**The Islamia University of Bahawalpur**

**Department of Software Engineering**

**Faculty of Computing**

****

**Software Design Document (SDD)**

**For**

**School Management System**

***By***

**Student Name: Usama Anwer**

**Roll No: F20BSEEN1E02044**

**Section: Evening A**

**Session: Spring 2020 – 2024**

**Supervisor:Dr. Daler Ali Awan**

**Bachelor of Science in Software Engineering**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Description** | **Author** | **Comments** |
| <date> | <Version 1> | <Your Name> | <First Revision> |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Document Approval**

The following Software Design Document (SDD) has been accepted and approved by the following:

|  |  |  |  |
| --- | --- | --- | --- |
| **Signature** | **Printed Name** | **Title** | **Date** |
|  | . | Supervisor, **SENG-4301** | <date> |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

1. Introduction 3

1.1. Purpose 3

1.2. Scope 4

1.3. Overview 4

1.4. Reference Material 4

1.5. Definitions and Acronyms 4

2. System Overview 4

3. System Architecture 4

3.1. Architectural Design 4

3.2. Decomposition Description 5

3.3. Design Rationale 5

4. Data Design 5

4.1. Data Description 5

4.2. Data Dictionary 5

5. Component Design 5

6. Human Interface Design 6

6.1. Overview of User Interface 6

6.2. Screen Images 6

6.3. Screen Objects and Actions 6

7. Requirements Matrix 6

8. Appendices 6

# 1. Introduction

## 1.1. Purpose

## This software design document (SDD) outlines the architecture and system design of the School Management System, a web-based application developed using ASP.NET technology. It aims to provide a comprehensive overview of the system's components, their interactions, and the design decisions made during development. This document serves as a guide for:

## Developers: To ensure a clear understanding of the system's structure and implementation strategies, facilitating efficient development and maintenance.

## Testers: To develop comprehensive test plans and identify potential issues within the system.

## Stakeholders: To provide visibility into the system's design and functionality, ensuring alignment with project goals.

## 1.2. Scope

1. The software product to be produced is a comprehensive School Management system developed using ASP.NET technology.
2. The software will streamline various administrative tasks within schools, including student registration, fee management, attendance tracking, score assignment, learning material uploads, online classes and much more. It will facilitate secure communication channels between administrators, teachers, students, and parents.
3. **Executive-Level Summary:** Our School Management ERP system stands as a transformative solution, digitally enhancing educational institutions. It combine administrative, teaching, and communication modules into a unified platform.
4. **Relevant Benefits**:

* **Streamlined Operations:** Enhances efficiency by simplifying administrative, teaching and other tasks, ensuring a smooth workflow.
* **Data Security:** Implements robust protocols to safeguard institutional data, ensuring integrity and confidentiality.
* **Unified Communication:** Fosters collaborative interactions among administrators, teachers, students, and parents through interconnected communication channels.
* **Financial Transparency:** Provides clarity in financial transactions, offering insights into fee structures and payment histories.
* **Advanced Learning Tools:** Empowers educators and students with cutting-edge tools, fostering an enriched online learning environment.

# 2. System Overview

The School Management System is a comprehensive and integrated platform designed to revolutionize the administrative, educational, and communication processes within educational institutions. This system is meticulously crafted to address the specific challenges faced by schools and provide an efficient, secure, and user-friendly solution for administrators, teachers, students, and parents.

1. **Educational Environment:**

* The system recognizes and adapts to the dynamic nature of educational environments, accommodating varying class structures, academic sessions, and student enrollment changes.

1. **Streamlined Task Management:**

* A primary objective is to streamline administrative tasks, enhancing overall efficiency for school staff. This includes student registration, fee management, and other day-to-day operations.

1. **User-Centric Design:**

* Tailored to distinct user groups, the system defines clear roles and responsibilities for administrators, teachers, students, and parents. Each user experiences a customized interface catering to their specific needs.

1. **Robust Data Security:**

* Prioritizing data security, the system implements robust measures to ensure the confidentiality and integrity of sensitive information. It guards against unauthorized access and data breaches.

1. **Seamless Communication:**

* Facilitating effective communication is a core aspect of the system. Features include announcement boards, query resolution, and complaint registration to foster collaboration among stakeholders.

1. **Educational Content Delivery:**

* Teachers can seamlessly upload learning materials, ensuring students have easy access to relevant educational resources for their classes.

1. **Online Class Experience:**

* The system supports a seamless online class experience for teachers and students. It provides an intuitive interface for participating in virtual learning sessions.

1. **Fee Management Excellence:**

* The system simplifies fee management, allowing administrators to set fee structures, manage class sessions, and collect payments effortlessly. Parents benefit from a transparent and user-friendly payment system.

1. **Attendance and Performance Tracking:**

* Tools for attendance marking and performance tracking empower teachers to assess student progress. Parents can actively engage in monitoring their child's academic achievements.

1. **Query and Complaint Resolution:**

* A dedicated system for registering queries and complaints ensures a prompt and effective resolution process, enhancing overall stakeholder satisfaction.

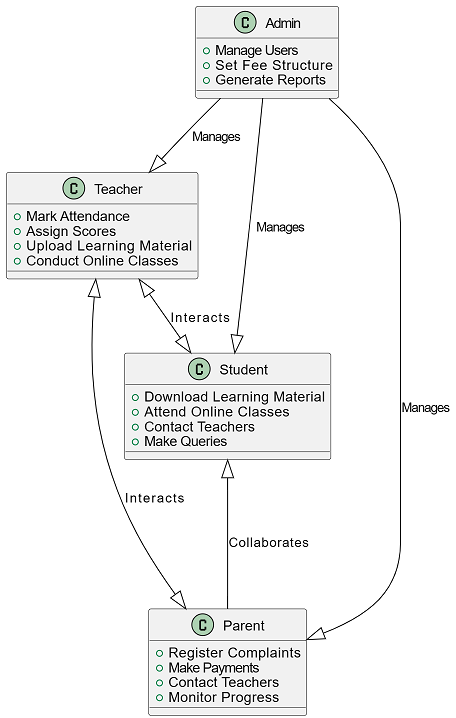
# 3. System Architecture

## 3.1. Architectural Design

1. **Admin Module:**
   * **Responsibilities:**
     + User management (Admin, Teachers, Students, Parents).
     + Class and session management.
     + Fee structure management.
     + Report generation.
2. **Teacher Module:**
   * **Responsibilities:**
     + Marking attendance.
     + Assigning scores to students.
     + Uploading learning materials.
     + Conducting online classes.
3. **Student Module:**
   * **Responsibilities:**
     + Downloading learning materials.
     + Attending online classes.
     + Contacting teachers.
4. **Parent Module:**
   * **Responsibilities:**
     + Registering complaints.
     + Making payments.
     + Contacting teachers.
     + Monitoring students' progress.

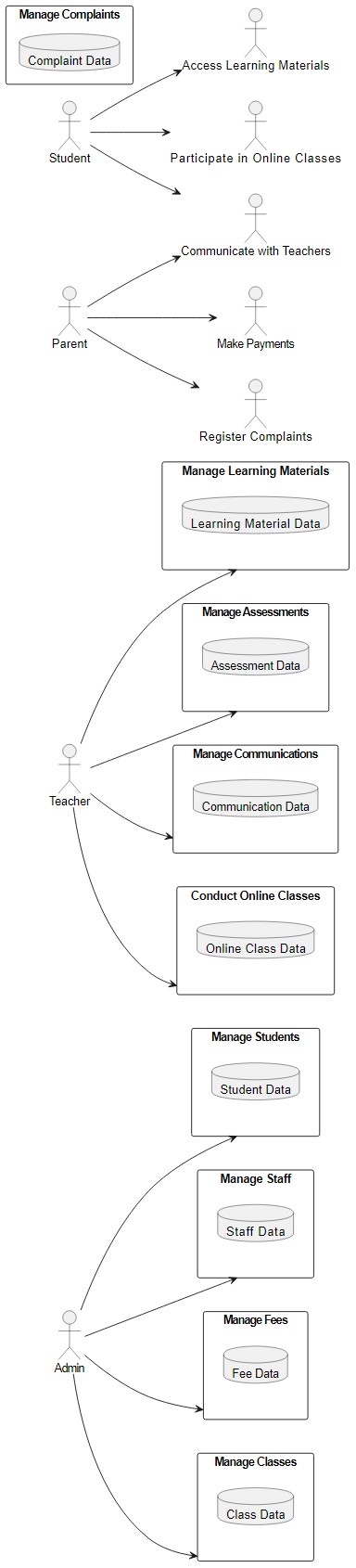
**Subsystem Collaboration:**

* **Admin-Teacher Collaboration:**
  + Admin assigns classes to teachers.
  + Admin sets fee structures that affect teachers and students.
* **Admin-Student Collaboration:**
  + Admin registers students and assigns them to classes.
  + Admin manages class and session information that students access.
* **Admin-Parent Collaboration:**
  + Admin handles complaints registered by parents.
  + Admin manages fee-related information that parents access.
* **Teacher-Student Collaboration:**
  + Teachers mark attendance and assign scores to students.
* **Teacher-Parent Collaboration:**
  + Teachers communicate with parents regarding students' progress.
* **Student-Parent Collaboration:**
  + Parents monitor their children's progress and communicate with teachers
* **Diagram:**



