure messages should provide insight into what went wrong in these cases. The problem may stem from how the score is calculated or how the input data is processed.

5 Setup and Operation of the Software Solution

5.1 Dependencies on other software system

The application has dependencies on specific software systems and packages to ensure its smooth functioning. In order for the application to run successfully, Python 3.10.9 must be installed, along with the following required packages:

- Matplotlib 3.7.0: for data visualization.

- Pandas 1.5.3: for data manipulation.
- Numpy 1.23.5: to operate with matrices.
- Json (comes prebuilt with python): to read json file.
- Csv (comes prebuilt with python): to read csv file.
- Byte64 (comes prebuilt with python): required for html creation.
- Io (comes prebuilt with python): required for html creation.

5.2 Software Availability

The application is available in a BitBucket repository.

The BitBucket repository contains the source code, documentation, and any other relevant resources for the software. By accessing the repository, users can review the code, download the necessary files, or contribute to the development process.

5.3 Installing the Software:

To install the software, follow these steps:

- Access the BitBucket repository using the provided link.
- Clone the repository to your local machine by either downloading the repository as a ZIP file or using a Git client to clone the repository.
- Ensure that Python 3.10.9 is installed on your system. If not, download and install Python 3.10.9 from the official Python website .
- Open a command-line interface or terminal and navigate to the location where the repository was cloned or extracted.

Install the required dependencies (Matplotlib, NumPy, Pandas) by executing the following command:

- pip install -r requirements.txt

This command will install all the required packages specified in the requirements.txt file. Once the dependencies are installed, the software is ready to run.

5.4 Opportunities for later adaptation and further development

There are several potential areas for future development and improvement of the application. Let's discuss the suggested enhancements:

One could improve scoring system by including information about which examiners failed the exam and have to retake it. It can be a valuable addition to the scoring system. By considering individual examiner failures, the system can provide a fairer evaluation of exam plans, penalizing them less for consecutive exams for those specific students. This enhancement would require modifying the evaluation criteria and calculation algorithms to incorporate examiner failure data. By the time we were developing the project, we did not have data for that.

To enhance the user experience and make the application more user-friendly, improvements can be made to the installation process. This could involve creating an installer or package that simplifies the installation steps, automatically handles dependencies, and provides clear instructions for setting up and running the application. Additionally, providing a user-friendly installation guide or script can help streamline the installation process.

Also developing a Graphical User Interface (GUI) for the application can significantly enhance the user experience. A GUI would provide a visual interface with intuitive controls and interactive elements, making it easier for professors to input exam plans, view evaluation results, and interact with the system. The GUI can incorporate features such as drag-and-drop functionality, visual representations of data, and real-time updates, offering a more engaging and efficient user experience.

6 Retrospective

6.1 Evaluation of the project results

The team's assessment of the project is that while it is not considered fully complete, it showcases promising potential for further enhancements, as mentioned earlier. However, the inner logic of the application is well-designed and demonstrates a high level of quality. The project successfully incorporates multiple evaluation criteria and effectively models them in an efficient and robust manner. This accomplishment represents a significant step forward in providing a viable solution to the problem of exam scoring.

The successful implementation of the evaluation criteria demonstrates the team's ability to handle complex requirements and deliver a functional solution. The application's capacity to assess exam plans based on predefined criteria contributes to improving the quality and fairness of the evaluation process. The team's effort in designing and implementing the inner logic of the application is commendable.

Moving forward, the team acknowledges the areas for improvement highlighted earlier, such as enhancing the scoring system, improving installability, and developing a graphical user interface. By addressing these aspects, the application can be further refined to offer an even better user experience and provide more accurate evaluation results.

Overall, the team's evaluation of the project is positive, recognizing the solid foundation and potential for future improvements. The successful implementation of the inner logic and the incorporation of multiple evaluation criteria demonstrate the team's competence and commitment to delivering a valuable solution to the problem of exam scoring.

6.2 Project management method and tools used

The project team employed the Agile and Scrum methodologies for project management. These methodologies provided a suitable framework for managing the workload while maintaining flexibility in terms of work schedules. Here are some details regarding the project management method and tools used:

For our project, we used Agile and Scrum methodologies. Agile principles emphasize flexibility, collaboration, and iterative development. Scrum, a specific Agile framework, was adopted

to organize the work into time-boxed iterations (sprints) and facilitate regular communication and feedback.

