**1. Admission Policies and Fee Categories**

|  |  |
| --- | --- |
| **User Story** | **As an admin, I want to define admission policies and fee categories so that the fee structure is clear and organized.** |
| **User Story Details** | This story involves creating interfaces to define admission policies and fee categories. |
| **User Story Output** | Admins can define and manage admission policies and fee categories. |
| **Data Entities Input List** | AdmissionPolicy (PolicyId, Name, Description, DateCreated),  FeeCategory (CategoryId, Name, Description, FeeStructure) |
| **Task** | 1. Create a React component for admission policy management. 2. Create a React component for fee category management. 3. Create a .NET Core API endpoint to handle policy and category management. 4. Connect the React components to the API endpoint using Axios. 5. Store policy and category details in the MSSQL database. |
| **Dev Notes** | * Ensure proper validation for policy and category details. * Implement version control for admission policies. |
| **Acceptance Criteria** | * Admins can define and manage admission policies. * Admins can define and manage fee categories. * Policy and category details are stored and managed in the database. * Confirmation messages are displayed upon successful actions. |
| **Estimated Points** | 8 |
| **Testing Time** | 4 Hrs. |

**2. Fee Structure**

|  |  |
| --- | --- |
| **User Story** | **As an admin, I want to define fee categories and structures so that billing can be managed.** |
| **User Story Details** | This story involves creating a form to define fee categories and structures. |
| **User Story Output** | Fee structures are defined and stored. |
| **Data Entities Input List** | Fee (FeeId, Category, Amount, DueDate) |
| **Task** | 1. Create a React component for the fee structure form. 2. Implement form validation. 3. Create a .NET Core API endpoint to handle fee structure creation. 4. Connect the React form to the API endpoint using Axios. 5. Store fee structures in the MSSQL database. |
| **Dev Notes** | * Include fields for late fees and discounts. * Consider different payment schedules (monthly, quarterly, annually). |
| **Acceptance Criteria** | * An admin can fill out and submit the fee structure form. * The form performs client-side validation. * Upon submission, the form sends a POST request to the backend. * The backend stores the fee structures in the database. * Confirmation message is displayed upon successful fee structure creation. |
| **Estimated Points** | 7 |
| **Testing Time** | 3 Hrs. |

**3. Student Enrollment**

|  |  |
| --- | --- |
| **User Story** | **As an admin, I want to register new students so that I can manage their profiles and information.** |
| **User Story Details** | This story involves creating a form to register new students and storing their information in the database. |
| **User Story Output** | A new student is registered, and their profile is stored in the system. |
| **Data Entities Input List** | Student (StudentId, Name, DOB, Address, Phone, Email, EnrollmentDate) |
| **Task** | 1. Create a React component for the registration form. 2. Implement form validation. 3. Create a .NET Core API endpoint to handle student registration. 4. Connect the React form to the API endpoint using Axios. 5. Store student information in the MSSQL database. |
| **Dev Notes** | * Ensure proper validation for email and phone number. * Use Model State in .NET Core for server-side validation. |
| **Acceptance Criteria** | * A user can fill out and submit the registration form. * The form performs client-side validation. * Upon submission, the form sends a POST request to the backend. * The backend stores the student information in the database. * Confirmation message is displayed upon successful registration. |
| **Estimated Points** | 8 |
| **Testing Time** | 4 Hrs. |

**4. Discipline and Academia Management**

|  |  |
| --- | --- |
| **User Story** | **As a teacher, I want to manage student discipline and academic reports so that their overall performance is tracked.** |
| **User Story Details** | This story involves creating interfaces to manage and upload academic and discipline reports. |
| **User Story Output** | Teachers can edit and upload students' academic and discipline reports. |
| **Data Entities Input List** | DisciplineReport (ReportId, StudentId, ReportDetails, Date),  AcademicReport (ReportId, StudentId, Grades, Comments, Date) |
| **Task** | 1. Create a React component for discipline report management. 2. Create a React component for academic report management. 3. Create a .NET Core API endpoint to handle report submissions. 4. Connect the React components to the API endpoint using Axios. 5. Store report details in the MSSQL database. |
| **Dev Notes** | * Ensure proper validation for report details. * Implement role-based access control for report management. |
| **Acceptance Criteria** | * Teachers can edit and upload discipline reports. * Teachers can edit and upload academic reports. * Report details are stored and managed in the database. * Confirmation messages are displayed upon successful actions. |
| **Estimated Points** | 10 |
| **Testing Time** | 4 Hrs. |

