# Assignment2

**Objective:**

•Design an application for student management system that includes all the functionalities and concepts learned in this module
•Follow the instructions to create solutions according to the concepts taught in each module

**Problem Statement:**

A popular education society "EduExcellence" is setting up a new school in the city. They want a "School Management Software" that should match the high standards of the school that they aspire to build. They have hired you to provide them with the software that should enable them to manage all aspects of running a school successfully, but they would want the application to grow gradually with the growth of the new school.

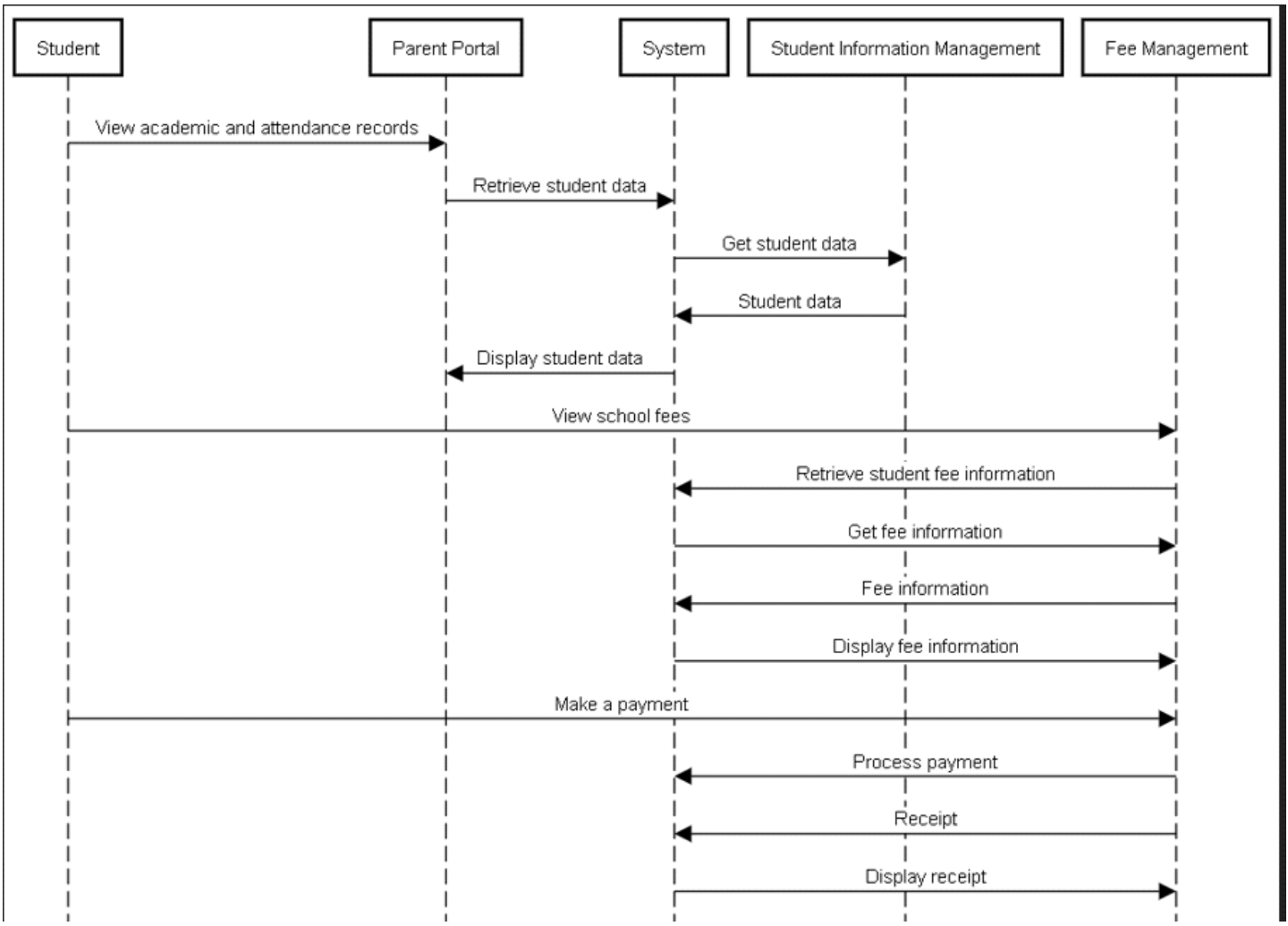
Now that the basic design is ready, add to the design, critical non-functional aspects and mention the tools that you would choose.
1.It should be highly scalable.
2.It should be highly available.
3.It should be resilient