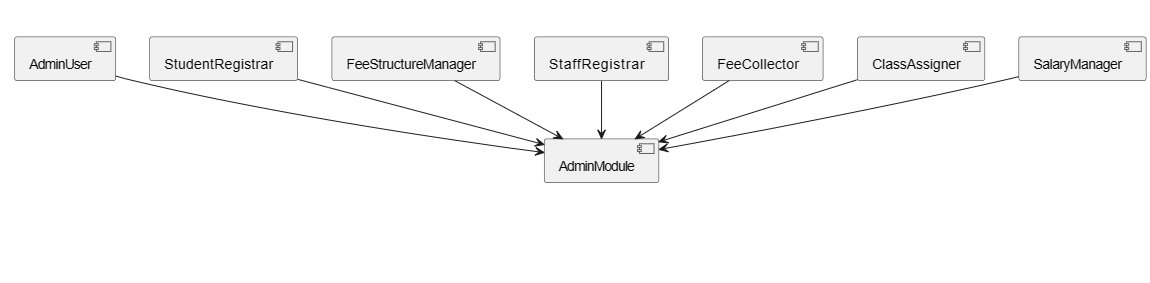
## 3.2. Decomposition Description

* **Top Level Data Flow Diagram:**

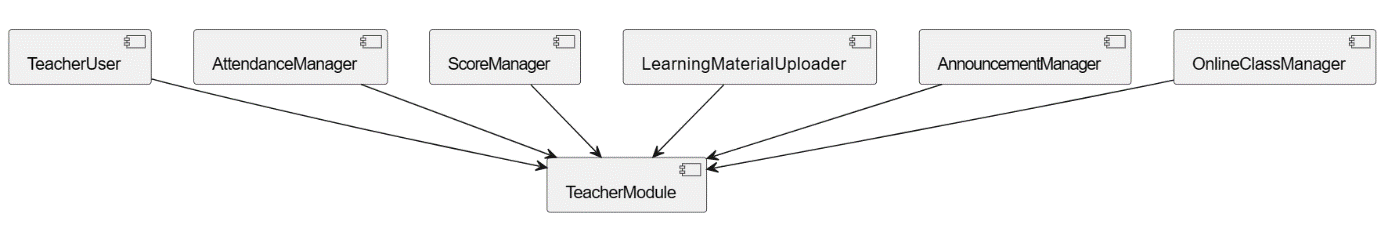
****

**Structural Composition Diagrams:**

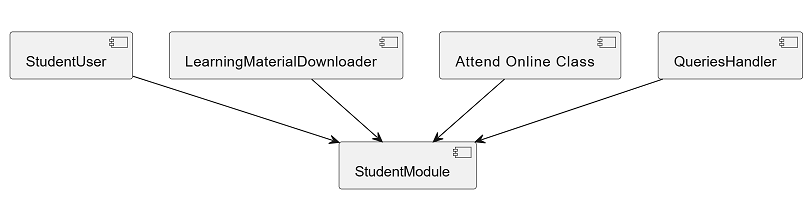
* **Admin Module:**



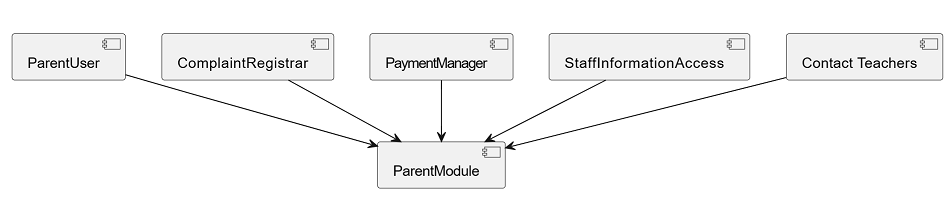
* **Teacher Module:**

****

* **Student Module:**

****

* **Parent Module:**

****

**Figure 2.1: Parent Module**

## 3.3. Design Rationale

The selected architecture for the School Management System is a modular and layered architecture, which includes separate modules for Admin, Teacher, Student, and Parent. Each module encapsulates specific functionalities, promoting a clear division of responsibilities and facilitating easier maintenance and scalability. Here's the rationale for choosing this architecture along with considerations of critical issues and trade-offs:

1. **Modularity and Separation of Concerns:**
   * **Rationale:** Modular architecture allows for independent development, testing, and maintenance of each module. This promotes code reusability and makes it easier to extend or modify specific functionalities without affecting the entire system.
   * **Critical Issues:** The main challenge lies in ensuring effective communication and coordination between modules.
2. **Layered Structure:**
   * **Rationale:** The layered architecture separates concerns into different levels, such as presentation, business logic, and data access. This separation enhances maintainability and flexibility by allowing changes in one layer without affecting others.
   * **Critical Issues:** The main consideration is the potential performance impact due to inter-layer communication
3. **User-Centric Design:**
   * **Rationale:** The architecture is designed with a focus on the different user roles (Admin, Teacher, Student, Parent), ensuring that each module caters to the specific needs and responsibilities of that user type.
   * **Critical Issues:** The challenge lies in maintaining a consistent user experience across modules.
4. **Consideration of Alternatives:**
   * **Monolithic Architecture:** While a could simplify initial development, it poses challenges in terms of scalability and maintenance as the system grows.
   * **Microservices Architecture:** This was considered but deemed overly complex for the current scale of the project. Microservices might be more suitable for larger-scale systems with independent services.
5. **Technology Stack:**
   * **Rationale:** The use of ASP.NET technology aligns with the developer's background in software engineering. ASP.NET provides a robust and scalable framework for web applications.
   * **Critical Issues and Trade-offs:** While ASP.NET offers strong server-side capabilities, there might be considerations for client-side technologies for a more dynamic user interface.
6. **Data Security and Privacy:**
   * **Rationale:** The architecture considers the importance of data safety, especially in handling sensitive information related to students and staff.
   * **Critical Issues and Trade-offs:** The challenge is to implement robust security measures, including encryption and access controls, which may add complexity but are essential for compliance and data protection.

In summary, the chosen architecture strikes a balance between modularity, maintainability, and user-centric design. It aligns with the developer's expertise while considering critical issues such as communication between modules and potential performance impacts. The architecture is scalable for future enhancements and provides a foundation for a reliable and user-friendly School ERP System.

# 4. Data Design

## 4.1. Data Description

1. **Information Domain Transformation:**
   * **Explanation:** The information domain of the School ERP System encompasses various entities like students, teachers, classes, attendance, scores, learning materials, fees, announcements, complaints, payments, and staff information. This information needs to be structured and organized to facilitate efficient processing and retrieval.
   * **Implementation:** Each entity in the information domain is transformed into corresponding data structures within the system. For example, students, teachers, and staff information may be represented as separate tables in a database, with attributes like names, IDs, and contact details.
2. **Storage, Processing, and Organization:**
   * **Explanation:** The major entities are stored in a relational database, allowing for efficient data management. The data is processed using CRUD (Create, Read, Update, Delete) operations to perform tasks like adding new students, retrieving attendance records, updating scores, or deleting complaints.
   * **Implementation:** The system employs a relational database management system (RDBMS) to store data. Tables are created for entities such as Students, Teachers, Classes, Attendance, Scores, Learning Materials, Fees, Announcements, Complaints, Payments, and Staff Information. Relationships between entities are established through keys (e.g., student ID linking attendance records).
3. **Databases and Data Storage:**
   * **Explanation:** The School Management System relies on a central database, specifically Microsoft SQL Server, to efficiently manage and organize data. This database plays a crucial role in maintaining consistency and integrity across different modules of the system.
   * **Implementation:** The choice of Microsoft SQL Server as the database management system (DBMS) is based on considerations such as scalability, performance, and the developer's familiarity with the technology. Tables within the SQL Server database store data related to each entity, ensuring a structured and organized data repository. The relational nature of SQL Server facilitates establishing relationships between entities and supports the systematic organization of information.