**5. Daily Attendance Tracking**

|  |  |
| --- | --- |
| **User Story** | **As a teacher, I want to record daily attendance so that I can track student presence.** |
| **User Story Details** | This story involves creating an interface for teachers to mark attendance and storing the records. |
| **User Story Output** | Attendance records are updated daily. |
| **Data Entities Input List** | Attendance (AttendanceId, StudentId, Date, Status) |
| **Task** | 1. Create a React component for the attendance form. 2. Implement a calendar/date-picker for selecting the date. 3. Create a .NET Core API endpoint to handle attendance submission. 4. Connect the React form to the API endpoint using Axios. 5. Store attendance records in the MSSQL database. |
| **Dev Notes** | * Consider edge cases like holidays and weekends. * Implement bulk attendance upload feature if needed. |
| **Acceptance Criteria** | * A teacher can select a date and mark attendance for each student. * The form performs client-side validation. * Upon submission, the form sends a POST request to the backend. * The backend stores the attendance records in the database. * Confirmation message is displayed upon successful submission. |
| **Estimated Points** | 5 |
| **Testing Time** | 3 Hrs. |

**6. Grading System**

|  |  |
| --- | --- |
| **User Story** | **As a teacher, I want to manage grades and generate report cards so that I can evaluate student performance.** |
| **User Story Details** | This story involves creating an interface to input grades and generate report cards. |
| **User Story Output** | Grades are managed, and report cards are generated. |
| **Data Entities Input List** | Grades (GradeId, StudentId, CourseId, Grade, Comments) |
| **Task** | 1. Create a React component for the grading form. 2. Implement form validation. 3. Create a .NET Core API endpoint to handle grade submission. 4. Connect the React form to the API endpoint using Axios. 5. Store grades in the MSSQL database. 6. Create a report card generation feature. |
| **Dev Notes** | * Ensure that only authorized users can input grades. * Consider different grading scales (letters, percentages). |
| **Acceptance Criteria** | * A teacher can input and submit grades for each student. * The form performs client-side validation. * Upon submission, the form sends a POST request to the backend. * The backend stores the grades in the database. * Report cards are generated and can be viewed/downloaded. |
| **Estimated Points** | 6 |
| **Testing Time** | 3 Hrs. |

**7. Course Creation**

|  |  |
| --- | --- |
| **User Story** | **As an admin, I want to define courses and curricula so that they can be assigned to teachers and students.** |
| **User Story Details** | This story involves creating a form to define courses and curricula and storing the information. |
| **User Story Output** | New courses and curricula are created and stored. |
| **Data Entities Input List** | Course (CourseId, Name, Description, Credits) |
| **Task** | 1. Create a React component for the course creation form. 2. Implement form validation. 3. Create a .NET Core API endpoint to handle course creation. 4. Connect the React form to the API endpoint using Axios. 5. Store course information in the MSSQL database. |
| **Dev Notes** | * Include fields for course prerequisites and corequisites. * Consider the need for course categorization (mandatory, elective). |
| **Acceptance Criteria** | * An admin can fill out and submit the course creation form. * The form performs client-side validation. * Upon submission, the form sends a POST request to the backend. * The backend stores the course information in the database. * Confirmation message is displayed upon successful course creation. |
| **Estimated Points** | 8 |
| **Testing Time** | 4 Hrs. |

