UNIVERSITY OF WESTMINSTER

INFORMATICS INSTITUTE OF TECHNOLOGY

Informatics Institute of Technology

BEng (Hons) in Software Engineering

Software Engineering Principles

5SENG007W

Individual Report

- Aaysha Fazal Mohamed
- UoW Number – w1956175
- IIT Number – 20221493
- Group – G02

Table of Contents

1. Reflecting on the design methodology lifecycle

The software development methodology is widely used in the software industry to describe the stages of creating a successful software product. This methodology is an excellent tool for planning, writing, modifying, and maintaining software. and to optimise the outcome.

The Agile methodology is a project management approach that emphasises continuous collaboration and improvement. It involves breaking the project into small phases and making it easily manageable. The Agile methodology is designed to adapt to changes quickly and with ease, and teams adopt constant changes via a repetitive approach to software design and development. (Nehra, M., 2022).

Agile was determined to be incorporated as the design methodology to implement this student performance application to enhance the educational experience of the students of Sri Lanka. Agile was chosen to be the best fit after a close analysis and it concluded that the project must have the capability to ever evolve with the trends and technological features and also cater to all the student's needs to enhance the educational experience.

This design methodology enables developers to ensure that software is produced with the lowest cost and highest possible quality in the shortest time. There are seven stages in the SDLC, and six standard models used for different projects. (Coursera Staff, 2023).

The seven stages of the SDLC and how compatible Agile methodology is given below.

1. Plan and Brainstorm:

   In agile methodology, planning is done in short cycles or sprints, and it is capable of quick adjustments and changes to the plan needed.

2. Analyse requirements:

   This methodology emphasises collaboration between the development team and the customer, which helps ensure that the requirements are well-understood and met.

3. Design:

   Agile methodology, design is done iteratively, with each iteration resulting in a prototype, draft, or workable version of the final product.

4. Develop:

   Agile methodology involves breaking the project into small phases, which makes it easier to manage and develop.

5. Test:

   In the Agile methodology, testing is done continuously throughout the development process, which helps ensure that the product meets the requirements.

6. Deploy:

   Agile methodology emphasises collaboration between the development team and the customer, which helps ensure that the product is deployed successfully.

7. Maintain:

   In the Agile methodology, maintenance is done continuously throughout the development process, which helps ensure that the product is always up-to-date and meets the customer's needs

To sum it up, the Agile methodology is an effective project management approach that emphasises collaboration, adaptability, and breaking projects into small phases. Following the seven stages of the SDLC, Agile methodology can ensure that software is produced with the lowest cost and highest possible quality in the shortest time. Incorporating Agile methodology is suitable for different projects, including the student performance application, to enhance the educational experience for students. (Coursera Staff, 2023).

1.1 Analysis of Project Management Methodology

Several project management methodologies can be employed for effective project execution. Among these methodologies are Agile, Waterfall, and Spiral, each with its unique strengths and weaknesses. Agile methodology, for instance, is known for its remarkable flexibility and focus on collaboration, adaptability, and customer satisfaction. It suits projects with constantly evolving requirements and encourages regular end-user feedback. On the other hand, the Waterfall methodology is more structured and ideal for projects with well-defined requirements and a fixed timeline. Lastly, Spiral methodology is a risk-driven approach that emphasises risk analysis and mitigation. With a focus on identifying and addressing potential risks early on, the Spiral methodology allows for greater predictability and control over project outcomes.

Methodology	Strengths	Weakness
Agile	Flexibility, adaptability, customer satisfaction, teamwork, accountability, face-to-face communication.	It may not be suitable for projects with well-defined requirements and a fixed timeline.
Waterfall	Structured, predictable, easy to understand, easy to manage.	Not adaptable to changing requirements, less customer engagement, less feedback from end-users.
Spiral	Risk-driven, emphasises risk analysis and mitigation, adaptable to changing requirements.	Complex, difficult to manage, may require more resources.

Table 1: Analysis of Project Management Methodology

Agile methodology is an excellent fit for a student performance monitoring application as it requires constant end-user feedback. This ensures that the app meets the needs of students and teachers, and that any necessary changes can be made quickly. Additionally, Agile promotes teamwork, accountability, and face-to-face communication, which can help ensure that all stakeholders are on the same page and working towards a common goal. However, it's important to note

that each project management methodology has its strengths and weaknesses, and the choice of methodology should be based on the specific requirements and constraints of the project. Therefore, evaluating the project's needs is crucial before deciding on a methodology.

1.2. Risk and Mitigation

The student performance applications are designed to help students improve their academic performance. However, several risks associated with these applications need to be mitigated. (NEA, 2021).

1. Privacy and Security - Student performance applications collect sensitive data such as grades, attendance, and behavioural patterns, which can be misused if not secured properly. To mitigate this risk, student performance applications should use encryption to protect sensitive data and implement access controls to ensure that only authorised personnel can access the data. (Soumya, MD. and Krishnamoorthy, S., 2021).

2. Bias - Student performance applications may be biased towards certain groups of students, leading to unfair treatment and discrimination. To mitigate this risk, student performance applications should be designed to be fair and unbiased. This can be achieved by using diverse data to train the application and by regularly auditing the bias application. (Soumya, MD. and Krishnamoorthy, S., 2021).

3. Accuracy - Student performance applications may not always provide accurate predictions or recommendations, which can lead to poor decision-making and negative outcomes. To mitigate this risk, student performance applications should be validated using real-world data and by comparing their predictions to actual outcomes. (Soumya, MD. and Krishnamoorthy, S., 2021).

2. Class Diagram and its Justification.

The diagram below depicts the class diagram of the student performance monitoring system. As the design methodology utilised for this study was object-oriented, using class diagrams is necessary to illustrate the components needed to create the prototype. (Pressman, S., 2010).

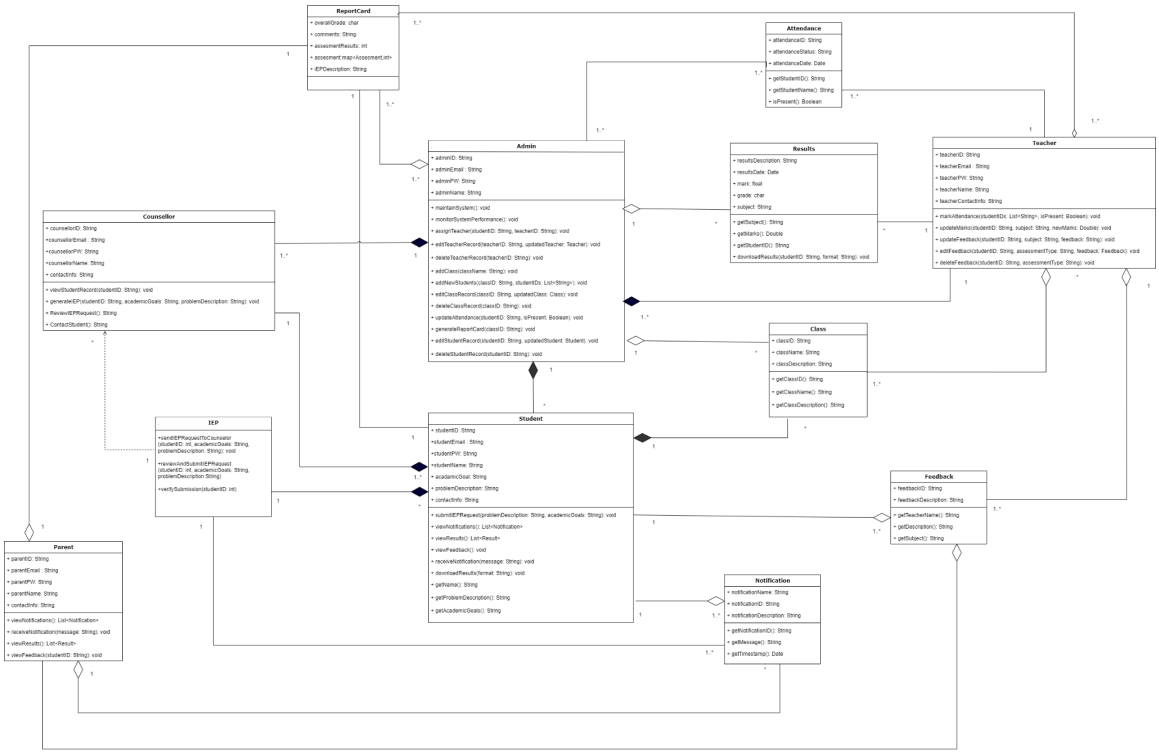

Figure 1: Class diagram for Students performance monitoring application.

