Zuno School Hub Admin Backend - API Documentation

Base URL

`http://localhost:3000/api`

Authentication

Most endpoints require authentication via JWT token in the Authorization header:

```
Authorization: Bearer <token>
```

Common Response Format

All API responses follow this format:

```
{
  "success": true/false,
  "data": { ... },
  "message": "Optional message",
  "error": "Error message if success is false"
}
```

Common Query Parameters

- `page` : Page number (default: 1)
- `limit` : Items per page (default: 10, max: 100)
- `sortBy` : Field to sort by
- `sortOrder` : 'asc' or 'desc' (default: 'desc')

Authentication APIs

1. Login

POST `/auth/login`

Request Body:

```
{
  "email": "user@example.com",
  "password": "password123"
}
```

Response:

```
{
  "success": true,
  "data": {
```

```
    "token": "jwt_token_here",
    "user": {
      "id": "user_id",
      "email": "user@example.com",
      "firstName": "John",
      "lastName": "Doe",
      "role": "admin"
    }
  }
}
```

2. Register

POST `/auth/register`

Request Body:

```
{
  "email": "user@example.com",
  "password": "password123",
  "firstName": "John",
  "lastName": "Doe",
  "role": "admin"
}
```

Response:

```
{
  "success": true,
  "data": {
    "id": "user_id",
    "email": "user@example.com",
    "firstName": "John",
    "lastName": "Doe",
    "role": "admin"
  }
}
```

3. Logout

POST `/auth/logout`

Request Body: None

Response:

```
{
  "success": true,
  "message": "Logged out successfully"
}
```

4. Get Profile

GET `/auth/me`

Response:

```
{
  "success": true,
  "data": {
    "id": "user_id",
    "email": "user@example.com",
    "firstName": "John",
    "lastName": "Doe",
    "role": "admin"
  }
}
```

5. Refresh Token

POST `/auth/refresh`

Response:

```
{
  "success": true,
  "data": {
    "token": "new_jwt_token"
  }
}
```

6. Forgot Password

POST `/auth/forgot-password`

Request Body:

```
{
  "email": "user@example.com"
}
```

Response:

```
{
  "success": true,
  "message": "Password reset email sent"
}
```

7. Reset Password

POST `/auth/reset-password`

Request Body:

```
{
  "token": "reset_token",
  "newPassword": "newpassword123"
}
```

Response:

```
{
  "success": true,
  "message": "Password reset successfully"
}
```

Student APIs

1. Get All Students

GET `/students`

Query Parameters:

- `page` : Page number
- `limit` : Items per page
- `sortBy` : Field to sort by
- `sortOrder` : Sort order

Response:

```
{
  "success": true,
  "data": {
    "students": [
      {
        "_id": "student_id",
        "studentId": "ST001",
        "firstName": "John",
        "lastName": "Doe",
        "email": "john@example.com",
        "phone": "1234567890",
        "dateOfBirth": "2005-01-01",
        "gender": "male",
        "address": {
          "street": "123 Main St",
          "city": "City",
          "state": "State",
          "zipCode": "12345",
          "country": "Country"
        },
        "parentInfo": {
          "fatherName": "Father Name",
          "motherName": "Mother Name",
```

```
          "contactNumber": "1234567890",
          "email": "parent@example.com"
        },
        "class": "10",
        "section": "A",
        "rollNumber": "001",
        "admissionDate": "2020-04-01",
        "isActive": true,
        "createdAt": "2024-01-01T00:00:00.000Z",
        "updatedAt": "2024-01-01T00:00:00.000Z"
      }
    ],
    "pagination": {
      "page": 1,
      "limit": 10,
      "total": 100,
      "pages": 10
    }
  }
}
```

2. Get Student by ID

GET `/students/:id`

Response:

```
{
  "success": true,
  "data": {
    "_id": "student_id",
    "studentId": "ST001",
    "firstName": "John",
    "lastName": "Doe",
    // ... full student object
  }
}
```

3. Create Student

POST `/students`

Authorization: Required (admin, staff)