**8. Exam Scheduling**

|  |  |
| --- | --- |
| **User Story** | **As an admin, I want to define exam dates and allocate exam halls so that exams can be conducted efficiently.** |
| **User Story Details** | This story involves creating a scheduling interface for exams and managing exam halls. |
| **User Story Output** | Exam schedules are created and stored. |
| **Data Entities Input List** | Exam (ExamId, CourseId, Date, Time, Hall) |
| **Task** | 1. Create a React component for the exam scheduling form. 2. Implement form validation. 3. Create a .NET Core API endpoint to handle exam scheduling. 4. Connect the React form to the API endpoint using Axios. 5. Store exam schedules in the MSSQL database. |
| **Dev Notes** | * Ensure no conflicts in exam timings and hall allocations. * Implement a calendar view for better scheduling visualization. |
| **Acceptance Criteria** | * An admin can fill out and submit the exam scheduling form. * The form performs client-side validation. * Upon submission, the form sends a POST request to the backend. * The backend stores the exam schedules in the database. * Confirmation message is displayed upon successful scheduling. |
| **Estimated Points** | 8 |
| **Testing Time** | 4 Hrs. |

**9. Automated Notifications for Exam Scheduling**

|  |  |
| --- | --- |
| **User Story** | **As a teacher, I want to send automated notifications for exam scheduling so that students and parents are informed about exam dates.** |
| **User Story Details** | This story involves creating an interface to send automated notifications for exam schedules. |
| **User Story Output** | Notifications are sent for upcoming exams. |
| **Data Entities Input List** | ExamSchedule (ScheduleId, ExamId, Date, Time, Location, NotificationsSent) |
| **Task** | 1. Create a React component for exam schedule notifications. 2. Integrate with an email/SMS service provider. 3. Create a .NET Core API endpoint to handle notification sending. 4. Connect the React component to the API endpoint using Axios. 5. Store notification logs in the MSSQL database. |
| **Dev Notes** | * Ensure notifications are sent securely and timely. * Implement scheduling for future notifications. |
| **Acceptance Criteria** | * Notifications are sent for upcoming exams. * Notification logs are stored in the database. * Confirmation messages are displayed upon successful sending. |
| **Estimated Points** | 8 |
| **Testing Time** | 4 Hrs. |

**10. Sponsor Registration and Profile Management**

|  |  |
| --- | --- |
| **User Story** | **As an admin, I want to manage sponsor registrations and profiles so that sponsors can support students.** |
| **User Story Details** | This story involves creating interfaces for sponsor sign-up, profile management, and admin-level sponsor management. |
| **User Story Output** | Sponsors can register, log in, and manage their profiles. Admins can manage sponsor records. |
| **Data Entities Input List** | Sponsor (SponsorId, Name, Email, Phone, Address, RegistrationDate, LoginCredentials) |
| **Task** | 1. Create a React component for sponsor sign-up form. 2. Implement sponsor login functionality. 3. Create a .NET Core API endpoint to handle sponsor registration. 4. Create a .NET Core API endpoint to handle login and authentication. 5. Create a React component for sponsor profile management. 6. Connect the sponsor registration and profile forms to the API using Axios. 7. Store sponsor information in the MSSQL database. 8. Implement admin functionalities for adding and deleting sponsors. |
| **Dev Notes** | * Ensure proper validation for email and phone number. * Use secure password hashing for login credentials. |
| **Acceptance Criteria** | * A sponsor can register and create a profile. * A sponsor can log in using their credentials. * A sponsor can edit and save their profile. * Admins can add and delete sponsors. * Confirmation messages are displayed upon successful actions. |
| **Estimated Points** | 6 |
| **Testing Time** | 3 Hrs. |