The figure above illustrates the class diagram of the student performance monitoring system.

Classes are briefly explained below.

The student performance monitoring application can view results and feedback and request an Individualized Education Program (IEP) when a student faces a learning difficulty. Additionally, the system can send notifications for low attendance or low marks and when an IEP request has been completed.

There are multiple well-defined user roles, including students, counsellors, and administrators, each with specific responsibilities in the student performance monitoring process.

The system allows users (in this context, students) to submit, review, and manage Individualized Education Plan (IEP) requests. Users can log in, navigate to the IEP request section, and enter information about academic goals and specific issues.

When requesting a submission for IEP, Students log in and navigate to the IEP request section to enter information, academic goals, and problem descriptions. After submission, a notification is sent to the student's assigned counsellor.

Then, the system allows Counsellors to review the IEP request and contact the student to schedule a meeting for a detailed assessment.

The counsellor meets with the student for assessment, and the details are entered into the system. Then, he/ she submits the IEP for review. The system reviews and approves/rejects the submitted IEP based on the completion of the request.

Feedback notifications are sent to the student for adjustments if rejected. Notifications of successful IEP requests is sent to the assigned counsellor, student, and relevant staff if approved.

The system has authentication and access flows for student, counsellor, and administrator dashboards. It also facilitates communication between the counsellor and student while validating and storing IEP request information.

The student class contains details about the student's IEP request-related data and multiple students' data (For IEP requests) that can be managed.

This system ensures a structured and organised approach to managing IEP requests within a school, involving multiple stakeholders and leveraging a user-friendly interface for effective communication and decision-making.

CRC cards were used here in a table format to promote a collaborative system design and help to develop a better understanding of objects. (Wells, D., 2009).

Class	Responsibility	Collaboration
Student	Contains student details.	Feedback, Notification, IEP, Counsellor, Class.
Teacher	Contains teacher details.	Attendance, Results, Feedback, Admin.
Parent	Contains parent details.	Notification, ReportCard, Feedback.
Counsellor	Contains counsellor details.	Student, IEP.
Admin	Contains admin details.	Teacher, Class, Student, Attendance, Results, Counsellor.
Class	Contains class details.	Teacher, Admin.
Notification	Contains notification details.	Student, Parent.
Feedback	Contains feedback details.	Student, Teacher, Parent.
IEP	Contains Individualized Education plan details.	Student, Counsellor, Notification.
Attendance	Contains student attendance details.	Teacher, Admin.
Results	Contains results details of students.	Teacher, Admin.
ReportCard	Contains generated report card details.	Admin, Teacher, Student, Parent

Table 2: CRC of student's performance monitoring application

The student class is an important component of a student management system. It is responsible for storing and managing vital information such as student ID, email, password, names, academic goals, and academic challenges faced. This information is crucial for creating a personalised and effective learning experience for each student.

In addition to storing information, the student class allows students to request Individualized Education Plans (IEPs), view notifications when received, view their academic results, and download important documents. These features help to ensure that students have access to the resources they need to succeed academically.

The student class is a critical link between students and educators, fostering seamless communication and collaboration. By offering a centralised hub for essential information and resources, this platform creates a well-organized and efficient learning environment. This approach enables students to access all the necessary information effortlessly, empowering them to easily achieve their academic aspirations.

The Teacher class is a vital component of any academic system, as it contains crucial information such as the teacher's ID, email, password, name, and contact details. With this information, teachers can perform various functions, including marking attendance, updating student grades, providing feedback based on performance, and editing or deleting previously given feedback. This information is critical to the success of any educational institution, and the Teacher class is an essential tool for teachers to manage their classes effectively.

The Parent class is a crucial component of this system that empowers parents to stay informed about their child's educational journey. It stores important details such as the parent's ID, email, password, name, and contact information. By using the Parent class, parents can view notifications, receive updates about their child's academic performance, track their exam scores, and read teacher feedback. This feature provides parents with a comprehensive view of their child's progress and helps them stay involved in their education.

In this student performance monitoring system, the role of a counsellor is vital in assisting students who may be facing personal difficulties. Counsellors provide support to help students overcome their problems and make appropriate changes in their academic journey. The counsellor class contains important details such as counsellor ID, email, password, name and contact information. Using this information, the counsellor can access a student's record, generate an Individualized Education Plan (IEP), and schedule a one-on-one session with the student to gain feedback on their academic shortcomings. Once the feedback is received, the counsellor will then create an IEP to address the student's specific needs. Before contacting the student, the counsellor will first review the student's request and validate it before accepting it.

The role of the admin is crucial in ensuring that this system runs smoothly. The admin is responsible for directing the main functionalities of the application and ensuring a seamless experience for all users. Admin credentials, including the admin ID, email, password, and name, are securely stored within the system.

Beyond managing the system, the admin also monitors its performance, ensuring that any issues are identified and resolved promptly. The admin has the ability to assign, edit, or delete teachers, classes, and students as needed. With access to student records, the admin can generate student reports and make necessary edits to ensure accuracy.

Additionally, the admin has the authority to delete student records when necessary. Overall, the admin plays a vital role in maintaining the integrity and functionality of this system.

The class will manage the system and will allow the school to store important information such as class IDs, class names, and class descriptions. This will enable efficient tracking and recording of all classes available within the school system, ensuring a seamless workflow.

The Notification Class in this system saves notification types based on their name and description, allowing students and parents to receive notifications related to their generated IEP, low attendance, and low results warnings. Users can receive these messages with a time stamp, ensuring they are always up-to-date with important information about their academic progress.

The Feedback class is a crucial component of the system as it stores the unique feedback ID of each instance along with their descriptions and the relevant subject name. Once the teacher uploads the feedback, it is added to the system, where it can be accessed and reviewed by relevant parties. This ensures that the feedback loop is complete and valuable insights can be gained to improve the education experience for everyone involved.

The IEP class of the student management system is designed to send requests to the appropriate counsellor as soon as a student becomes eligible for it. When students apply for an IEP, they need to review and submit the request for verification. This ensures that all the necessary information is provided and the request is properly processed.

The attendance class is a reliable system that allows teachers to easily keep track of student attendance daily. By uploading the status of each student in a boolean value, the system accurately records who is present and who is absent, making it a convenient and time-saving solution for teachers and administrators alike. With this system in place, there is no need for manual attendance-taking, reducing the risk of errors and ensuring that attendance records are always up-to-date and accurate.

The Results class provides a detailed description of the release date, marks, and grades earned in each subject. This information is stored and used to generate appropriate reports, which are then used to provide student feedback. This helps to identify areas where students need improvement and allows for targeted support to be provided where necessary.

The Report card class functions as the main class to store all academic-related information of students, In order to generate an accurate report card for each student, the class stores information on their grades and comments from each assessment. This data is then stored in a map along with other relevant information such as IEP descriptions. This ensures that the appropriate report card is generated and that each student's progress is accurately reflected.

2.1 A summary of OOP concepts is found within the class diagram.

Aggregation Relationship:

Aggregation is a fundamental concept in object-oriented programming, where it allows for the creation of complex structures by combining simpler objects. In an aggregation relationship, one object is made up of one or more sub-objects, but these sub-objects can still exist independently outside of the main object. This helps to make programs more modular, flexible, and easier to maintain over time. (No Author, 2022).

1. Students can receive feedback. Student and Feedback classes indicate that a student can receive one or many feedback instances associated with them, and each feedback instance can belong to only one student. The multiplicity is one-to-many (1:*).
2. Counsellors give IEPs to students. The Counsellor and Student classes imply that a counsellor can be assigned one or more students simultaneously, and each student can have only one counsellor. The multiplicity is one-to-many (1:*).
3. Admin and Class classes signify that the admin can add one or more classes to the system, and the system can have multiple classes. The multiplicity is one-to-many (1:*).

