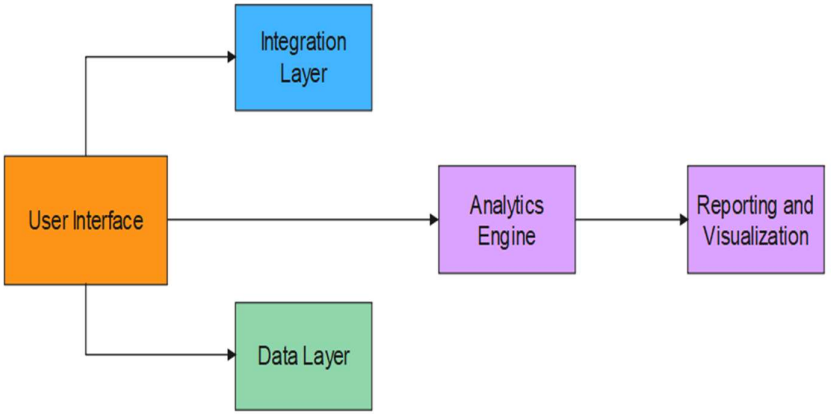
Project Design Phase-II
Technology Stack (Architecture & Stack)

Date	21 Oct 2023
Team ID	NM2023TMID05094
Project Name	Unleashing the Potential of Our Youth: A Student Performance Analysis
Maximum Marks	4 Marks

Technical Architecture:

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2



Guidelines:

1. **User Interfaces**: Design intuitive and user-friendly interfaces for easy system interaction.
2. **Integration Layer**: Develop robust APIs for seamless integration with external systems.
3. **Data Layer**: Implement a scalable and secure data storage solution.
4. **Analytics Engine**: Utilize advanced analytics techniques to derive actionable insights from data.
5. **Reporting and Visualization**: Create comprehensive and visually appealing reports for effective data presentation.

Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interfaces	The user interfaces component includes the graphical interfaces through which users interact with the system. It enables users to input data, view reports and visualizations, and access system functionalities.	HTML, CSS, JavaScript, React.js
2.	Integration Layer	The integration layer facilitates the communication and integration between different components and external systems. It enables data exchange and interoperability between the system and external data sources or APIs.	RESTful APIs, JSON, XML, SOAP
3.	Data Layer	The data layer component manages the storage and retrieval of data within the system. It includes databases and data management systems that store and organize student information, performance data, and other relevant data.	Relational Database Management System (e.g., MySQL, PostgreSQL), NoSQL Database (e.g., MongoDB), Data Warehousing
4.	Analytics Layer	The analytics layer component handles the processing and analysis of data to derive meaningful insights and performance metrics. It involves statistical analysis, machine learning algorithms, and data visualization techniques.	Python, R, TensorFlow, PyTorch, Tableau, Power BI
5.	SIS (Student Information System)	The SIS component manages student-related data such as personal information, enrollment, attendance, and grades. It serves as a central repository for student data.	Student Information System software (e.g., PowerSchool, Infinite Campus)
6.	Learning Management System (LMS)	The LMS component facilitates online learning and course management. It includes features such as course content delivery, assignments, quizzes, and tracking student progress.	Learning Management System software (e.g., Moodle, Canvas)
7.	External Data Sources	External data sources refer to any external systems, databases, or APIs that provide additional data relevant to the student performance analysis, such as educational standards, research databases, or online resources.	Various external systems and APIs, specific to the data sources being utilized

8.	Analysis Engine	The analysis engine component performs advanced analysis and modeling on the collected data to generate insights, predictions, and recommendations. It applies statistical algorithms, machine learning techniques, and data mining methods.	Python, R, scikit-learn, TensorFlow, PyTorch
9.	Reporting and Visualization	The reporting and visualization component generates reports, charts, and visual representations of the analyzed data to present performance metrics, trends, and comparisons in an understandable format.	Tableau, Power BI, D3.js, matplotlib, ggplot
10.	Collaboration and Support	The collaboration and support component provides communication tools and support features for stakeholders, allowing them to interact, share information, and seek assistance within the system.	Chat or messaging systems, discussion forums, helpdesk software
11.	Stakeholders	The stakeholder's component represents the users and individuals involved in the system, including administrators, teachers, parents, and students who interact with the system and benefit from the student performance analysis.	User management system, role-based access control

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	The system should leverage open-source frameworks and technologies to benefit from community-driven development, flexibility, and cost-effectiveness.	Examples include Django, Flask, Ruby on Rails, Node.js, AngularJS, React.js.
2.	Security Implementations	The system should have robust security measures in place, such as encryption, access control, and secure authentication, to protect sensitive data and ensure the privacy of users.	Encryption algorithms (AES, RSA), Secure Socket Layer (SSL), Transport Layer Security (TLS), OAuth, JSON Web Tokens (JWT), firewalls, intrusion detection systems.
3.	Scalable Architecture	The system should be designed with a scalable architecture that can handle increased workloads, accommodate growing user bases, and easily adapt to changing needs without sacrificing performance or functionality.	Microservices architecture, containerization (Docker, Kubernetes), horizontal scaling, load balancing, cloud computing platforms (AWS, Google Cloud, Microsoft Azure).
4.	Availability	The system should be highly available, with minimal downtime and disruptions, ensuring that users can access and use it whenever needed.	Redundancy, failover mechanisms, load balancing, distributed systems, fault-tolerant infrastructure.
5.	Performance	The system should be optimized for efficient performance, with fast response times, minimal latency, and the ability to handle large volumes of data processing and analysis.	Caching mechanisms (Redis, Memcached), asynchronous processing, performance monitoring tools, optimization techniques (code optimization, database indexing, query optimization).