**11. Student Sponsorship**

|  |  |
| --- | --- |
| **User Story** | **As a sponsor, I want to manage student sponsorships so that I can support students financially.** |
| **User Story Details** | This story involves creating an interface for sponsors to add students to their sponsorship cart and manage payment schedules. |
| **User Story Output** | Sponsors can add/delete students from their cart and manage sponsorship details. |
| **Data Entities Input List** | Sponsorship (SponsorshipId, SponsorId, StudentId, PaymentSchedule, SponsorshipDetails) |
| **Task** | 1. Create a React component for adding students to the sponsorship cart. 2. Implement functionality for deleting students from the sponsorship cart. 3. Create a .NET Core API endpoint to handle student sponsorship management. 4. Connect the React component to the API using Axios. 5. Store sponsorship details in the MSSQL database. 6. Implement payment schedule management. 7. Create functionality to edit and update sponsorship details. |
| **Dev Notes** | * Ensure the payment schedule is flexible (monthly, quarterly, annually). * Implement notifications for upcoming payments. |
| **Acceptance Criteria** | * A sponsor can add students to their sponsorship cart. * A sponsor can delete students from their sponsorship cart. * Sponsorship details are stored and managed in the database. * Payment schedules can be set and managed. * Sponsorship details can be edited and updated. * Confirmation messages are displayed upon successful actions. |
| **Estimated Points** | 6 |
| **Testing Time** | 3 Hrs. |

**12. Leave Management for Teachers**

|  |  |
| --- | --- |
| **User Story** | **As a teacher, I want to manage my leave requests so that my absences are properly recorded.** |
| **User Story Details** | This story involves creating an interface for teachers to submit and manage leave requests. |
| **User Story Output** | Teachers can submit and manage their leave requests. |
| **Data Entities Input List** | LeaveRequest (RequestId, TeacherId, StartDate, EndDate, Reason, Status) |
| **Task** | 1. Create a React component for leave request submission. 2. Create a .NET Core API endpoint to handle leave request management. 3. Connect the React component to the API endpoint using Axios. 4. Store leave request details in the MSSQL database. |
| **Dev Notes** | * Ensure proper validation for leave request details. * Implement approval workflow for leave requests. |
| **Acceptance Criteria** | * Teachers can submit leave requests. * Teachers can manage their leave requests. * Leave request details are stored and managed in the database. * Confirmation messages are displayed upon successful actions. |
| **Estimated Points** | 6 |
| **Testing Time** | 3 Hrs. |

**13. Event Creation and Management**

|  |  |
| --- | --- |
| **User Story** | **As an admin, I want to manage events and programs so that they can be scheduled and tracked.** |
| **User Story Details** | This story involves creating an interface to schedule events, set up event folders, and upload images. |
| **User Story Output** | Events are scheduled, managed, and images are uploaded. |
| **Data Entities Input List** | Event (EventId, Title, Description, Date, Time, Location, Images) |
| **Task** | 1. Create a React component for event creation form. 2. Implement a calendar view for scheduling events. 3. Create a .NET Core API endpoint to handle event creation and management. 4. Connect the React form to the API endpoint using Axios. 5. Store event details in the MSSQL database. 6. Create functionality to set up event folders. 7. Implement image upload functionality for events. |
| **Dev Notes** | * Use a file storage solution for managing event images. * Ensure proper validation for event dates and times. |
| **Acceptance Criteria** | * An admin can fill out and submit the event creation form. * Events are displayed on a calendar view. * Event details are stored in the database. * Event folders can be set up. * Images can be uploaded and associated with events. * Confirmation messages are displayed upon successful actions. |
| **Estimated Points** | 6 |
| **Testing Time** | 3 Hrs. |

**14. Payroll Management**

|  |  |
| --- | --- |
| **User Story** | **As an HR manager, I want to process salaries and bonuses so that staff is compensated accurately.** |
| **User Story Details** | This story involves creating an interface to manage payroll processing. |
| **User Story Output** | Salaries and bonuses are processed and recorded. |
| **Data Entities Input List** | Payroll (PayrollId, StaffId, Salary, Bonus, Date) |
| **Task** | 1. Create a React component for the payroll processing form. 2. Implement form validation. 3. Create a .NET Core API endpoint to handle payroll processing. 4. Connect the React form to the API endpoint using Axios. 5. Store payroll records in the MSSQL database. |
| **Dev Notes** | * Include fields for deductions and allowances. * Consider different pay periods (weekly, bi-weekly, monthly). |
| **Acceptance Criteria** | * An HR manager can fill out and submit the payroll processing form. * The form performs client-side validation. * Upon submission, the form sends a POST request to the backend. * The backend stores the payroll records in the database. * Confirmation message is displayed upon successful payroll processing. |
| **Estimated Points** | 8 |
| **Testing Time** | 4 Hrs. |