Composition Relationship:

Composition relationship is a fundamental object-oriented programming concept that defines a relationship between classes, wherein one class is composed of one or more sub-classes, and the sub-classes are integral parts of the main class. The whole and the parts have a consistent lifetime and cannot be separated. In other words, the sub-classes have a strong ownership

relationship with the main class. If the main class ceases to exist, all the sub-classes will cease. This relationship is often used to model complex objects and highlight the importance of the sub-classes to the main class, as they cannot exist without each other. (No Author, 2022).

1. Student and Notification classes indicate that each student is composed of one or more notification objects, indicating that notifications are an integral part of the student. The multiplicity is one-to-many (1:*).
2. Parent and Notification classes suggest that a parent object comprises one or more notification objects, implying that notifications are part of a parent's composition. The multiplicity is one-to-many (1:*).
3. Parent and Student classes imply that a parent object is composed of the student class, meaning that the existence of a parent is closely tied to the existence of one or more students. The multiplicity is one-to-many (1:*).
4. The Composition Relationship between the Counsellor and IEP classes suggests that the counsellors are composed of the IEP class, indicating that IEPs are part of a counsellor's composition. The multiplicity is one-to-many (1:*).
5. The Composition Relationship between the Admin and Student classes indicates that the admin objects are composed of the student class, and the system would not function without each other. The multiplicity is one-to-many (1:*).

Association Relationship:

Association is a vital relationship in object-oriented programming that establishes a connection between two distinct object types. It allows a property of a class to hold a reference to an instance or multiple instances of another class. Although combinations and aggregations also fall under associative relationships, the bond between classes of associations is relatively weaker than the other two. However, it is the most commonly used relationship in programming, as it facilitates the smooth interaction between different types of objects, leading to efficient and reliable software systems. (No Author, 2022).

1. The Association Relationships between the Teacher and Attendance/Result/Feedback classes imply that the teacher can associate with and upload attendance details, results, and feedback for one or many instances. The multiplicity is one-to-many (1:*).
2. The Association Relationship between the Admin and Attendance/Result classes signifies that the admin can associate with and update attendance details and results in the system. The multiplicity is one-to-many (1:*).
3. The Association Relationship between the Parent and Report Card classes indicates that the parent can associate with and view report cards for one or more students. The multiplicity is one-to-many (1:*).

Dependency:

Dependencies are a fundamental concept in object-oriented programming that reflects the relationship between different classes and objects within a system. A dependency relationship is typically referred to as a "use" relationship, where one class uses another class to perform a particular operation or function. Dependencies are crucial because a change in one class can potentially affect other classes that depend on it, making it necessary to indicate dependencies where they exist. This relationship is often reflected in the methods of a class that use another class's object as a parameter. Overall, understanding dependencies is essential for building software systems that are robust, reliable, and maintainable over time, as well as being able to adapt to changing requirements and user needs. (No Author, 2022).

1. IEP requests depend on and cannot exist without the presence of students.
2. Counsellors and admins are interdependent, meaning that the system relies on the existence of both counsellors and admins.

A class diagram falls under the category of structure diagrams in the Unified Modelling Language (UML) and is utilised to represent the static structure of a system. It is instrumental in describing the various classes and their interrelationships in an object-oriented programming (OOP) system. (Nishadha, S., 2022).

Architecture, on the other hand, refers to the process of designing and planning a system. This entails making informed decisions regarding the different components of the system, their relationships, and how they function together.

In software engineering, architects make use of class diagrams to design and document the coded (or soon-to-be-coded) classes of the system. Additionally, they can employ component and deployment diagrams to test and validate the effectiveness of their design. (No Author, 2023).

Thus, class diagrams play a crucial role in the architecture design process, as they enable architects to visualise the system's structure and the relationships between classes. This is essential in creating a robust and scalable system.

3. Testing of the class diagram and Class Structure

3.1 Testing class diagram

At the heart of Object-Oriented Programming (OOP) lies the concept of a class, which serves as a blueprint for creating objects with specific attributes and behaviours. Class testing is an important part of OOP, and it involves verifying that the implementation of a class matches its specifications.

Data flow testing is a type of structural testing which involves in examining the data flow within the given software program. It is used to detect issues such as uninitialized variables or using variables before assigning them a value. (Siewert, A., 2023).

1. *Thoroughness:*

   Data flow testing is a comprehensive set of testing methodologies that thoroughly examines the intricate interplay between program variables' definitions and their uses. Each test objective is commonly referred to as a "def-use pair." The primary aim of data flow testing is to meticulously select test data guided by various test adequacy criteria, often termed data-flow coverage criteria. These criteria help ensure a comprehensive exercise of each def-use pair within the program's code. (Siewert, A., 2023).

2. *Precision:*

   Data flow testing is a method of analysing software programs in-depth. This method examines the complex path data takes as it flows through the program, particularly the variables used within the code. The aim is to identify possible issues related to variable usage, such as uninitialized variables or variables that are used before, they are assigned a value. (Siewert, A., 2023).

3. *Effectiveness:*

   Data flow testing provides valuable insights into data behavior during program execution, complementing static data flow testing findings.

   Understanding class testing and data flow testing is important in software development. Data flow testing can help identify errors related to variable usage, making it useful for school management to ensure reliable software for managing school operations. (No Author, 2023).

Testing templates of the main classes of this system are given below.

SCREEN NAME										
PREPARED BY		Aaysha Mohamed								
DESIGNATION		Student								
DATE		31.12.2023								
SCENARION ID		SCENARIO DESCRIPTION		IEP class testing						
S.NO	**TEST CASE ID**	**TEST CASE DESCRIPTION**	**PRECONDITION**	**TEST DATA**	**EXPECTED RESULTS**	**POSTCONDITION**	**ACTUAL RESULT**	**STATUS**	**DEFECTED ID**	**COMMENTS**
1	TC001	Student Enters Information, Academic Goals, and Problem Description.	1. The field is filled with accurate information.	<Valid user input> 1. Valid user password 2. Valid ID– contains 8 digits E.g. 19561750 3. Accurate academic goals description. 4. Accurate Problem Description	The form is Complete Review and submit	ReDirects to the review page.	Displays "The form is Complete Review and submit. "	Pass	NULL	Valid user input
2	TC001	Student Enters Information, Academic Goals, and Problem Description.	1. The field is filled with accurate information.	<Invalid user input> 1. Invalid user password – Doesn't have 8 characters. 2. Valid ID – Doesn't contain 8 digits. E.g. 19561750 3. Accurate academic goals description. 4. Accurate Problem Description	The form is Complete Review and submit	ReDirects to the review page.	The form is not submitted and throws an error, It omits the Academic goal description and Problem description. Display "Invalid User password"	Fail	IEP_DEFECTID1A	Invalid user input
2	TC001	Student Enters Information, Academic Goals, and Problem Description.	1. The field is filled with accurate information.	<Invalid user input> 1. Valid user password 2. Invalid ID 3. Accurate academic goals description. 4. Accurate Problem Description	The form is Complete Review and submit	ReDirects to the review page.	The form is not submitted and throws an error, It omits the Academic goal description and Problem description. Display "Invalid ID"	Fail	IEP_DEFECTID2B	Invalid user input
4	TC001	Student Enters Information, Academic Goals, and Problem Description.	1. The field is filled with accurate information.	<Invalid user input> 1. Valid user password 2. Valid ID 3. Incomplete academic goals description. 4. Accurate Problem Description	The form is Complete Review and submit	ReDirects to the review page.	The form is not submitted and throws an error Display "Incomplete Academic goal description!, Fill all forms."	Fail	IEP_DEFECTID3C	Invalid user input
5	TC001	Student Enters Information, Academic Goals, and Problem Description.	1. The form is filled with accurate information.	<Invalid user input> 1. Invalid user password 2. Valid ID 3. Accurate academic goals description. 4. Incomplete Problem Description	The form is Complete Review and submit	ReDirects to the review page.	The form is not submitted and throws an error Display "Incomplete Academic goal description!, Fill all forms."	Fail	IEP_DEFECTID4D	Invalid user input