***EduExcellence:***

The system should be scalable, flexible, and easy to use, and should incorporate all the necessary functionalities for managing a school successfully. The key features of the system will include:

1. **Student Onboarding**:
   A module for registering new students, including capturing student data, generating unique student IDs, and validating student information.
2. **Student Information Management**:
   A module for storing and managing student data, including personal information, academic records, attendance, and disciplinary history.
3. **Faculty Information Management**:
   A module for storing and managing faculty data, including personal information, academic records, and employment history.
4. **Curriculum Management**:
   A module for creating and managing academic programs, courses, and classes, including scheduling and course assignments.
5. **Assessment Management**:
   A module for managing student assessments, including grading, progress tracking, and report generation.
6. **Fee Management**:
   A module for managing school fees, including fee structures, invoicing, payment processing, and receipt generation.
7. **Library Management:**
   A module for managing school library resources, including book inventory, circulation, and borrowing.
8. **Transport Management**:
   A module for managing school transportation, including route planning, scheduling, and vehicle maintenance.
9. **Parent Portal**:
   A module for providing parents with access to their child's academic and attendance records, fee information, and school news and

events.




To ensure that the School Management Software is highly scalable, highly available, and resilient, the following non-functional aspects should be considered:

- **Scalability**: The software should be designed to handle increasing loads and growing numbers of students, teachers, and administrative staff. This can be achieved by implementing a microservices architecture, where each functionality is separated into smaller, independent services that can be easily scaled up or down based on demand. In addition, cloud-based hosting services like Amazon Web Services (AWS) or Microsoft Azure can be used to provide elastic scalability.

- **Availability**: The software should be available 24/7 without any downtime. This can be ensured by using a load balancer to distribute incoming requests among multiple servers, each of which runs multiple instances of the application. In addition, the software should be designed to handle automatic failover in case of server failure. Tools like Amazon Elastic Load Balancer (ELB) or Microsoft Azure Load Balancer can be used to provide high availability.

- **Resilience**: The software should be able to recover from any failures or disruptions, whether they are caused by hardware, software, or network issues. This can be achieved by implementing disaster recovery and backup mechanisms, such as storing data in multiple locations and having

redundant copies of critical components. Tools like Amazon Simple Storage Service (S3) or Microsoft Azure Backup can be used to provide resilience.

Tools that can be used to implement the non-functional aspects of the software include:

- **Amazon Web Services (AWS)**: AWS provides a wide range of cloud-based services that can be used to build scalable, available, and resilient applications. Services like Amazon Elastic Compute Cloud (EC2), Amazon Simple Queue Service (SQS), and Amazon Relational Database Service (RDS) can be used to implement a microservices architecture, while services like Amazon Elastic Load Balancer (ELB), Amazon Route 53, and Amazon Elastic Block Store (EBS) can be used to provide high availability.

- **Microsoft Azure**: Microsoft Azure is another cloud-based platform that provides a variety of services to build and deploy scalable, available, and resilient applications. Services like Azure App Service, Azure Functions, and Azure Service Bus can be used to implement a microservices architecture, while services like Azure Load Balancer, Azure Traffic Manager, and Azure Storage can be used to provide high availability.

- **Kubernetes**: Kubernetes is an open-source container orchestration system that can be used to manage and scale containerized applications. It provides features like auto-scaling, load balancing, and rolling updates, which can be used to implement a scalable and available architecture. In addition, Kubernetes can be used to implement disaster recovery and backup mechanisms, such as replicating data across multiple clusters.

- **Docker**: Docker is a containerization platform that can be used to package and deploy applications in a lightweight, portable format. It provides features like container orchestration, load balancing, and networking, which can be used to implement a microservices architecture. In addition, Docker can be used to implement backup and disaster recovery mechanisms, such as creating snapshots of containers and storing them in a remote location.