The team members worked individually or in pairs, depending on the task requirements. This approach allowed for a balanced distribution of workload and encouraged collaboration and knowledge sharing among team members.

The team held regular meetings to ensure effective communication and progress tracking. This included weekly meetings with the supervisor to receive guidance, feedback, and align project goals. Additionally, weekly team meetings were conducted to discuss implementation details, and address any challenges or concerns.

For the collaboration, we were having a WhatsApp group chat. Maybe it was unprofessional from us, but that made it very easy to collaborate and discuss our project. By doing it in a regular messenger, we established efficient and quick way to communicate. For the version control system, we used BitBucket, which was offered to us in the very beginning of the project.

6.3 Description of the project process

The project process began with a kick-off session, followed by the implementation of "BigExamsEarly" over the course of approximately three sprints. Initially, the team encountered challenges in modeling the rules and started with a naive approach, but ended up with an advanced way of handling the rule. As more rules were identified, the team made a decision not to spend excessive time on implementing a perfect solution but focused on implementing each rule incrementally.

During the subsequent sprints, the team worked on one rule at a time. However, it was later realized that task splitting among team members proved to be difficult, prompting a shift to working on two rules simultaneously. The duration of sprints varied, typically ranging from one to two weeks.

In the middle of the project, the team recognized the need to rework the code structure and organization due to its initial lack of organization. This restructuring phase took around 2-3 weeks to complete. It is worth noting that during an international week, all team members participated in external activities, resulting in no project work during that period. The team resumed working on the project the following week.

Towards the end of the project, the remaining rules, specifically those related to professors, were implemented. Extensive testing was performed, and final adjustments were made to the code. Due to time constraints, the decision was made to conclude the project at this stage.

6.4 Evaluation of one's own way of working and cooperation in the team

6.4.1 Illia Rohalskyi

As a person that was having good knowledge of Python, I was having lots of questions in my team. At some point, it felt like I was taking a leadership role, since I was helping out each of the team members, answering their questions and pushing them to fight procrastination and finish the task before deadline. I found it to be difficult managing people, as everyone has their own way of thinking and you have to find individual way to interact with people. It came down to even explaining the same concept in different ways. It felt like speaking separate languages.

But I think I did a good job, as my team was grasping what I was explaining to them. There were technical difficulties due to different backgrounds. Me and Kemal Eren knew how to do linear regression and we proposed lots of ways of doing evaluation criterias, but we were not as proficient in object oriented programming and organising the project as Kaan and Muberra were. In the end, I believe, because we were very diverse, it turned out to be a very juiceful blend of our knowledge.

6.4.2 Kaan Özer

There were a few challenges. First of all, gathering every team member for a meeting was very difficult because each member had different exams, responsibilities, and needed to go to other cities. However, we solved this issue by discussing the days that suited everyone, meeting at the university to prevent distance problems, and sometimes having online meetings.

Secondly, there were some technical problems. I was not familiar with programming in Python and didn't have enough knowledge about regressions, pilots, and machine learning topics.

Thirdly, we sometimes had communication problems and couldn't agree on certain topics. We arranged a meeting with Peter Braun to ask our questions and clarify everything.

It was challenging to write code in Python. Therefore, I started with implementing the fundamental rules and sought help from my teammates when things became more difficult. Additionally, I had other skills from my past experiences. Apart from implementing rules, I also assisted in the output processes to make our results visible on the website, enabling us to discuss on results and generate a comprehensive report with the total score. Moreover, I had experience with LaTeX, so I contributed to the documentation by embedding my teammates' findings to LaTeX. I attended all meetings, just like my teammates, and frequently expressed my opinions about the tasks and implementation processes.

6.4.3 Müberra Şeyma Uslu

At the beginning, it was a bit challenging to decide on which programming language to use, as we were familiar with different languages. Additionally, understanding each other's perspectives was difficult due to our individual coding styles. However, we overcame these communication problems by frequently coming together and having discussions. Through these conversations, we identified each team member's strengths and weaknesses, allowing us to allocate tasks accordingly and proceed with the project in that manner.