6	TC001	Student Enters Information, Academic Goals, and Problem Description.	1. The form is filled with accurate information.	<Invalid user input> 1. Invalid user password 2. Invalid ID 3. Accurate academic goals description. 4. Incomplete Problem Description	The form is Complete Review and submit	ReDirects to the review page.	The form is not submitted and throws an error Display "Incomplete ProblemDescription."	Fail	IEP_DEFECTID5C	Invalid user input
7	TC002	Review and Submit	1. The filled form is accurate and complete.	<Valid user input> 1. Valid user password 2. Valid ID	The form is submitted. "The Submission was successful!"	The student's IEP request is sent to their counsellor's email address. • The student receives a notification that their IEP request has been submitted.	The form Submission was Successful!	Pass	NULL	Valid User Input
8	TC003	Review and Submit	1. The filled form is accurate and complete.	<In valid user input> 1. Invalid user password 2. Valid ID	The form is submitted. "The Submission was successful!"	The student's IEP request is sent to their counsellor's email address. • The student receives a notification that their IEP request has been submitted.	The form Submission was not Successful. Display "Invalid user password"	Fail	IEP_DEFECTID2A	Invalid user input
9.	TC003	Review and Submit	1. The filled form is accurate and complete.	<In valid user input> 1. Valid user password 2. Invalid ID	The form is submitted. "The Submission was successful!"	The student's IEP request is sent to their counsellor's email address. • The student receives a notification that their IEP request has been submitted.	The form Submission was not Successful. Display "Invalid userID"	Fail	IEP_DEFECTID2B	Invalid user input
10.	TC003	Review and Submit	1. The filled form is accurate and complete.	<in valid user input> 1. Invalid user password 2. Invalid ID	The form is submitted. "The Submission was successful!"	The student's IEP request is sent to their counsellor's email address. • The student receives a notification that their IEP request has been submitted.	The form Submission was not Successful. Display "Invalid userID or Password"	Fail	IEP_DEFECTID2C	Invalid user input
11.	TC004	Verify Submission	1. The filled form is accurate and complete.	<Valid user input> 1. Valid user password 2. Valid ID	The form is submitted. Display "Verification was successful."	The student's IEP request is sent to their counsellor's email address. • The student receives a notification that their IEP request has been submitted.	The form Submission was Successful. Display "Verification was successful."	PASS	IEP_DEFECTID2D	Valid user input
12.	TC004	Verify Submission	1. The filled form is accurate and complete.	<Invalid user input> 1. Invalid user password 2. Valid ID	The form is submitted. Display "Verificatio	The student's IEP request is sent to their counsellor's email address.	The form Verification was not Successful.	Fail	IEP_DEFECTID3A	Invalid user input

					n was successful."	• The student receives a notification that their IEP request has been submitted.	Display "Invalid user password"			
13.	TC004	Verify Submission	1. The filled form is accurate and complete.	<In valid user input> 1. Valid user password 2. Invalid ID	The form is submitted. Display "Verification was successful."	The student's IEP request is sent to their counsellor's email address. • The student receives a notification that their IEP request has been submitted.	The form Verification was not Successful. Display "Invalid userID"	Fail	IEP_DEFECTID3B	Invalid user input
14.	TC004	Verify Submission	1. The filled form is accurate and complete.	<in valid user input> 1. Invalid user password 2. Invalid ID	The form is submitted. Display "Verification was successful."	The student's IEP request is sent to their counsellor's email address. • The student receives a notification that their IEP request has been submitted.	The form Verification was not Successful. Display "Invalid userID or Password"	Fail	IEP_DEFECTID3c	Invalid user input

SCREEN NAME										
PREPARED BY		Aaysha Mohamed								
DESIGNATION		Student								
DATE		31.12.2023								
SCENARION ID		SCENARIO DESCRIPTION		Counsellor Class Testing						
S.NO	**TEST CASE ID**	**TEST CASE DESCRIPTION**	**PRECONDITION**	**TEST DATA**	**EXPECTED RESULTS**	**POSTCONDITION**	**ACTUAL RESULT**	**STATUS**	**DEFECTED ID**	**COMMENTS**
1	TC001	View Student record	1. Logged into the system. 2. Student credentials are accurate. 3. Student records must be available	<Valid user input> 1. Valid student Username 2. Valid student email	Displays student record.	The student record remains accessible for further action.	Displays student record.	Pass	NULL	Valid user input
2	TC001	View Student record	1. Logged into the system. 2. Student credentials are accurate. 3. Student records must be available	<Valid user input> 1. Invalid student Username 2. Valid student email	Displays student record.	The student record remains accessible for further action.	Displays " Invalid Student Name"	Fail	STU_DEFECT_A1	Invalid user input
3	TC001	View Student record	1. Logged into the system. 2. Student credentials are accurate. 3. Student records must be available	<Valid user input> 1. Valid student Username 2. Invalid student email	Displays student record.	The student record remains accessible for further action.	Displays " Invalid Student email"	Fail	STU_DEFECT_A2	Invalid user input
4	TC001	View Student record	1. Logged into the system. 2. Student credentials are accurate. 3. Student records must be available	<Valid user input> 1. Valid student Username 2. Valid student email	Displays student record.	The student record remains accessible for further action.	Displays " Student Record is unavailable."	Fail	STU_DEFECT_A3	System error
5	TC002	Generate IEP	1. Logged into the system. 2. Student credentials are accurate. 3. Student records must be available	<Valid user input> 1. Valid student email. 2. The information entered is complete and accurate.	Displays student record. The IEP is successfully generated for the specified student.	The system displays a confirmation message, and the generated IEP is stored in the system.	The system displays a confirmation message, and the generated IEP is stored in the system.	PASS	NULL	Valid User input
6	TC002	Generate IEP	1. Logged into the system. 2. Student credentials are accurate. 3. Student records must be available	<Valid user input> 1. Invalid student email. 2. The information entered is complete and accurate.	Displays student record. The IEP is successfully generated for the specified student.	The system displays a confirmation message, and the generated IEP is stored in the system.	The system displays an Error message. Display "Invalid student Email"	Fail	STU_DEFECT_B1	Invalid User input

7	TC002	Generate IEP	1. Logged into the system. 2. Student credentials are accurate. 3. Student records must be available	<Valid user input> 1. valid student email. 2. The information entered is not complete.	Displays student record. The IEP is successfully generated for the specified student.	The system displays a confirmation message, and the generated IEP is stored in the system.	The system displays an Error message. Display "Fill in all fields"	Fail	STU_DEFECT_C1	Invalid User input
8	TC002	Generate IEP	1. Logged into the system. 2. Student credentials are accurate. 3. Student records must be available	<Valid user input> 1. Valid student email. 2. The information entered is complete and accurate.	Displays student record. The IEP is successfully generated for the specified student.	The system displays a confirmation message, and the generated IEP is stored in the system.	The system displays an Error message. Display "Fill in all fields"	Fail	STU_DEFECT_C1	System error
9	TC003	Review IEP	1. Logged into the system. 2. Student credentials are accurate. 3. Generated IEP must be accurate.	<Valid user input> 1. Valid student email. 2. The Generated IEP must be complete and accurate.	Displays generated IEP for student record.	The system displays a confirmation message, and the generated IEP is stored in the system. Counsellor uploads to the system.	The system displays a confirmation message, and the generated IEP is stored in the system.	PASS	NULL	Valid User input
10	TC003	Review IEP	1. Logged into the system. 2. Student credentials are accurate. 3. Generated IEP must be accurate.	<Valid user input> 1. Invalid student email. 2. The Generated IEP must be complete and accurate.	Displays generated IEP for student record.	The system displays a confirmation message, and the generated IEP is stored in the system. Counsellor uploads to the system.	The system displays "Invalid student email"	Fail	STU_DEFECT_D1	Invalid User input
11	TC003	Review IEP	1. Logged into the system. 2. Student credentials are accurate. 3. Generated IEP must be accurate.	<Valid user input> 1. Valid student email. 2. The Generated IEP not complete.	Displays generated IEP for student record.	The system displays a confirmation message, and the generated IEP is stored in the system. Counsellor uploads to the system.	The system displays "Fill out all fields"	Fail	STU_DEFECT_D2	Invalid User input
12	TC003	Review IEP	1. Logged into the system. 2. Student credentials are accurate. 3. Generated IEP must be accurate.	<Valid user input> 1. Valid student email. 2. The Generated IEP must be complete and accurate.	Displays generated IEP for student record.	The system displays a confirmation message, and the generated IEP is stored in the system. Counsellor uploads to the system.	The system displays "Permission Denied"	Fail	STU_DEFECT_D3	Permission not given to particular students data.