## 4.2. Data Dictionary

1. **Admin:**
   * **Attributes:**
     + UserID (int): Unique identifier for the admin user.
     + Username (varchar): Username for admin login.
     + Password (varchar): Password for admin login.
   * **Methods:**
     + Login(username: varchar, password: varchar): Authenticates the admin user.
2. **Teacher:**
   * **Attributes:**
     + TeacherID (int): Unique identifier for the teacher.
     + Name (varchar): Full name of the teacher.
     + Subject (varchar): Subject taught by the teacher.
   * **Methods:**
     + MarkAttendance(studentID: int, date: date, status: bool): Records student attendance.
     + AssignMarks(studentID: int, subject: varchar, marks: int): Assigns marks to a student.
3. **Student:**
   * **Attributes:**
     + StudentID (int): Unique identifier for the student.
     + Name (varchar): Full name of the student.
     + Class (varchar): Class in which the student is enrolled.
   * **Methods:**
     + AccessLearningMaterial(): Allows the student to access learning materials.
     + ParticipateOnlineClass(): Enables the student to participate in online classes.
     + ContactTeacher(message: varchar): Sends a message to a teacher.
4. **Parent:**
   * **Attributes:**
     + ParentID (int): Unique identifier for the parent.
     + Name (varchar): Full name of the parent.
     + ChildName (varchar): Full name of the associated child.
   * **Methods:**
     + RegisterComplaint(complaintText: varchar): Registers a complaint.
     + ProcessPayment(amount: decimal): Processes a payment for fees.
     + AccessStaffInformation(): Allows access to information about school staff.
     + ContactTeacher(message: varchar): Sends a message to a teacher.

# Component Design

1. **Admin Module:**
   * **Register Students:**

Pseudocode:

FUNCTION AddNewStudent(studentDetails: StudentDetails)

IF ValidateStudentDetails(studentDetails) THEN

CreateNewStudentRecord(studentDetails)

**Description:**

* AddNewStudent function is responsible for adding a new student to the system.
* It first validates the provided student details using the ValidateStudentDetails function.
* If the details are valid, a new student record is created using CreateNewStudentRecord.
  + **Fee Structure and Class Setup:**

Pseudocode:

FUNCTION ManageFeeStructureAndClass(classDetails:ClassDetails,feeStructure: FeeStructure)

IF ValidateInput(classDetails, feeStructure) THEN

UpdateOrCreateFeeStructureAndClass(classDetails, feeStructure)

**Description:**

* ManageFeeStructureAndClass function handles the management of fee structure and class setup.
* It validates the input data using ValidateInput.
* If the input is valid, it updates or creates fee structure and class records using UpdateOrCreateFeeStructureAndClass.
  + **Register Staff:**

Pseudocode:

FUNCTION AddNewStaff(staffDetails: StaffDetails)

IF ValidateStaffDetails(staffDetails) THEN

CreateNewStaffRecord(staffDetails)

**Description:**

* AddNewStaff function is responsible for adding a new staff member to the system.
* It validates the provided staff details using ValidateStaffDetails.
* If the details are valid, a new staff record is created using CreateNewStaffRecord.
  + **Fee Collection:**

Pseudocode:

FUNCTION CollectFee(studentID: STRING, feeDetails: FeeDetails) IF ValidateInput(studentID, feeDetails) THEN ProcessFeePayment(studentID, feeDetails)

**Description:**

* CollectFee function manages the process of fee collection for a student.
* It validates the input data, including student ID and fee details, using ValidateInput.
* If the input is valid, it proceeds to process the fee payment using ProcessFeePayment.
  + **AssignClass:**

Pseudocode:

FUNCTION AssignClass(studentID: STRING, classDetails: ClassDetails) IF ValidateInput(studentID, classDetails) THEN UpdateClassAssignment(studentID, classDetails) END IF END FUNCTION

**Description:**

* AssignClass function is responsible for assigning a class to a student.
* It validates the input data, including student ID and class details, using ValidateInput.
* If the input is valid, it updates the class assignment for the student using UpdateClassAssignment.
  + **Staff Salaries:**

Pseudocode:

FUNCTION ProcessSalaries(staffID: STRING, salaryDetails: SalaryDetails) IF ValidateInput(staffID, salaryDetails) THEN CalculateAndProcessSalary(staffID, salaryDetails) END IF END FUNCTION

**Description:**

* ProcessSalaries function handles the processing of staff salaries.
* It validates the input data, including staff ID and salary details, using ValidateInput.
* If the input is valid, it calculates and processes the salary using CalculateAndProcessSalary.

1. **Teacher Module:**
   * **ManageAttendance:**

Pseudocode:

FUNCTION ManageAttendance(classDetails: ClassDetails, studentAttendance: StudentAttendance) IF ValidateInput(classDetails, studentAttendance) THEN UpdateAttendanceRecords(classDetails, studentAttendance) END IF END FUNCTION

**Description:**

* ManageAttendance function manages the process of marking attendance for students in a class.
* It validates the input data, including class details and student attendance, using ValidateInput.
* If the input is valid, it updates the attendance records using UpdateAttendanceRecords.
  + **Manage Marks:**

Pseudocode:

FUNCTION ManageMarks(studentID: STRING, subject: STRING, marks: Marks) IF ValidateInput(studentID, subject, marks) THEN UpdateOrCreateMarksRecord(studentID, subject, marks) END IF END FUNCTION

**Description:**

* ManageMarks function handles the process of assigning marks to a student for a specific subject.
* It validates the input data, including student ID, subject, and marks, using ValidateInput.
* If the input is valid, it updates or creates marks records using UpdateOrCreateMarksRecord.
  + **Manage Learning Material:**

Pseudocode:

FUNCTION ManageLearningMaterial(materialDetails: MaterialDetails) IF ValidateInput(materialDetails) THEN UploadLearningMaterial(materialDetails) END IF END FUNCTION

**Description:**

* ManageLearningMaterial function is responsible for managing the upload of learning materials.
* It validates the input data, including material details, using ValidateInput.
* If the input is valid, it proceeds to upload the learning material using UploadLearningMaterial.
  + **Make Announcement:**

Pseudocode:

FUNCTION MakeAnnouncement(announcementDetails: AnnouncementDetails) IF ValidateInput(announcementDetails) THEN PublishAnnouncement(announcementDetails) END IF END FUNCTION

**Description:**

* MakeAnnouncement function handles the process of making announcements.
* It validates the input data, including announcement details, using ValidateInput.
* If the input is valid, it proceeds to publish the announcement using PublishAnnouncement.
  + **Conduct Online Class:**

Pseudocode:

FUNCTION ConductOnlineClass(classDetails: ClassDetails) IF ValidateInput(classDetails) THEN ScheduleAndConductOnlineClass(classDetails) END IF END FUNCTION

**Description:**

* ConductOnlineClass function manages the scheduling and conducting of online classes.
* It validates the input data, including class details, using ValidateInput.
* If the input is valid, it schedules and conducts the online class using ScheduleAndConductOnlineClass.

1. **Student Module:**
   * **Access Learning Material:**

Pseudocode:

FUNCTION AccessLearningMaterial(materialID: STRING) RetrieveAndDisplayLearningMaterial(materialID) END FUNCTION

**Description:**

* AccessLearningMaterial function allows students to access and retrieve learning materials.
* It retrieves and displays the learning material based on the provided material ID using RetrieveAndDisplayLearningMaterial.
  + **Participate In Online Class:**

Pseudocode:

FUNCTION ParticipateInOnlineClass(classDetails: ClassDetails) JoinOnlineClass(classDetails) END FUNCTION

**Description:**

* ParticipateInOnlineClass function allows students to participate in online classes.
* It joins the online class based on the provided class details using JoinOnlineClass.
  + **CommunicateWithTeacher:**

Pseudocode:

FUNCTION CommunicateWithTeacher(messageDetails: MessageDetails) SendAndReceiveMessages(messageDetails) END FUNCTION

**Description:**

* CommunicateWithTeacher function facilitates communication between students and teachers.
* It sends and receives messages based on the provided message details using SendAndReceiveMessages.
  + **SubmitQuery:**

Pseudocode:

FUNCTION SubmitQuery(queryDetails: QueryDetails)

StoreQueryInDatabase(queryDetails)

END FUNCTION

**Description:**

SubmitQuery function allows students to submit queries.