Request Body:

```
{
  "studentId": "ST001",
  "firstName": "John",
  "lastName": "Doe",
  "email": "john@example.com",
  "phone": "1234567890",
  "dateOfBirth": "2005-01-01",
```

```
  "gender": "male",
  "address": {
    "street": "123 Main St",
    "city": "City",
    "state": "State",
    "zipCode": "12345",
    "country": "Country"
  },
  "parentInfo": {
    "fatherName": "Father Name",
    "motherName": "Mother Name",
    "contactNumber": "1234567890",
    "email": "parent@example.com"
  },
  "class": "10",
  "section": "A",
  "rollNumber": "001",
  "admissionDate": "2020-04-01",
  "bloodGroup": "A+",
  "medicalConditions": ["Asthma"],
  "emergencyContact": {
    "name": "Emergency Contact",
    "relationship": "Uncle",
    "phone": "9876543210"
  }
}
```

Response:

```
{
  "success": true,
  "data": {
    "id": "student_id",
    // ... complete student object
  }
}
```

4. Update Student

PUT `/students/:id`

Authorization: Required (admin, staff)

Request Body: Same as create, all fields optional

Response:

```
{
  "success": true,
  "data": {
    // ... updated student object
  }
}
```

5. Delete Student

DELETE `/students/:id`

Authorization: Required (admin)

Response:

```
{
  "success": true,
  "message": "Student deleted successfully"
}
```

6. Get Students by Class

GET `/students/class/:className`

Response:

```
{
  "success": true,
  "data": [
    // ... array of students
  ]
}
```

7. Search Students

GET `/students/search`

Query Parameters:

- `q` : Search query
- `class` : Filter by class
- `section` : Filter by section

Response:

```
{
  "success": true,
  "data": [
    // ... array of matching students
  ]
}
```

8. Upload Student Photo

POST `/students/:id/upload-photo`

Authorization: Required (admin, staff)

Request Body: Form data with photo file

Response:

```
{
  "success": true,
  "data": {
    "photoUrl": "path/to/photo.jpg"
  }
}
```

Teacher APIs

1. Get All Teachers

GET `/teachers`

Query Parameters: Same as students

Response:


```
{
  "success": true,
  "data": {
    "teachers": [
      {
        "_id": "teacher_id",
        "teacherId": "T001",
        "firstName": "Jane",
        "lastName": "Smith",
        "email": "jane@example.com",
        "phone": "1234567890",
        "dateOfBirth": "1980-01-01",
        "gender": "female",
        "address": {
          "street": "456 Oak St",
          "city": "City",
          "state": "State",
          "zipCode": "12345",
          "country": "Country"
        },
        "qualifications": ["B.Ed", "M.A"],
        "subjects": ["Mathematics", "Physics"],
        "experience": 5,
        "joiningDate": "2020-01-01",
        "salary": 50000,
        "isActive": true,
        "emergencyContact": {
          "name": "Emergency Contact",
          "relationship": "Spouse",
          "phone": "9876543210"
        },
        "createdAt": "2024-01-01T00:00:00.000Z",
        "updatedAt": "2024-01-01T00:00:00.000Z"
      }
```

```
    ],
    "pagination": {
      "page": 1,
      "limit": 10,
      "total": 50,
      "pages": 5
    }
  }
}
```

2. Get Teacher by ID

GET `/teachers/:id`

Response:

```
{
  "success": true,
  "data": {
    // ... full teacher object
  }
}
```

3. Create Teacher

POST `/teachers`

Authorization: Required (admin)

Request Body:

```
{
  "teacherId": "T001",
  "firstName": "Jane",
  "lastName": "Smith",
  "email": "jane@example.com",
  "phone": "1234567890",
  "dateOfBirth": "1980-01-01",
  "gender": "female",
  "address": {
    "street": "456 Oak St",
    "city": "City",
    "state": "State",
    "zipCode": "12345",
    "country": "Country"
  },
  "qualifications": ["B.Ed", "M.A"],
  "subjects": ["Mathematics", "Physics"],
  "experience": 5,
  "joiningDate": "2020-01-01",
  "salary": 50000,
  "emergencyContact": {
    "name": "Emergency Contact",
```

```
    "relationship": "Spouse",
    "phone": "9876543210"
  },
  "bankDetails": {
    "accountNumber": "1234567890",
    "bankName": "Bank Name",
    "ifscCode": "BANK0001234"
  }
}
```

Response:

```
{
  "success": true,
  "data": {
    "id": "teacher_id",
    // ... complete teacher object
  }
}
```