Figure 2: Test Case templates for IEP class and Counsellor class

3.2 Refining Functional Requirements through Class Diagram

The refinement of functional requirements is a crucial step in the software development process. It is widely acknowledged that getting the requirements right is the key to the success of any project. This process consists of a set of activities that produce requirements for a product. According to the systems engineering standard, a requirement is defined as "something that governs what, how well, and under what conditions a product will achieve a given purpose." In essence, requirements define the functions, performance, and environment of the system under development to a level that can be built through functional and non-functional requirements. It is, therefore imperative to pay utmost attention to this crucial step in order to ensure the successful development of a software product. (Kasse Initiatives, 2004).

1. **Analyse and Prioritize Requirements:**

   Identify critical requirements such as IEP submission, notification triggers, and result viewing. Prioritize based on impact on core functionalities.

2. **Identify and Eliminate Ambiguities:**

   Terms within the system must be clearly defined, and any ambiguous statements should be clarified to avoid misunderstandings.

3. **Ensure Completeness:**

   Ensure comprehensive coverage by confirming essential aspects like IEP management, notifications, results, and feedback and addressing any gaps in the requirements.

4. **Validate Requirements:**

   Assess the feasibility of automatic notifications and real-time updates. Ensure system capabilities align with requirements.

5. **Document Requirements:**

   Assess the feasibility of automatic notifications and real-time updates. Ensure system capabilities align with requirements.

6. **Review and Refine:**

   It is important to regularly review requirements with stakeholders and refine them based on feedback, technology updates, or evolving educational needs.

Applying these steps ensures that the student performance monitoring system's requirements are well-analysed, prioritised, clear, comprehensive, realistic, and continuously refined throughout development. This approach contributes to the successful development of a system that meets the needs of all stakeholders.

4. System Architecture

When it comes to software development, the system architecture is the foundation that sets the stage for the automation of work. It encompasses the structural design of the software and plays a crucial role in ensuring that the software functions smoothly and efficiently. (Spacey, J., 2018).

System architecture is the process of defining and designing the structure and behaviour of software systems that automate work. It involves identifying the components of a system, their relationships, and how they interact to achieve specific goals.

System architecture provides a blueprint for building maintainable, scalable, reliable, and efficient complex software systems. It encompasses the fundamental organisation of a system, the mapping of functionality onto hardware and software components, an allocated arrangement of physical elements, strategic decisions, detailed descriptions, and representation refinement.

A system architecture is a conceptual model that defines a system's structure, behaviour, and other views and serves as a formal description and representation of a system, organised in a way that supports reasoning about its components, interrelationships, and functionality. (Parker, J., 2023).

4.1 Importance of system architecture

System architecture is important because it explains the system's structure and guides design decisions. (No Author, 2024).

The three key features of important software architecture, based on Bass and his colleagues are listed below

1. Enabling communication between all parties, such as stakeholders interested in building this system.
2. Highlights early design decisions that will profoundly impact all software engineering approach decisions as important to the ultimate success of systems as an operational entity.
3. Capture insights into how a system is structured in a relatively small intellectually graspable model of how the system is structured and, how their components work together.

The design model and architecture pattern such as genres, styles and patterns contained within are transferable.

Which is applicable to the design of other systems and represents a set of abstractions that enable to describe and prediction of the architecture interpreted in many ways. (Pressman, S., 2010).

4.2 Architecture Description of MVC

The architectural description is a set of work products that reflects different views of the system.

MVC stands for Model-View-Controller, a software design pattern that separates an application into three main components. The Model component handles data-related logic that the user works with, including adding or retrieving data from the database. The View component is responsible for all the user interface logic of the application, generating a user interface for the user. The Controller component acts as an intermediary between the Model and View components, enabling communication between them. The main goal of the MVC architecture is to promote organised programming by separating functionality, logic, and the interface of an application. It also allows multiple developers to work on the same project, making it scalable and extensible. Popular web frameworks that use MVC include Ruby on Rails, Express, Backbone, and Angular. (Patel, D., 2019).

MVC, or Model-View-Controller, is a web development architecture that offers several benefits. One of its key features is the ability to separate business logic, user interface, and input logic, which makes it easier to manage and maintain different aspects of an application.

Additionally, with full control over HTML and URLs, designing a web application architecture that fits the project's specific needs is easy. The URL-mapping component helps create understandable and easy-to-search URLs, improving the application's navigation and performance.

Finally, MVC supports test-driven development, which allows developers to write tests before writing the actual code, ensuring that the code functions correctly from the start. (Geeks Community, 2023).

Benefits of using MVC

1. Easy to organise large-sized web applications. As this segregate the code among three levels, it's easier to organise and find specific potions of code quickly.
2. It supports AMI (Asynchronous Method Invocation), allowing faster loading of applications.
3. Easily modifiable, MVC simplifies adding/updating new views and promotes flexibility and scalability by allowing easy modification of the entire application without affecting the entire architecture.
4. Faster development process. This allows developers to work on the same project simultaneously to easily implement business logic, accelerating the development process.
5. Easy to plan, maintain and arrange ideas of the code.
6. Return data without formatting; MVC allows components to be reused with the interface.
7. Support Test-driven development, allowing easier debugging on a larger scale.
8. Multiple views limit code duplication by separating data and business logic.
9. SEO-Friendly platform, MVC hugely generates more visitors to the application.

The MVC design pattern is an excellent approach for developing WEB applications, and it's a widely used system architecture style. This school performance monitoring application also aims to be a web-based application and integration of MVC as the architectural pattern was a seamless choice. (Geeks Community, 2023).

The decision to implement the Model-View-Controller (MVC) architecture was made due to the project's requirements for a framework that can not only meet the current needs but also provide the foundation for future scalability and adaptability. The separation of concerns in MVC provides a clear structure, making it easier to understand and modify the application as required.

The decision to adopt the Model-View-Controller (MVC) architecture was a deliberate and well-informed choice, taking into consideration its established advantages such as simplified maintenance, enhanced code reusability, and decreased code intricacy. These benefits are in line with the objectives and specifications of the software system, making MVC a fitting and effective architectural solution for the project.

4.3 MVC Diagram of school performance monitoring application.

The image below illustrates the MCV Architecture diagram of the school performance monitoring application.