It stores the submitted query in the database using StoreQueryInDatabase.

1. **Parent Module:**
   * **Register Complaint:**

Pseudocode:

FUNCTION RegisterComplaint(complaintDetails: ComplaintDetails)

StoreComplaintInDatabase(complaintDetails)

END FUNCTION

**Description:**

* RegisterComplaint function allows parents to register complaints.
* It stores the registered complaint in the database using StoreComplaintInDatabase.
  + **ProcessPayment:**

Pseudocode:

FUNCTION ProcessPayment(paymentDetails: PaymentDetails)

ProcessPaymentAndUpdateRecords(paymentDetails)

END FUNCTION

**Description:**

* ProcessPayment function manages the processing of payments by parents.
* It processes the payment and updates relevant records using ProcessPaymentAndUpdateRecords.
  + **Access Staff Information:**

Pseudocode:

FUNCTION AccessStaffInformation()

RetrieveAndDisplayStaffInformation()

END FUNCTION

**Description:**

* AccessStaffInformation function allows parents to access staff information.
* It retrieves and displays staff information using RetrieveAndDisplayStaffInformation.
  + **CommunicateWithTeacher:**

Pseudocode:

FUNCTION CommunicateWithTeacher(messageDetails: MessageDetails)

SendAndReceiveMessages(messageDetails)

END FUNCTION

**Description:**

* CommunicateWithTeacher function facilitates communication between parents and teachers.
* It sends and receives messages based on the provided message details using SendAndReceiveMessages

# 6. Human Interface Design

## 6.1. Overview of User Interface

1. **Admin Module:**

Overview:

The Admin Module provides administrators with the tools to manage various aspects of the school, including student registration, fee management, staff registration, class assignments, and salary processing.

**User Interaction:**

* **Register Students:**
  + Admins can access a user-friendly form to input student details.
  + Feedback: Confirmation message upon successful registration, or error message if there are issues.
* **Fee Structure and Class Setup:**
  + Admins can set up and manage fee structures and class assignments.
  + Feedback: Confirmation messages for successful updates or error messages for any issues.
* **Register Staff:**
  + Admins can add new staff members to the system.
  + Feedback: Confirmation messages or error notifications based on the success of staff registration.
* **Fee Collection:**
  + Admins can collect fees from students.
  + Feedback: Confirmation messages for successful transactions or error notifications for any failures.
* **Class Assignment:**
  + Admins can assign students to classes.
  + Feedback: Confirmation messages or error notifications based on the success of class assignments.
* **Staff Salaries:**
  + Admins can process salaries for staff members.
  + Feedback: Confirmation messages or error notifications for successful or unsuccessful salary processing.

1. **Teacher Module:**

Overview:

The Teacher Module empowers teachers to manage attendance, assign marks, upload learning materials, make announcements, and conduct online classes.

**User Interaction:**

1. **Mark Attendance:**
   * Teachers can mark attendance for their classes.
   * Feedback: Confirmation messages or error notifications based on the success of attendance marking.
2. **Assign Marks:**
   * Teachers can assign marks to students.
   * Feedback: Confirmation messages or error notifications for successful or unsuccessful mark assignments.
3. **Learning Material Upload:**
   * Teachers can upload learning materials for students.
   * Feedback: Confirmation messages or error notifications based on the success of material uploads.
4. **Announcements:**
   * Teachers can make announcements for their classes.
   * Feedback: Confirmation messages for successful announcements.
5. **Online Class Conduct:**
   * Teachers can schedule and conduct online classes.
   * Feedback: Confirmation messages for successful scheduling and conducting of online classes.

**Student Module:**

Overview:

The Student Module offers students access to learning materials, participation in online classes, communication with teachers, and submission of queries.

**User Interaction:**

1. **Learning Material Access:**
   * Students can access learning materials uploaded by teachers.
   * Feedback: Display of accessed learning materials.
2. **Online Class Participation:**
   * Students can join online classes.
   * Feedback: Participation status and relevant information during online classes.
3. **Communication with Teacher:**
   * Students can communicate with teachers.
   * Feedback: Message exchange with teachers.

**Parent Module:**

Overview:

The Parent Module enables parents to register complaints, process payments, access staff information, and communicate with teachers.

**User Interaction:**

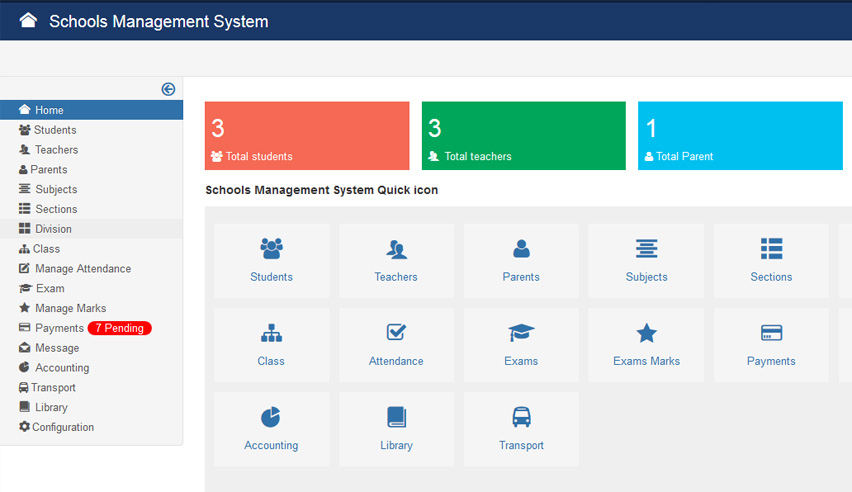
1. **Complaint Registration:**
   * Parents can register complaints.
   * Feedback: Confirmation messages for successfully registered complaints.
2. **Communication with Teacher:**
   * Parents can communicate with teachers.
   * Feedback: Message exchange with teachers.

**Note:**

* The user interface design should prioritize simplicity, clarity, and intuitive navigation to ensure a positive user experience.
* Specific design elements, such as buttons, forms, and messages, will depend on the chosen technology stack and design principles.