**15. Parent Portal**

|  |  |
| --- | --- |
| **User Story** | **As a parent, I want to track my child's progress and attendance so that I can stay informed.** |
| **User Story Details** | This story involves creating a parent portal interface. |
| **User Story Output** | Parents can view their child's progress and attendance. |
| **Data Entities Input List** | Student, Attendance, Grades |
| **Task** | 1. Create a React component for the parent portal. 2. Implement data fetching from the backend. 3. Create a .NET Core API endpoint to fetch student progress and attendance. 4. Connect the React portal to the API endpoint using Axios. |
| **Dev Notes** | * Ensure data privacy and security. * Implement a notification system for important updates. |
| **Acceptance Criteria** | * Parents can log in and view their child's progress and attendance. * The portal displays accurate and up-to-date information. * Parents can receive notifications for important updates. |
| **Estimated Points** | 6 |
| **Testing Time** | 3 Hrs. |

**16. Communication System**

|  |  |
| --- | --- |
| **User Story** | **As an admin, I want to send automated emails and SMS so that I can notify users about important updates.** |
| **User Story Details** | This story involves integrating email and SMS APIs. |
| **User Story Output** | Automated emails and SMS are sent to users. |
| **Data Entities Input List** | Notifications (NotificationId, UserId, Message, Date) |
| **Task** | 1. Integrate an email/SMS API (e.g., Twilio, SendGrid). 2. Create a .NET Core API endpoint to send notifications. 3. Connect the API endpoint with the notification triggers in the application. |
| **Dev Notes** | * Ensure compliance with email/SMS regulations. * Implement logging for sent notifications. |
| **Acceptance Criteria** | * Automated emails and SMS are sent for important updates. * Notifications are logged in the database. |
| **Estimated Points** | 6 |
| **Testing Time** | 3 Hrs. |

**17. Security and Access Control**

|  |  |
| --- | --- |
| **User Story** | **As an admin, I want to manage user roles and permissions so that access control is maintained.** |
| **User Story Details** | This story involves creating an access control system. |
| **User Story Output** | Users have appropriate roles and permissions. |
| **Data Entities Input List** | Users (UserId, Role, Permissions) |
| **Task** | 1. Implement role-based access control (RBAC). 2. Create a .NET Core API endpoint to manage roles and permissions. 3. Integrate RBAC with the frontend. |
| **Dev Notes** | * Use JWT/Identity for authentication and authorization. * Ensure secure password storage (e.g., hashing). |
| **Acceptance Criteria** | * Users have roles and permissions assigned. * Access control is enforced throughout the application. |
| **Estimated Points** | 5 |
| **Testing Time** | 2 Hrs. |

**18. Reporting and Analytics**

|  |  |
| --- | --- |
| **User Story** | **As an admin, I want to generate various reports so that I can analyze data and performance.** |
| **User Story Details** | This story involves creating a reporting and analytics interface. |
| **User Story Output** | Custom reports and analytics are generated. |
| **Data Entities Input List** | Various (e.g., Attendance, Grades, Fees) |
| **Task** | 1. Create a React component for generating reports. 2. Implement data fetching and visualization (e.g., charts). 3. Create a .NET Core API endpoint to fetch data for reports. 4. Connect the React component to the API endpoint using Axios. |
| **Dev Notes** | * Consider different report formats (PDF, Excel). * Implement data export functionality. |
| **Acceptance Criteria** | * Admins can generate custom reports. * Data is accurately visualized and exported. |
| **Estimated Points** | 7 |
| **Testing Time** | 3 Hrs. |