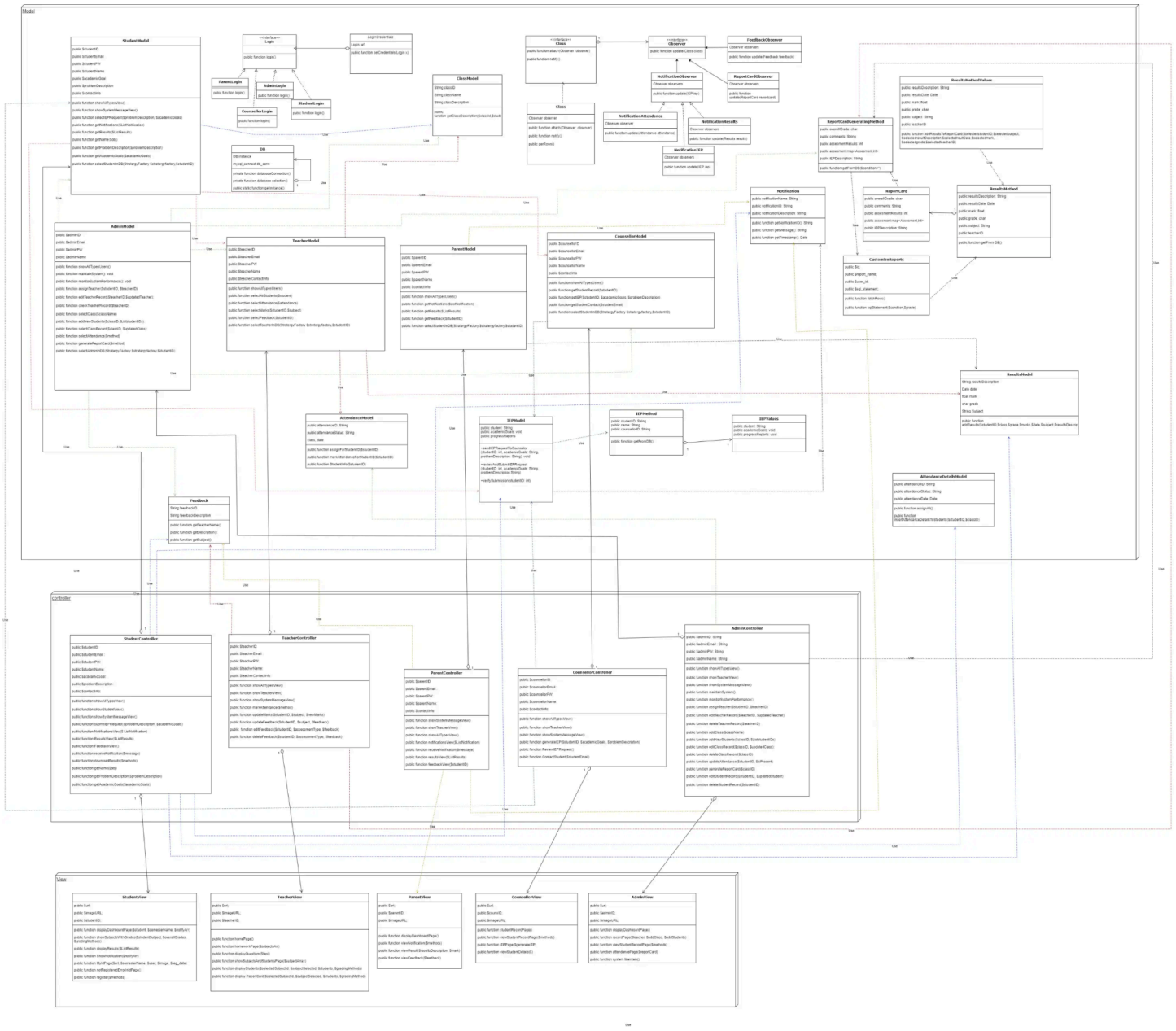

Figure 3: MVC Architecture of Students performance monitoring application.

The Model-View-Controller (MVC) architectural pattern is commonly used in software development to separate an application into three main logical components: Model, View, and Controller.

In the context of a performance monitoring application, an MVC diagram given above can be used to illustrate how the different components of the application interact with each other to monitor the system's performance. The diagram can help identify potential bottlenecks, optimise system performance, and ensure the system runs smoothly. (Berry, D., 2013)

Let's consider for this system, an MVC diagram of a performance monitoring application shows how the Model component collects data from various sources, such as server logs or application metrics and databases, and processes the data to generate performance reports (IEP).

In a student performance application, the MVC components can be described as follows**:**

1. **Model:**

   Collects and stores data related to students such as student ID, email, password, name, academic goals, challenges, attendance, and results of other relevant functions to access the method classes in the model class. This MVC Diagram of student's performance monitoring application includes user model classes that define the attributes and behaviours of the users, method classes that encapsulate the logic and operations performed by the user model, and relevant method value classes that hold the output values returned by the method classes. Additionally, the diagram features interface classes that define the interaction between the user and the application, ensuring a smooth and user-friendly experience.

2. **View:**

   Displays information related to the user's role, the user will be able to view this information in a user-friendly format, enabling them to monitor and support the student's progress. This MVC diagram of a student's performance monitoring application consists of relevant user view classes, In order to maintain security and confidentiality, the views presented to each user will vary depending on their assigned role. This ensures that each user is only able to access the information that is necessary for them to perform their job duties.

3. **Controller:**

   Facilitates the interaction between the user and the system, managing the flow of data between the model and view components. The controller class associated with each user (parent, teacher, counsellor, admin) retrieves information essential to their respective functions and allows them to access the relevant database to perform their tasks efficiently. Controller

classes act as the bridge between view and model classes connected through aggregation and simple association relationships.

In summary, an MVC diagram of a performance monitoring application can help illustrate how the application's different components work together to monitor the system's performance. By providing a clear and concise representation of the system, the diagram can help to identify potential issues and optimise system performance.

5. Testing of design - Architecture

5.1 Function testing of MVC Architecture of students performance monitoring application.

To test the architecture of the Student's performance monitoring application the functional testing methodology is followed Functional Testing. Functional Testing is a type of software testing in which the software application is tested against its functional requirements. It focuses on verifying that the software application functions as expected and meets the desired quality standards.

The Student Model of Student Performance monitoring application collects data from Students such as student ID, email, password, name, academic goals, the challenges they are facing, and their contact details.

The model saves all the relevant information in a Database(DB), enabling showing all types of views available for students, showing system messages, and selection of IEP Request by the counsellor model, getting name results for generating report card and updating attendance, and getting problem descriptions and academic goals to generate IEP through IEP Model class.

Making possible all the functionalities by calling out the student database through the student model.

The admin model is a comprehensive tool that collects and organizes important data related to admin IDs, emails, passwords, and names. This enables the admin to effectively manage and monitor all types of admin users while maintaining optimal system performance. Admins can easily assign, edit, and check teacher and teacher records by connecting to the relevant classes.

Furthermore, the Admin class is equipped with the capability to generate report cards by accessing the method in the model class. All of these functions are made possible through the admin's access to the designated database.

The Teacher model is an essential component that collects important data related to teacher IDs, emails, passwords, names, and contact information. This enables the system to effectively access all types of teacher users and select all the students. Moreover, the Teacher model is equipped with the

capability to mark the students' attendance by selecting the attendance option and accessing the method in the model class.

In addition, the Teacher model has the capability to select the marks option and generate report cards for students by accessing the generate report model. The Teacher model is also capable of adding feedback to each student by accessing the feedback class in the model. All of these functions are made possible by accessing all the relevant information in the Teacher database within the database.

The parent model is capable of retrieving information about parent ID, parent email, parent password and parent name and their contact information, the parent model is also capable of showing all types of parent users. parent model has functionalities such as getting notifications through accessing the notification class in the student model. The parent model also accesses get feedback function through the class feedback in the model.

parent model is also capable of accessing all the relevant information parent database in order to complete the functionalities.

Counsellor model collects the counsellor's ID,counsellor email, counsellor password and their name and contact information. counsellor model can show all types of counsellors available in the system.

student model can retrieve student records through the model class student record class, counsellor model can also retrieve IEP through the IEP class in the model class. The counsellor model is capable of getting student information using the student model.

The counsellor model is also capable of accessing all relevant information retrieved through the counsellor within the database.

The Login interface in the model is a crucial component that facilitates the login process for various users such as parents, counsellors, admins, and students. Each user is represented as a separate class that inherits properties from the Login class.

The Login class is aggregated with the login interface since both classes depend on each other to perform their functions. This ensures that the login process is smooth and efficient and allows for the seamless functioning of the entire system.

Moreover, the Login interface is designed to be flexible and adaptable, which means that each user class can exist independently while still being able to interact with the login process as and when required. This ensures that the system remains efficient and effective, even as the number of users and classes grows over time.

Database (DB) class retrieves private and public functions to connect various classes to retrieve information by calling the class. This class represents the whole database of the system.

The class model is used by other classes within the model to get class description details.

Class interface is specialised by a class and class interface is connected to feedback observer, report card observer and notification observer.

The notification observer inherits notification attendance, results and IEP therefore, the relevant user will be able to receive the relevant notification from the system.

Attendance model and attendance details model classes enable to marking of the attendance of the students by the teacher controller and updated by the admin controller.

The system's results model in results model accessed through relevant stakeholders will enable to collection of information related to results and store it.

Customize report class enables the generated report card method to use it and create customised reports according to the admin's preferences.

IEP Model uses the IEP method class and retrieves the form from the system database. The IEP value class collects relevant attributes that are required to generate an IEP. When these two classes are aggregated, it allows for an IEP method call to access multiple values simultaneously. This feature enables the IEP generation of multiple students to occur efficiently and effectively, making the process more streamlined.

The View component of the student's performance monitoring application will display these IEP reports to the users in a user-friendly format.

The Student View component is responsible for displaying various information related to the student's academic performance. It retrieves image URLs and displays the Dashboard, along with the subject grades, results, and any notifications received by the user. In addition, it also allows users to view their ID page and register for courses. If a user is not registered, the component is capable of displaying an error page.