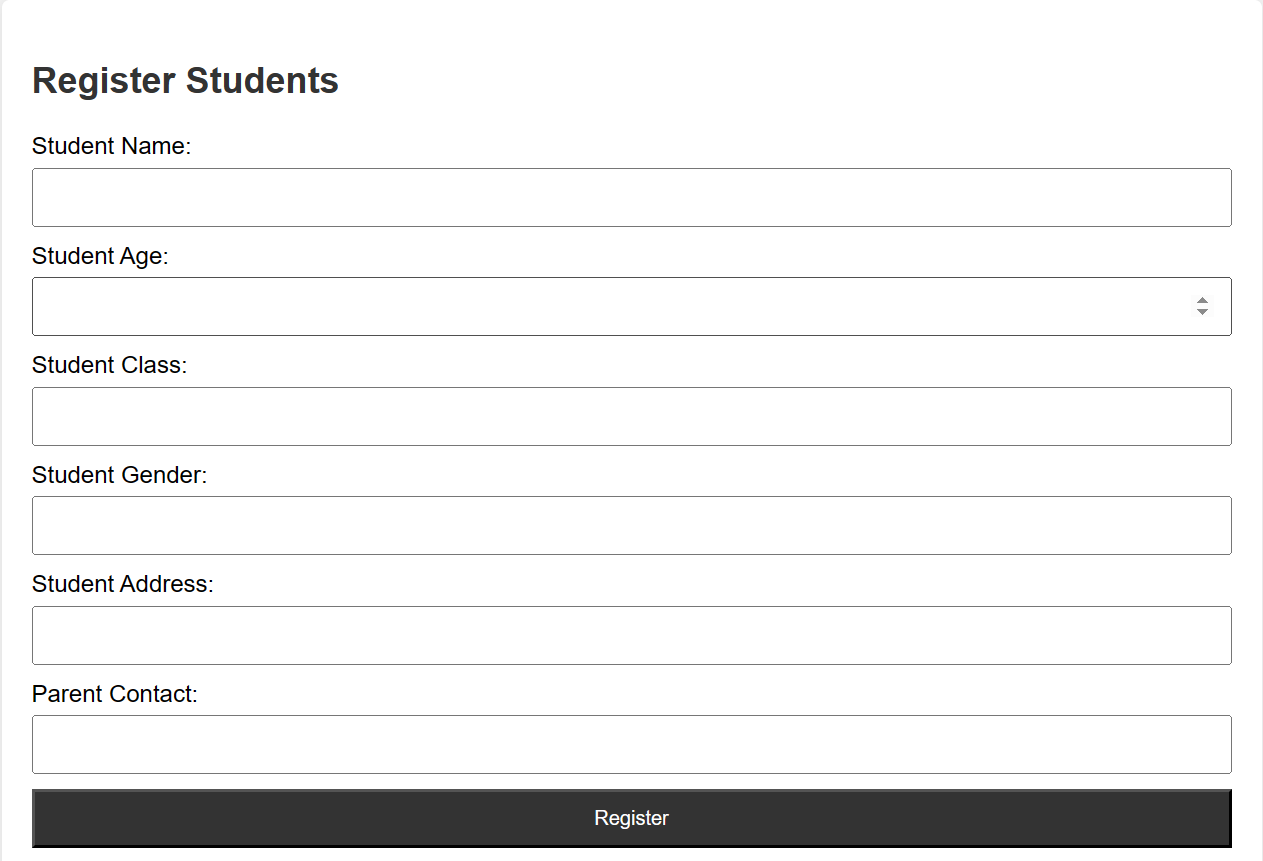
## 6.2. Screen Images

* **Abstract Overview**(as accurate as possible)