4. Update Teacher

PUT `/teachers/:id`

Authorization: Required (admin)

Request Body: Same as create, all fields optional

Response:

```
{
  "success": true,
  "data": {
    // ... updated teacher object
  }
}
```

5. Delete Teacher

DELETE `/teachers/:id`

Authorization: Required (admin)

Response:

```
{
  "success": true,
  "message": "Teacher deleted successfully"
}
```

6. Get Teachers by Subject

GET `/teachers/subject/:subject`

Response:

```
{
  "success": true,
  "data": [
    // ... array of teachers
  ]
}
```

7. Get Teachers by Class

GET `/teachers/class/:classId`

Response:

```
{
  "success": true,
  "data": [
    // ... array of teachers
  ]
}
```

8. Upload Teacher Photo

POST `/teachers/:id/upload-photo`

Authorization: Required (admin)

Request Body: Form data with photo file

Response:

```
{
  "success": true,
  "data": {
    "photoUrl": "path/to/photo.jpg"
  }
}
```

9. Assign Class to Teacher

POST `/teachers/:id/assign-class`

Authorization: Required (admin)

Request Body:

```
{
  "classId": "class_id",
```

```
  "subject": "Mathematics"
}
```

Response:

```
{
  "success": true,
  "message": "Class assigned successfully"
}
```

10. Remove Class from Teacher

POST `/teachers/:id/remove-class`

Authorization: Required (admin)

Request Body:

```
{
  "classId": "class_id"
}
```

Response:

```
{
  "success": true,
  "message": "Class removed successfully"
}
```

Attendance APIs

1. Get All Attendance

GET `/attendance`

Query Parameters: Standard pagination

Response:

```
{
  "success": true,
  "data": {
    "attendance": [
      {
        "_id": "attendance_id",
        "studentId": "student_id",
        "date": "2024-01-01",
        "status": "present",
        "checkInTime": "2024-01-01T09:00:00.000Z",
        "checkOutTime": "2024-01-01T15:00:00.000Z",
        "notes": "On time",
```

```
        "markedBy": "teacher_id",
        "createdAt": "2024-01-01T00:00:00.000Z",
        "updatedAt": "2024-01-01T00:00:00.000Z"
      }
    ],
    "pagination": {
      "page": 1,
      "limit": 10,
      "total": 100,
      "pages": 10
    }
  }
}
```

2. Get Attendance by ID

GET `/attendance/:id`

Response:

```
{
  "success": true,
  "data": {
    // ... full attendance object
  }
}
```

3. Mark Attendance

POST `/attendance`

Authorization: Required (admin, teacher, staff)

Request Body:

```
{
  "studentId": "student_id",
  "date": "2024-01-01",
  "status": "present",
  "checkInTime": "2024-01-01T09:00:00.000Z",
  "checkOutTime": "2024-01-01T15:00:00.000Z",
  "notes": "On time"
}
```

Response:

```
{
  "success": true,
  "data": {
    "id": "attendance_id",
    // ... complete attendance object
```

```
  }
}
```

4. Update Attendance

PUT `/attendance/:id`

Authorization: Required (admin, teacher, staff)

Request Body:

```
{
  "status": "absent",
  "notes": "Sick leave"
}
```

Response:

```
{
  "success": true,
  "data": {
    // ... updated attendance object
  }
}
```

5. Delete Attendance

DELETE `/attendance/:id`

Authorization: Required (admin)

Response:

```
{
  "success": true,
  "message": "Attendance deleted successfully"
}
```

6. Get Student Attendance

GET `/attendance/student/:studentId`

Response:

```
{
  "success": true,
  "data": [
    // ... array of attendance records for student
  ]
}
```

7. Get Class Attendance

GET `/attendance/class/:className`

Response:

```
{
  "success": true,
  "data": [
    // ... array of attendance records for class
  ]
}
```

8. Get Attendance by Date

GET `/attendance/date/:date`

Response:

```
{
  "success": true,
  "data": [
    // ... array of attendance records for date
  ]
}
```

9. Get Attendance Report

GET `/attendance/report/:studentId`

Response:

```
{
  "success": true,
  "data": {
    "studentId": "student_id",
    "studentName": "John Doe",
    "totalDays": 100,
    "presentDays": 85,
    "absentDays": 10,
    "lateDays": 5,
    "excusedDays": 0,
    "attendancePercentage": 85
  }
}
```