The Teacher View Class Component allows teachers to retrieve URLs and Image URLs and display them on the homepage. With this component, teachers can also set up questions for their students and view the subjects and respective students in one convenient location. Additionally, teachers can easily view the progress of individual students and their respective classes, and even access report cards to track their academic achievements. This tool is a valuable asset for any teacher looking to efficiently manage their classroom and help their students succeed.

The Parent View component of the performance monitoring application is an essential feature that enables parents to view their child's academic progress. It is capable of collecting relevant URLs, student IDs, and image URLs from the system database and displaying the dashboard in a user-friendly format. Parents can view their child's subject grades, and academic results, and receive notifications, along with any feedback added by the teacher in the respective sections. This feature helps parents to monitor their child's academic performance and stay updated on their progress. Any enhancements to

this component can be made to improve its usability and functionality and ensure that parents have the necessary information to support their child's education.

The View component of the Counsellor allows the counsellor to easily access URLs and their respective counsellor ID and image URLs, thus enabling them to view the appropriate student's student record page. Additionally, the Counsellor can also view the IEP page assigned to each student upon request. Furthermore, the Counsellor can also view the student details page to obtain the necessary information to generate an IEP. This feature simplifies the process of managing student records and helps Counsellors provide better support to their students. while the Controller component might handle user input and manage the data flow between the Model and View components.

The view component of admin is capable of retrieving e important information such as URLs and admin IDs, as well as their respective URLs. This information helps to enable the display of the dashboard and student record pages, attendance pages, and system maintenance pages. It plays a crucial role in ensuring the smooth and efficient functioning of the system.

The Controller component handles user input and manages the data flow between the Model and View components.

The controller class associated with each user facilitates the retrieval of attributes that are essential to the controller's functions. Additionally, it seamlessly manages the interaction between the model and view components. The controller class achieves this by accessing relevant class methods and classes, model classes within the model component.

In software development, controller classes form the backbone of an application. These classes are intricately linked together through aggregation and simple relationships. One of the key advantages of using controller classes is their ability to retrieve multiple data at the same time through a single function, streamlining the entire process. Furthermore, all relevant simple relationships between classes are established using the appropriate model/model class method or method value class or interface class, ensuring seamless and efficient usage between them.

In conclusion, testing is a critical aspect of the software development process that helps to ensure that the application works as intended and meets the needs of its intended users. By testing each individual function or feature of the application, software developers can identify and address any issues or bugs before the application is released to the public. This helps to ensure that the application is reliable, efficient, and user-friendly and that it provides the intended value to its users.

5.2 Overview of the System's Components and Their Functions

The MVC pattern of student performance monitoring systems touches on several aspects of software design that are important for creating high-quality software applications, including extensibility, flexibility, performance, reusability, and testability.

1. Extensibility refers to the ability of the software application to be easily extended or modified as new features or requirements are added. In the context of the student's performance monitoring application, the use of an MVC (Model-View-Controller) architecture can help to facilitate extensibility by separating the application into distinct components that can be modified or extended without affecting the other components.

2. Flexibility refers to the ability of the software application to adapt to changing requirements or conditions. The use of a modular design, such as an MVC architecture, can help to facilitate flexibility by allowing developers to modify or replace individual components as needed without affecting the rest of the application. This application Uses machine learning algorithms to monitor students' academic progress and is a flexible way to inform instructors about students at risk of poor performance.

3. Performance refers to the speed and efficiency of the software application. The paragraph mentions the importance of streamlining data retrieval, using appropriate class methods, and establishing simple relationships between classes to ensure efficient usage. These design considerations can optimise and help the performance of the application. This approach can help instructors take measures to offer additional support to struggling students promptly. The prediction model can be transformed into a clear shape to make it easy for the instructor to prepare the necessary precautionary procedures An instance of flexibility is the application of process flexibility in process management. It pertains to how an operation adapts to external factors, such as changes in supply or demand. If utilised efficiently, process flexibility can reduce the cost of external factors that affect student performance.

4. Reusability refers to the ability of software components to be reused in other applications or contexts. The paragraph mentions the use of controller classes, which can retrieve multiple data simultaneously through a single function, streamlining the entire process. This can help to improve reusability by creating modular components that can be used in other applications. The implementation of interface inheritance for class reusability, along with a common login, greatly enhances the reusability of the code. By allowing other user login classes to inherit the properties of the interface, the code becomes more streamlined and efficient. This approach also promotes code modularity and makes it easier for developers to add new features or modify existing ones.

5. Testability refers to the ease with which software components can be tested and verified for correctness and functionality. The student's performance monitoring application focuses on

function testing technique, the use of an MVC architecture can help to facilitate testability by separating the application into distinct components that can be tested independently considering each method's function.

6. Considerations of software engineering principles.

Software engineering is the process of creating software applications that meet the specific needs of users. To do this, software engineers must follow certain guidelines to ensure the software's success.

Firstly, it's important to conduct a thorough analysis of user requirements. This ensures that the software is designed to meet the users' needs. Secondly, the software design should be kept simple to make it easier for users to understand and use. A modular approach to developing software functionalities can also help streamline the design process, making it easier to handle workloads, reuse code, and debug errors.

Additionally, software engineers should continuously validate the software to ensure it's working correctly and make sure it can scale to meet the changing needs of users. By following these principles, software engineering can lead to the creation of high-quality software that meets the users' needs.

1. Rigour and formality

   Rigor and formality are two important principles in software engineering that ensures quality of software development. Rigour refers to strict adherence to rules and to strict adherence to the software development process.

   The IEP request system upholds a high level of rigor through its strict data validation and adherence to the MVC architecture. To ensure formality, the system follows formal specifications, documentation, and quality assurance processes, which contributes to an overall improvement in system quality and reliability.

2. Separation of Concerns

   Software engineering follows the design principle of separation of concerns, wherein a computer program is separated into different sections, each addressing a particular concern or set of information that affects the program's code. A program that demonstrates good separation of concerns is known as a modular program.

   In the IEP request system, the principle of "Separation of Concerns" is applied by distinctly organizing and isolating different aspects such as user interface, application logic, and data handling, ensuring clarity, maintainability, and flexibility in the software architecture.

3. Modularity

   Modularity and separation of concerns are crucial concepts in software engineering. They can be achieved through encapsulation, which involves hiding information within a specific section of code that has a clear interface. Modularity is a design principle that involves dividing a software system into separate functional units called modules. Each module has a well-defined interface and performs a specific task. Modularity helps to reduce the complexity of software systems, making them easier to understand, maintain, and modify.

This can be implemented by implemented by organizing distinct components for user interface, application logic, and data handling, ensuring clarity and maintainability in the software architecture.

4. Abstraction

   Is a fundamental design principle that involves selectively hiding certain details of a process or artifact while emphasizing other aspects, details, or underlying structures. The goal of abstraction is to simplify complex systems by breaking them down into smaller, more manageable components, making them easier to understand, modify, and maintain. This concept plays a critical role in modern software development, allowing software engineers to create more efficient, scalable, and maintainable applications.

   Abstraction is a technique used in system design to simplify complex processes and structures. For instance, in the Model-View-Controller (MVC) architecture, abstraction is employed to separate data (Model), user interface (View), and application logic (Controller). This approach helps to create a more modular and comprehensible design by focusing on essential components and interactions while concealing unnecessary details. By using abstraction, the system becomes easier to understand and maintain.

5. Anticipation of change

   In the field of software engineering, the principle of anticipating change involves predicting the areas where changes are likely to occur and preparing for those changes during the design phase. This principle is crucial to developing software that can adapt to change and helps to ensure that the software development process is methodical, dependable, and produces high-quality results.

   The IEP request system is designed to apply the principle of anticipating change to the software engineering process. During the design phase, the team carefully identifies areas where changes are likely to occur in the future. This proactive approach ensures that the system is designed in a way that can easily accommodate modifications or updates without major disruptions. By anticipating potential changes and incorporating flexibility into the design, the system is better equipped to adapt to evolving requirements, technologies, or user needs. This contributes to the system's robustness and longevity, ensuring that it continues to meet the specific needs of users and is adaptable to changing requirements.

Software engineering principles are crucial for the development of high-quality software. The principles of rigour and formality, separation of concerns, modularity, abstraction, and anticipation of change are essential for the success of software applications. By following these principles, software engineers can create software that meets the specific needs of users and is adaptable to changing requirements. Therefore, it is important for software engineers to adopt these principles in their software development processes.