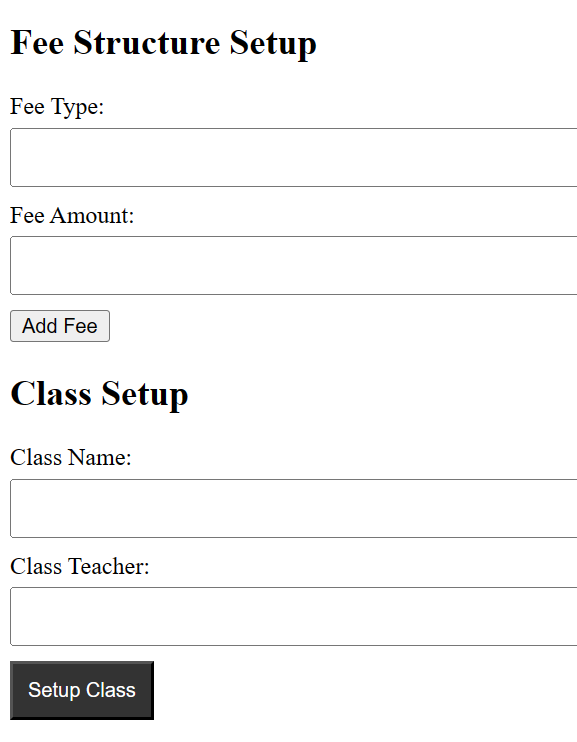
****

## 1. Admin Module:

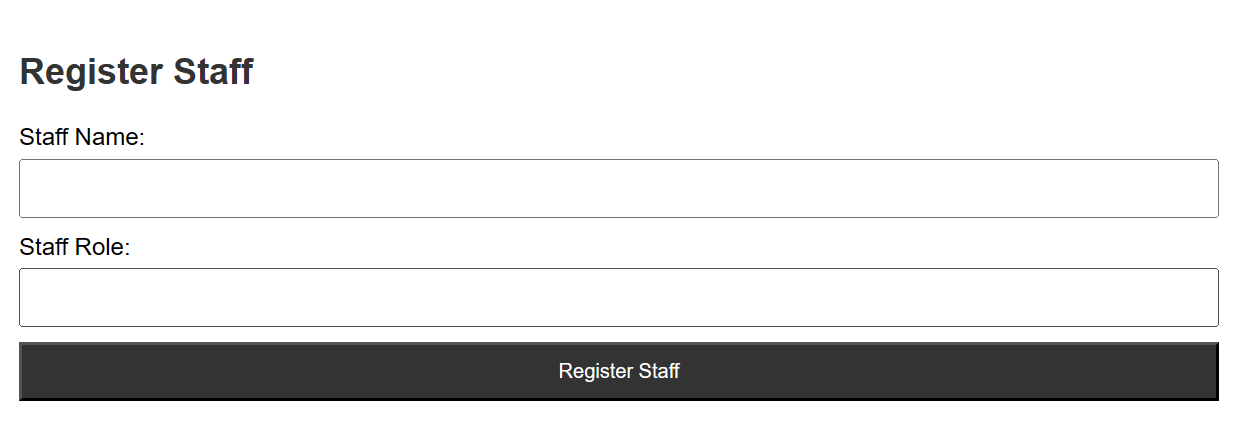
* **Register Students:**

****

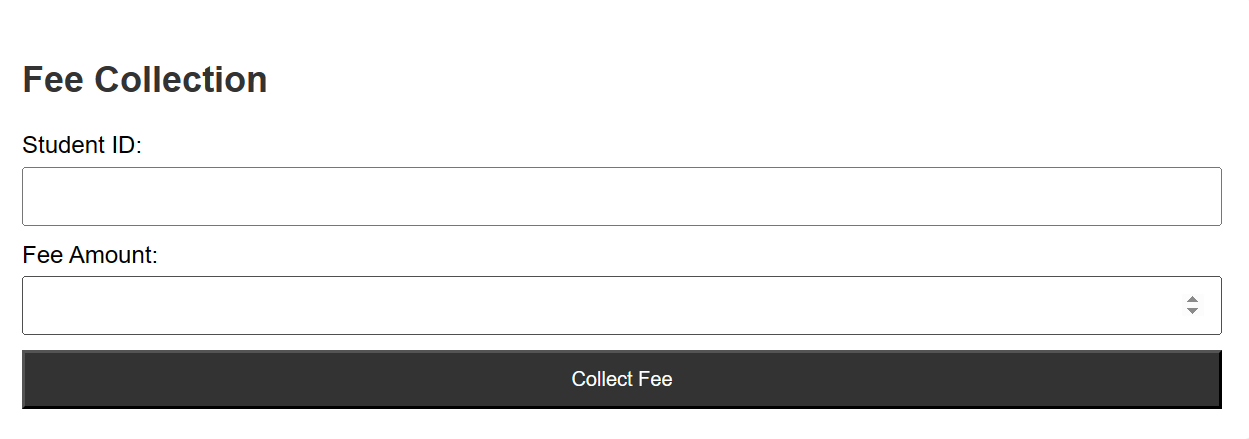
* **Fee Structure and Class Setup:**

****

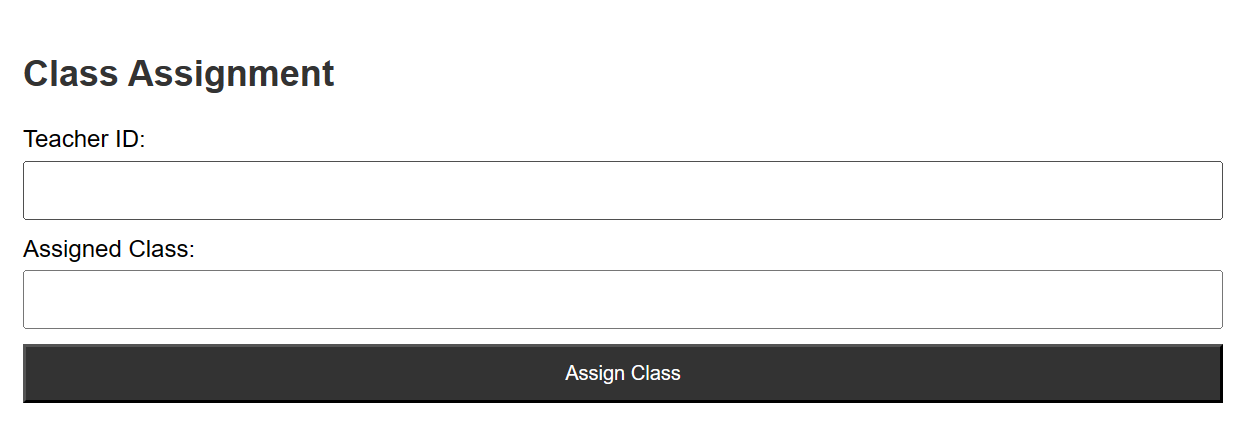
* **Register Staff:**

****

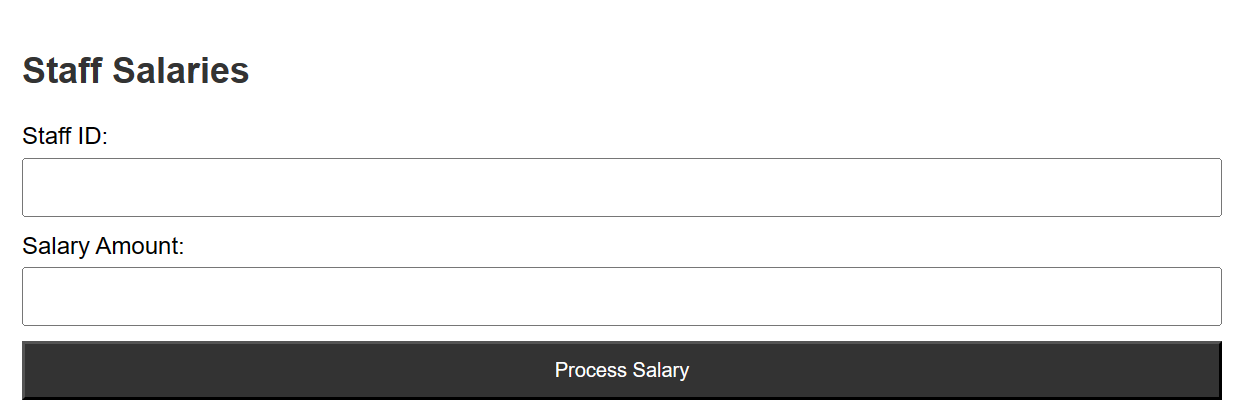
* **Fee Collection:**

****

* **Class Assignment:**

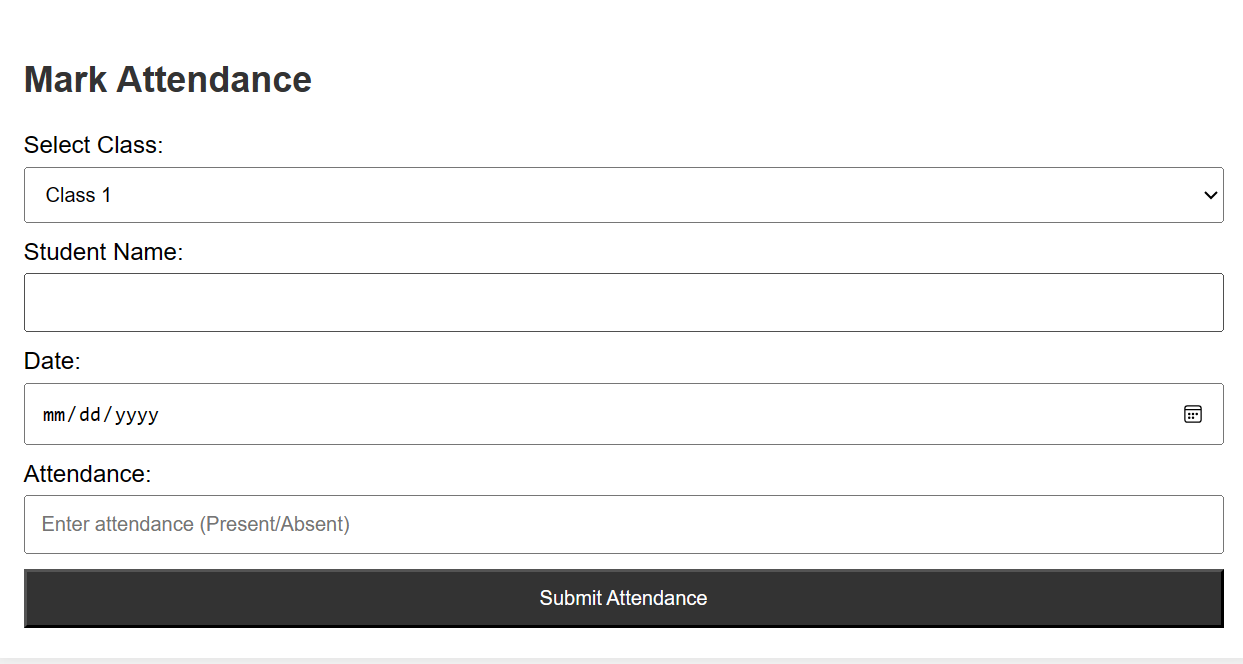
****

* **Staff Salaries:**

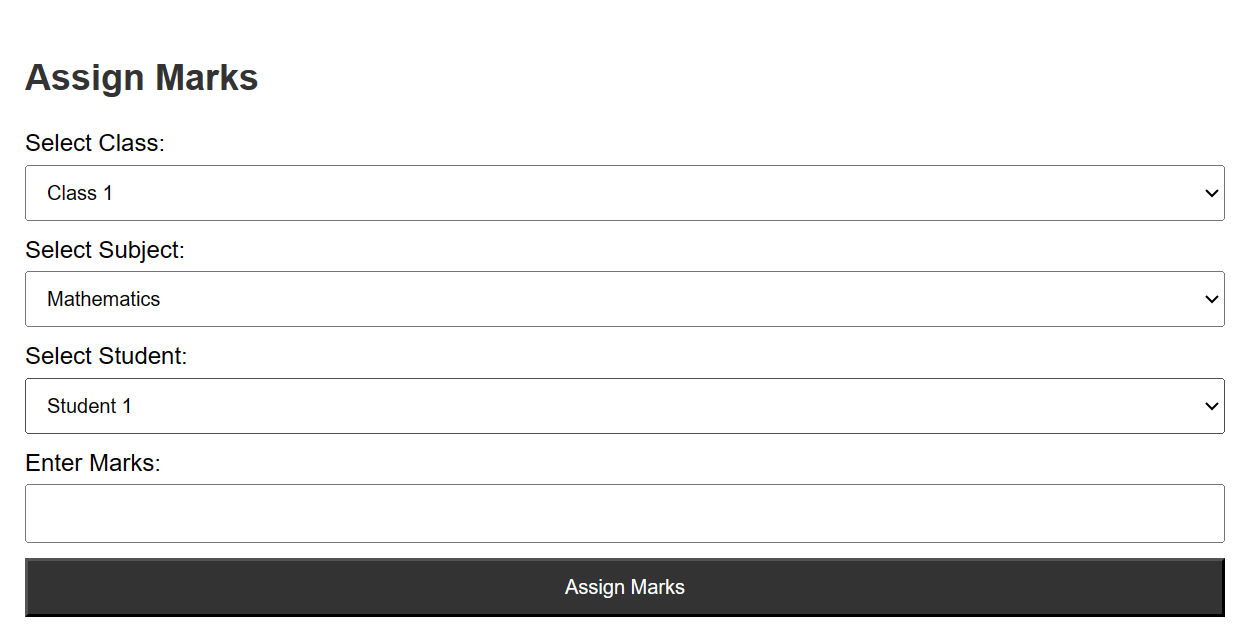
****

## 2. Teacher Module:

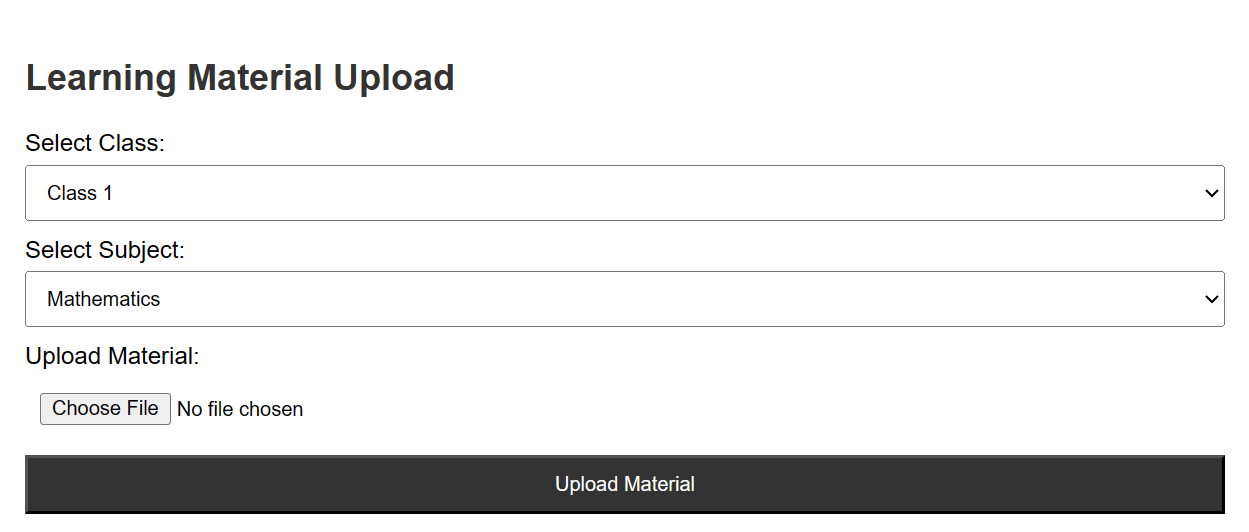
* **Mark Attendance:**

****

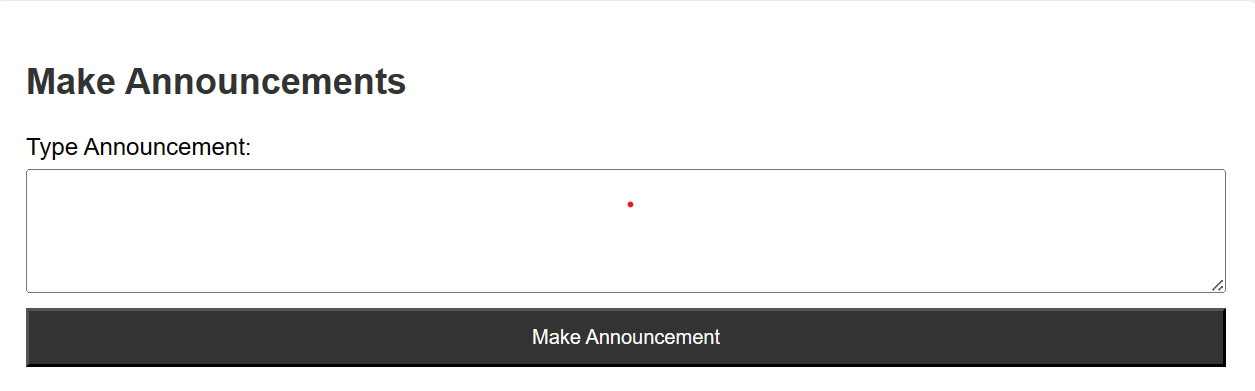
* **Assign Marks:**

****

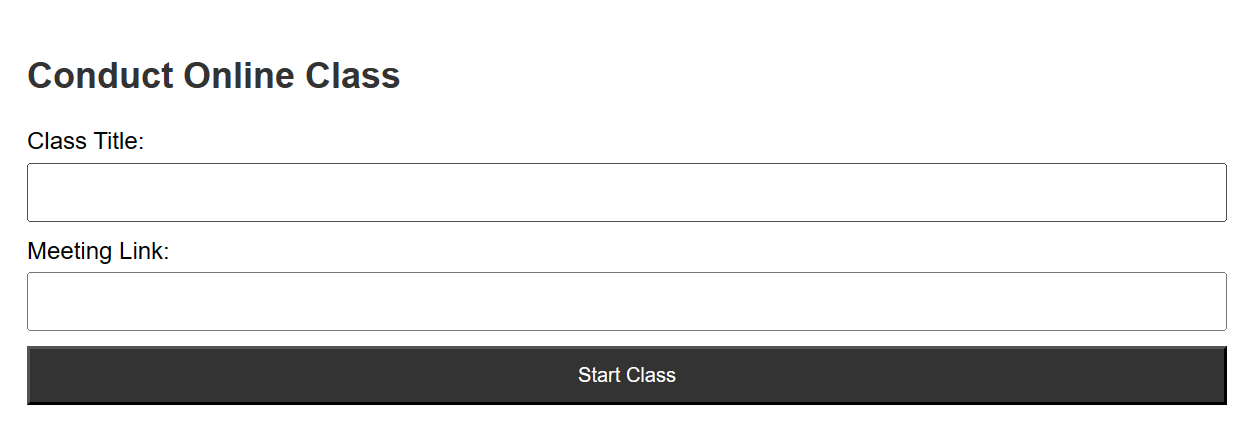
* **Learning Material Upload:**

****

* **Announcements:**

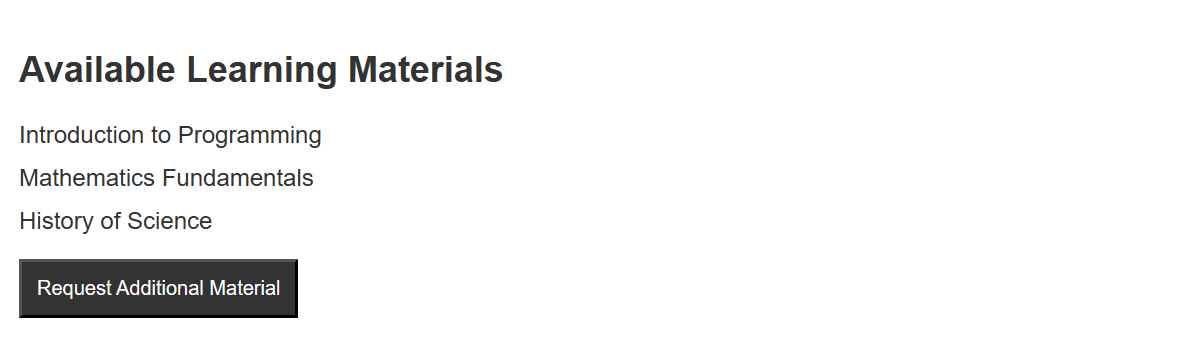
****

* **Online Class Conduct:**

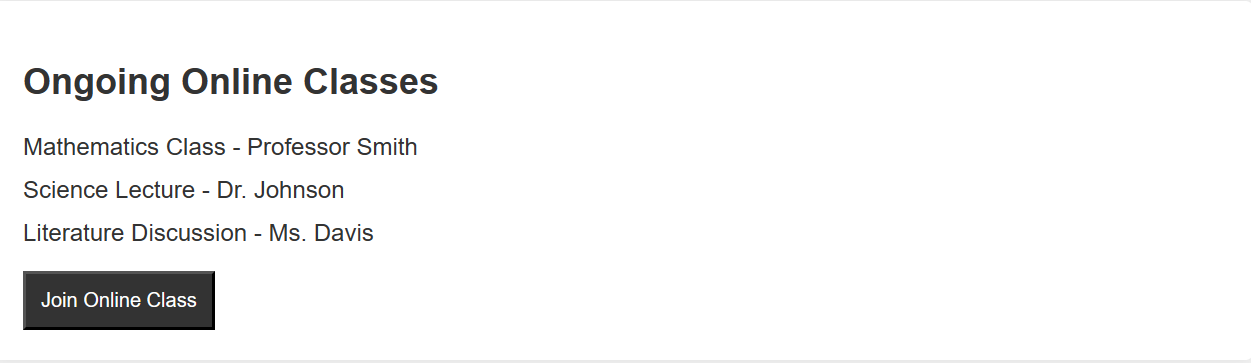
****

## Student Module:

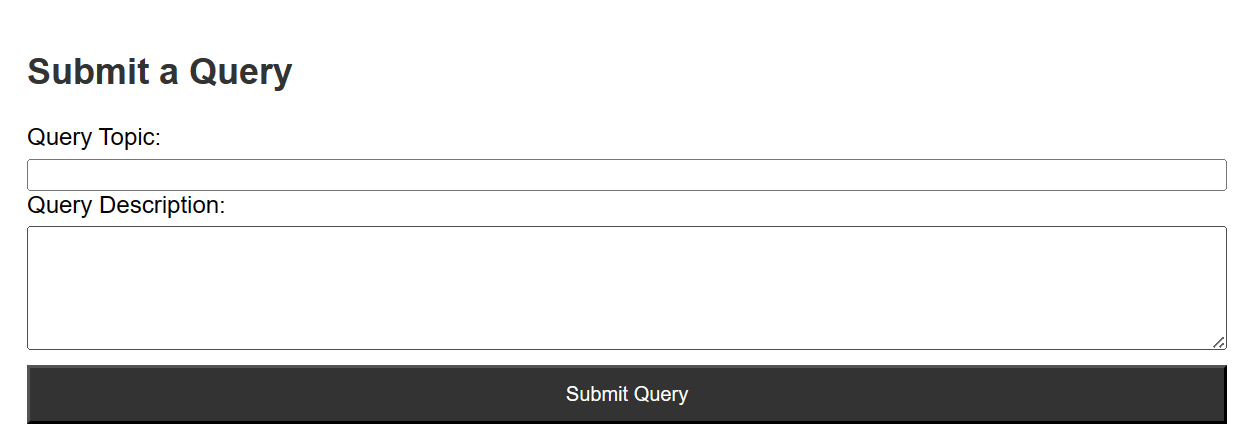
* **Learning Material Access:**

****

* **Online Class Participation:**