10. Bulk Mark Attendance

POST `/attendance/bulk`

Authorization: Required (admin, teacher, staff)

Request Body:

```
{
  "date": "2024-01-01",
  "attendance": [
    {
      "studentId": "student_id_1",
      "status": "present",
      "checkInTime": "2024-01-01T09:00:00.000Z"
    },
    {
      "studentId": "student_id_2",
      "status": "absent",
      "notes": "Sick"
    }
  ]
}
```

Response:

```
{
  "success": true,
  "data": {
    "marked": 2,
    "failed": 0
  }
}
```

Communication APIs

1. Get All Communications

GET `/communication`

Query Parameters: Standard pagination

Response:

```
{
  "success": true,
  "data": {
    "communications": [
      {
        "_id": "communication_id",
        "type": "announcement",
        "title": "School Holiday",
        "content": "School will be closed tomorrow",
        "sender": "admin_id",
        "recipients": {
          "type": "all",
          "targetIds": [],
```

```
          "classes": [],
          "sections": []
        },
        "priority": "high",
        "attachments": [],
        "scheduledDate": "2024-01-01T00:00:00.000Z",
        "expiryDate": "2024-01-07T00:00:00.000Z",
        "isActive": true,
        "readBy": [],
        "createdAt": "2024-01-01T00:00:00.000Z",
        "updatedAt": "2024-01-01T00:00:00.000Z"
      }
    ],
    "pagination": {
      "page": 1,
      "limit": 10,
      "total": 50,
      "pages": 5
    }
  }
}
```


2. Get Communication by ID

GET `/communication/:id`

Response:


```
{
  "success": true,
  "data": {
    // ... full communication object
  }
}
```


3. Create Communication

POST `/communication`

Authorization: Required (admin, teacher, staff)

Request Body:


```
{
  "type": "announcement",
  "title": "School Holiday",
  "content": "School will be closed tomorrow",
  "recipients": {
    "type": "all",
    "targetIds": [],
    "classes": ["10", "11"],
    "sections": ["A", "B"]
  },
```

```
  "priority": "high",
  "attachments": ["file1.pdf", "file2.jpg"],
  "scheduledDate": "2024-01-01T00:00:00.000Z",
  "expiryDate": "2024-01-07T00:00:00.000Z"
}
```

Response:

```
{
  "success": true,
  "data": {
    "id": "communication_id",
    // ... complete communication object
  }
}
```

4. Update Communication

PUT `/communication/:id`

Authorization: Required (admin, teacher, staff)

Request Body: Same as create, all fields optional

Response:

```
{
  "success": true,
  "data": {
    // ... updated communication object
  }
}
```

5. Delete Communication

DELETE `/communication/:id`

Authorization: Required (admin)

Response:

```
{
  "success": true,
  "message": "Communication deleted successfully"
}
```

6. Get Communications by Type

GET `/communication/type/:type`

Response:

```
{
  "success": true,
  "data": [
    // ... array of communications
  ]
}
```

7. Get Unread Communications

GET `/communication/unread/:userId`

Response:

```
{
  "success": true,
  "data": [
    // ... array of unread communications
  ]
}
```

8. Mark Communication as Read

POST `/communication/:id/read`

Response:

```
{
  "success": true,
  "message": "Communication marked as read"
}
```

9. Get Communications by Recipient

GET `/communication/recipient/:recipientId`

Response:

```
{
  "success": true,
  "data": [
    // ... array of communications for recipient
  ]
}
```

Fee APIs

1. Get All Fees

GET `/fees`

Query Parameters: Standard pagination

Response:


```
{
  "success": true,
  "data": {
    "fees": [
      {
        "_id": "fee_id",
        "studentId": "student_id",
        "feeType": "tuition",
        "amount": 5000,
        "dueDate": "2024-01-31",
        "status": "pending",
        "description": "Monthly tuition fee",
        "academicYear": "2024-25",
        "month": "January",
        "payments": [],
        "discounts": [],
        "createdAt": "2024-01-01T00:00:00.000Z",
        "updatedAt": "2024-01-01T00:00:00.000Z"
      }
    ],
    "pagination": {
      "page": 1,
      "limit": 10,
      "total": 100,
      "pages": 10
    }
  }
}
```