6.4.4 Kemal Eren Öztürk

Communication Problems:
There were a few instances of communication breakdowns within the team. Sometimes ideas were not communicated clearly, leading to misunderstandings and confusion. This resulted in some inefficiencies and rework. However, we quickly realized the importance of proper communication and made efforts to improve our clarity and actively listen to each other.

Distributing Tasks:
Initially, all team members attempted to code a single Python program, which proved to be highly ineffective. As a result, we decided to split into two groups, each responsible for a specific aspect of the project. This division significantly improved our efficiency and allowed for smoother progress in both designing and coding.

Design:
The process of programming each rule was relatively easy and straightforward. However, designing the software architecture posed significant challenges. We struggled with finding the most efficient and effective way to structure the software, which caused delays and confusion. But in the end, we pushed ourselves to design a proper software design.

Procrastination:
Procrastination became a minor issue during the project. We occasionally found ourselves delaying tasks or not fully utilizing our time. This led to some unnecessary pressure and rushed work towards the end. However, we were able to recognize this problem early on and actively worked on improving our time management and productivity.

6.5 Lessons learned for future projects

6.5.1 Illia Rohalskyi

I was not very familiar with git, so this really will help me in my future. I also was not aware of structuing the project, and, I guess, I would do it differently. I would think of structure in the very first meeting, instead of reworking everything in the middle of project development.

In contrast to my fellows, I believe, Python was a wise choice for this project. Python has a great ecosystem to work with data and is also very simple. In my perspective, if we would use other language (such as Java, Rust or any C-based language), our team would have harder times implementing the project, as those languages are more difficult to learn. We also did not need those extra miliseconds of faster running time. So overall, in my opinion, python was perfect for this task.

I learned how to communicate with people, share ideas and manage team with different backgrounds. I guess, this is the most valuable lesson from this project.

6.5.2 Kaan Özer

I learned that there is no shame in restarting everything from the beginning several times in a project because sometimes we might change our minds and need to modify our classes and implementations.

I learned that Starting a project with a new language that you don't know at all is not a big deal after learning some programming basics and having good teammates. Additionally, I learned many things about machine learning and the Python environment.

In future projects, I will avoid the mindset of "Let's complete it first, and then we'll organize it," as it often leads to a significant waste of time. Instead, I recognized the importance of organizing everything from scratch to the production, as procrastination can result in a chaotic and troublesome outcome.

6.5.3 Müberra Şeyma Uslu

I had never been involved in a team project where we collaborated on coding for such a long period before, so it was a significant plus and learning experience for me. Although I had previous experience with Git, I had not worked extensively with a version control system as a group, which allowed me to learn and utilize Git at a more advanced level. Our goal was to evaluate an existing exam plan based on specific criteria, and since each rule had different

evaluation criteria, we had to analyze the available data in various ways. For example, we leveraged regression for one of the rules. That was bit intensive and hard for me but I had fun while working on it. If I were to develop a similar program again, I would opt for a C-based programming language instead of Python.

6.5.4 Kemal Eren Öztürk

Don't code the first thing that comes to mind: Rushing into coding without sufficient planning and consideration can lead to inefficiencies and complications later on. It is essential to take the time to properly analyze and design the software architecture before diving into implementation. Explaining ideas properly to team members is crucial: Clear and effective communication is vital for efficient teamwork. It is essential to ensure that ideas and concepts are communicated clearly to avoid misunderstandings and confusion among team members. Encouraging active listening and providing regular opportunities for clarification can significantly improve collaboration.

7 Summary and Outlook

The technical results of the project are significant, as the team successfully implemented the core functionality of the application. The application provides a scoring system for evaluating exam plans based on predefined criteria, generating reports, visualizations, and identifying conflicts. The implemented solution demonstrates a good understanding of the problem domain and effectively addresses the primary objectives.

For the future development, one could refer to part 5.4, where it is broken down in details. But in nutshell: UI, better installability, more rules Throughout the project process, regular meetings were conducted with the project supervisor and team members to discuss progress, resolve challenges, and align goals. The Agile and Scrum principles guided the iterative development, allowing for flexibility and adapting to changing requirements as needed.

By following this iterative approach and incorporating stakeholder feedback, the project evolved over time, resulting in an application that addressed the core objectives of scoring exam plans while accommodating potential future enhancements and improvements.