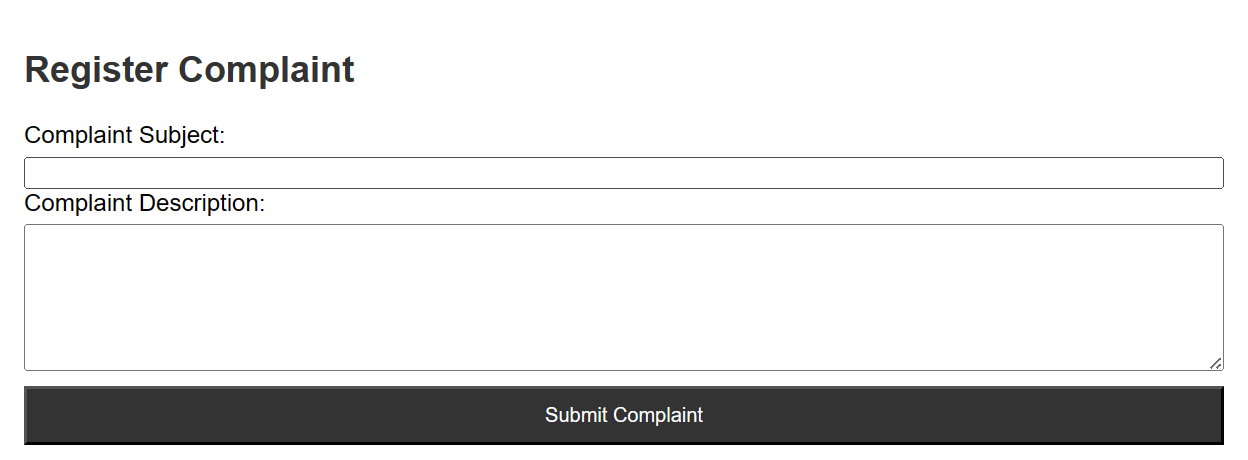
****

* **Query Submission:**

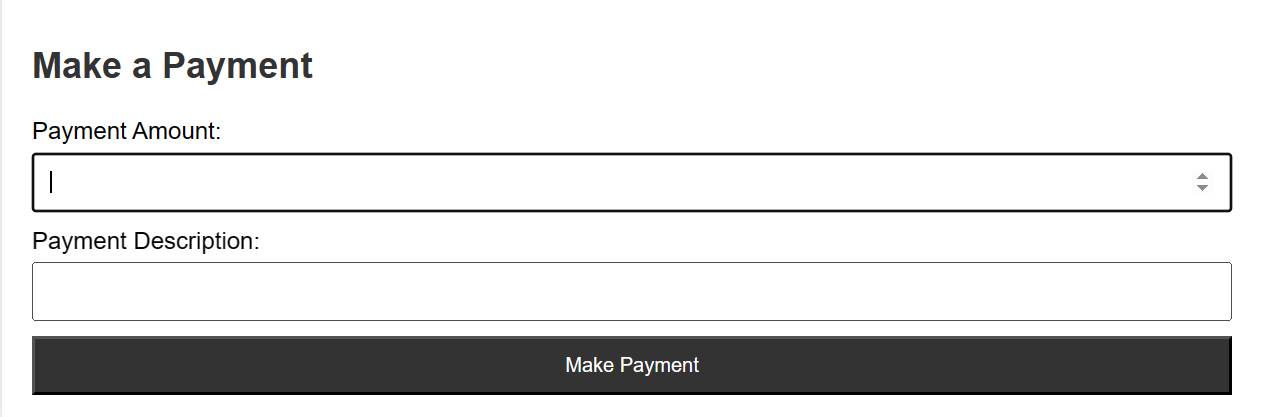
****

## 4. Parent Module:

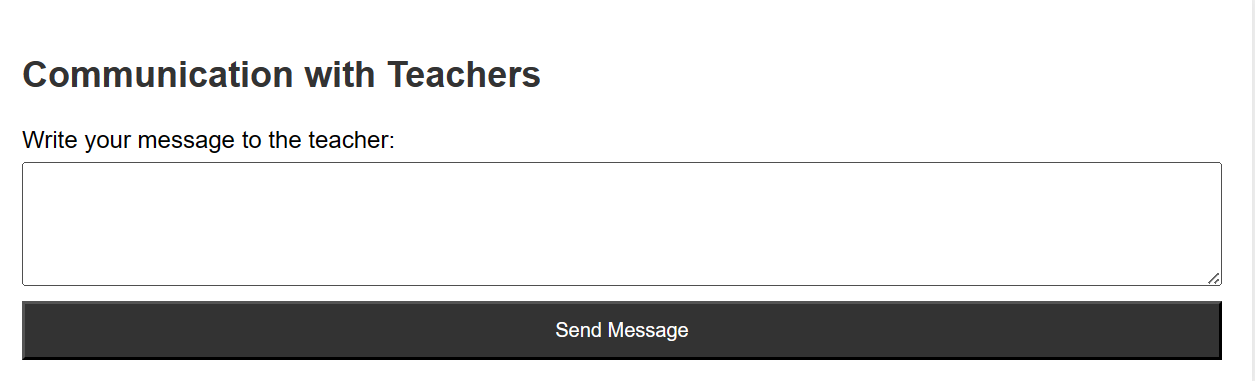
* **Complaint Registration:**

****

* **Payment Processing:**

****

* **Teacher Communication:**

****

# 7. Requirements Matrix

|  |  |  |
| --- | --- | --- |
| **Functional Requirement** | **System Component** | **Data Structure / Database Table** |
| 3.2.1 | Admin Module - User Management | UserDatabase |
| 3.2.1.1 | - Register Students | StudentRecords, ClassInformation |
| 3.2.1.2 | - Fee Structure and Class Setup | FeeStructure, ClassConfiguration |
| 3.2.1.3 | - Register Staff | StaffRecords |
| 3.2.1.4 | - Fee Collection | FeeTransactions |
| 3.2.1.5 | - Class Assignment | ClassAssignments, StudentRecords |
| 3.2.1.6 | - Staff Salaries | SalaryRecords, StaffInformation |
| 3.2.2 | Teacher Module - Classroom Management | Attendance, Marks, Announcements, LearningMaterial |
| 3.2.2.1 | - Mark Attendance | AttendanceRecords, StudentRecords |
| 3.2.2.2 | - Assign Marks | MarksRecords, StudentRecords |
| 3.2.2.3 | - Learning Material Upload | LearningMaterialRepository |
| 3.2.2.4 | - Announcements | AnnouncementData |
| 3.2.2.5 | - Online Class Conduct | OnlineClassData |
| 3.2.3 | Student Module - Learning Interaction | LearningMaterial, OnlineClass, Communication, Queries |
| 3.2.3.1 | - Learning Material Access | LearningMaterialRepository |
| 3.2.3.2 | - Online Class Participation | OnlineClassData, AttendanceRecords |
| 3.2.3.3 | - Teacher Communication | CommunicationData |
| 3.2.3.4 | - Query Submission | QueryData |
| 3.2.4 | Parent Module - Interaction and Support | Complaints, Payments, StaffInformation, TeacherCommunication |
| 3.2.4.1 | - Complaint Registration | ComplaintData |
| 3.2.4.2 | - Payment Processing | PaymentRecords, FeeTransactions |
| 3.2.4.3 | - Staff Information Access | StaffInformationDatabase |
| 3.2.4.4 | - Teacher Communication | CommunicationData |