7. Use of the modern operating systems and libraries

As a Student Performance Monitoring System, it is crucial to make use of operating system functionalities to ensure smooth, secure, and efficient operation. Each of these functionalities can contribute to the effective functioning of such a system. Here's how:

1. Process Management:

   Efficiently manage multiple processes related to student performance tracking, data analysis, and reporting. Allocate CPU and memory resources optimally to handle concurrent tasks such as generating reports, updating student records, and analysing performance trends.

2. Memory Management:

   Allocate and deallocate memory for storing student data, assessment results, and performance metrics. Ensure efficient memory usage to accommodate the growing database of student records while maintaining system responsiveness.

3. File Management:

   Organize and manage files related to student profiles, academic records, and performance reports. Provide secure and efficient access to directories and files for administrators, teachers, and other stakeholders involved in student monitoring.

4. Network Management:

   Facilitate network communication for real-time access to student data, especially in scenarios where multiple users or departments access the system simultaneously. Ensure secure data transmission over the network, supporting features like remote access for teachers, administrators, and parents.

5. Security and Protection:

   Implement robust security measures to safeguard sensitive student information and performance data. Utilise encryption, user authentication, and access control mechanisms to prevent unauthorised access and protect the privacy of student records.

6. Error Handling:

   Detect and handle errors that may arise during data processing, analysis, or reporting. Provide error logs and debugging tools to assist administrators and developers in identifying and resolving issues promptly

7. User Interface Management:

   Support the graphical user interface (GUI) of the Student Performance Monitoring System, ensuring a user-friendly experience for teachers, administrators, and other users. Manage input and output operations related to user interactions, data entry, and result presentation.

8. Concurrency Control:

Implement concurrency control mechanisms to manage simultaneous access to student records and prevent data inconsistencies. Ensure that updates to student information or performance metrics are handled in a controlled and synchronised manner.

9. Task Scheduling:

Schedule and prioritise system tasks such as automated report generation, data backups, and system maintenance. Optimise task scheduling to avoid system overload during peak usage times.

10. Backup and Recovery:

Implement backup and recovery mechanisms to protect against data loss or system failures. Regularly back up student records, assessment data, and system configurations to ensure quick recovery in unforeseen events.

In summary, leveraging these operating system functionalities is essential for the effective functioning of a Student Performance Monitoring System. They ensure data accuracy, system reliability, and a positive user experience.

Reference

1. Bass, L., Clements, P. and Kazman, R. (2003). *Software Architecture in Practice*, 2nd ed. USA: Pearson Education.
2. Beck, K. and Cunningham, W. (1989). A Laboratory for Teaching Object-Oriented Thinking: CRC Cards. *OOPSLA'89 Conference Proceedings,* 24(10), (No Page No.). Available from https://c2.com/doc/oopsla89/paper.html#cards. [Accessed 28 December 2023].
3. Berry, D. (2013). .NET: Building Performance Metrics into ASP.NET MVC Applications. *Redgate.* Available from https://www.red-gate.com/simple-talk/development/dotnet-development/building-performance-metrics-into-asp-net-mvc-applications/. [Accessed 30 December 2023].
4. Coursera Staff. (2023). Agile Project Management: What is it and Why does it Matter. *Coursera.* Available from https://www.coursera.org/articles/agile-project-management. [Accessed 26 December 2023].
5. Geeks Community. (2023). Benefit of Using MVC. *Geeks for Geeks.* Available from https://www.geeksforgeeks.org/benefit-of-using-mvc/. [Accessed 30 December 2023].
6. Geeks Community. (2023). MVC Framework Introduction. *Geeks for Geeks.* Available from https://www.geeksforgeeks.org/mvc-framework-introduction/. [Accessed 30 December 2023].
7. Inettutor. (2021). Student Academic Performance Tracking and Monitoring System. *iNetTutor.com.* Available from https://www.inettutor.com/programming-tutorial/bootstrap/student-academic-performance-tracking-and-monitoring-system/. [Accessed 01 January 2023].
8. Kasse Initiatives. (2004). Processes for Engineering a System EIA-632. *SE Tutorial Processes for Engr Sys - 1.* Available from https://www.acqnotes.com/Attachments/Processes%20for%20Engineering%20a%20System%20EIA%20-%20632.pdf. [Accessed 01 January 2024].
9. Khan, I et al. (2021). An Artificial Intelligence Approach to Monitor Student Performance and Devise Preventive Measures. *Smart Learning Environments,* 8(17). Available from https://slejournal.springeropen.com/articles/10.1186/s40561-021-00161-y. [Accessed 01 January 2024].
10. NEA. (2021). Mitigation Strategies for Safe In-Person Learning. *National Education Association.* Available from https://www.nea.org/sites/default/files/2021-03/Mitigation%20Strategies%20for%20Safe%20In-Person%20Learning.pdf. [Accessed 29 December 2023].
11. Nehra, M. (2022). 6 Stages of Agile Development Lifecycle. *Decipherzone.* Available from https://www.decipherzone.com/blog-detail/agile-development-lifecycle. [Accessed 25 December 2023].
12. Nishadha, S. (2022). UML Class Diagram Relationships Explained with Examples. *Creately.* Available from https://creately.com/guides/class-diagram-relationships/. [Accessed 28 December 2023].
13. No Author. (2022). IT: What are the Six Types of Relationships in UML Class Diagrams. *Visual Paradigm Online.* Available from

https://blog.visual-paradigm.com/what-are-the-six-types-of-relationships-in-uml-class-diagrams/#disqus_thread. [Accessed 01 January 2024].

14. No Author. (2023). Database: Class Diagram vs. Entity-Relationship Diagram (ERD): A Comparative Guide. *Visual Paradigm.* Available from https://guides.visual-paradigm.com/class-diagram-vs-entity-relationship-diagram-erd-a-comparative-guide/. [Accessed 29 December 2023].
15. No Author. (2023). Process: What is Data Flow Testing? Application, Examples and Strategies. *Testbytes.* Available from https://www.testbytes.net/blog/data-flow-testing/. [Accessed 28 December 2023].
16. No Author. 2024. System Architecture - Detailed Explanation. *Interview Bit.* Available from https://www.interviewbit.com/blog/system-architecture/. [Accessed 8 January 2024].
17. Parker, J. (2023). Why is System Architecture Important?. *Architecture.* Available from https://www.architecturemaker.com/why-is-system-architecture-important/. [Accessed 29 December 2023].
18. Patel, D. (2019). An Introduction to MVC Architecture: A Web Developer's Point of View. *DZone.* Available from https://dzone.com/articles/introduction-to-mvc-architecture-web-developer-poi. [Accessed 30 December 2023].
19. Pressman, S. (2010). *Software Engineering: A Practitioner's Approach,* 7th ed. New York: McGraw-Hill.
20. Spacey, J. (2018). What is System Architecture?. *Simplicable.* Available from https://simplicable.com/IT/system-architecture-definition. [Accessed 29 December 2023].
21. Siewert, A. (2023). Data Flow Testing: What is Data Flow Testing? DFT Coverage, Strategies and More. *Lambdatest.* Available from https://www.lambdatest.com/learning-hub/data-flow-testing. [Accessed 29 December 2023].
22. Soumya, MD. and Krishnamoorthy, S. (2021). Student Performance Prediction, Risk Analysis, and Feedback Based on Context-Bound Cognitive Skill Scores. *Education and Information Technologies,* 27(2022), 3981-4005. Available from https://link.springer.com/article/10.1007/s10639-021-10738-2. [Accessed 01 January 2024].
23. Wells, D. (2009). CRC Cards. *Extreme Programming.* Available from http://www.extremeprogramming.org/rules/crccards.html. [Accessed 25 December 2023].

Appendices

Appendix A

Drive links for the UML Class Diagram and MVC Architecture Diagram are given below.

- Class Diagram:
  https://drive.google.com/file/d/1-Vw_BoGB9ZFJlStkPR1edzIjzeR4Vnmy/view?usp=sharing

- MVC Architecture:
  https://drive.google.com/file/d/1GCnmtca4MMMtohd669FTi2rdI8aOTYSB/view?usp=sharing