2. Get Fee by ID

GET `/fees/:id`

Response:


```
{
  "success": true,
  "data": {
    // ... full fee object
  }
}
```


3. Create Fee

POST `/fees`

Authorization: Required (admin, staff)

Request Body:

```
{
  "studentId": "student_id",
  "feeType": "tuition",
  "amount": 5000,
  "dueDate": "2024-01-31",
  "description": "Monthly tuition fee",
  "academicYear": "2024-25",
  "month": "January"
}
```

Response:

```
{
  "success": true,
  "data": {
    "id": "fee_id",
    // ... complete fee object
  }
}
```

4. Update Fee

PUT `/fees/:id`

Authorization: Required (admin, staff)

Request Body: Same as create, all fields optional

Response:

```
{
  "success": true,
  "data": {
    // ... updated fee object
  }
}
```

5. Delete Fee

DELETE `/fees/:id`

Authorization: Required (admin)

Response:

```
{
  "success": true,
  "message": "Fee deleted successfully"
}
```

6. Get Fees by Student

GET `/fees/student/:studentId`

Response:

```
{
  "success": true,
  "data": [
    // ... array of fees for student
  ]
}
```

7. Get Fees by Class

GET `/fees/class/:classId`

Response:

```
{
  "success": true,
  "data": [
    // ... array of fees for class
  ]
}
```

8. Get Pending Fees

GET `/fees/pending`

Response:

```
{
  "success": true,
  "data": [
    // ... array of pending fees
  ]
}
```

9. Get Overdue Fees

GET `/fees/overdue`

Response:

```
{
  "success": true,
  "data": [
    // ... array of overdue fees
  ]
}
```

10. Pay Fee

POST `/fees/:id/pay`

Authorization: Required (admin, staff)

Request Body:

```
{
  "amount": 5000,
  "paymentMethod": "cash",
  "transactionId": "TXN123456",
  "notes": "Full payment"
}
```

Response:

```
{
  "success": true,
  "data": {
    "receiptNumber": "REC123456",
    "paymentDate": "2024-01-01T00:00:00.000Z",
    "amount": 5000
  }
}
```

11. Generate Fee Report

GET `/fees/reports`

Response:

```
{
  "success": true,
  "data": {
    "totalFees": 100000,
    "collectedFees": 75000,
    "pendingFees": 25000,
    "overdueFees": 10000
  }
}
```

12. Send Fee Reminder

POST `/fees/:id/reminder`

Authorization: Required (admin, staff)

Response:

```
{
  "success": true,
```

```
  "message": "Fee reminder sent successfully"
}
```

13. Apply Discount

POST `/fees/:id/discount`

Authorization: Required (admin)

Request Body:

```
{
  "type": "percentage",
  "value": 10,
  "reason": "Scholarship"
}
```

Response:

```
{
  "success": true,
  "data": {
    "originalAmount": 5000,
    "discountAmount": 500,
    "finalAmount": 4500
  }
}
```

14. Get Fee Statistics

GET `/fees/statistics`

Response:

```
{
  "success": true,
  "data": {
    "totalStudents": 500,
    "totalFees": 100000,
    "collectedFees": 75000,
    "pendingFees": 25000,
    "collectionRate": 75
  }
}
```

Homework APIs

1. Get All Homework

GET `/homework`

Query Parameters: Standard pagination

Response:


```
{
  "success": true,
  "data": {
    "homework": [
      {
        "_id": "homework_id",
        "title": "Mathematics Assignment",
        "description": "Complete exercises 1-10",
        "subject": "Mathematics",
        "class": "10",
        "section": "A",
        "assignedBy": "teacher_id",
        "assignedDate": "2024-01-01",
        "dueDate": "2024-01-07",
        "totalMarks": 100,
        "attachments": ["assignment.pdf"],
        "instructions": "Show all working",
        "isActive": true,
        "createdAt": "2024-01-01T00:00:00.000Z",
        "updatedAt": "2024-01-01T00:00:00.000Z"
      }
    ],
    "pagination": {
      "page": 1,
      "limit": 10,
      "total": 50,
      "pages": 5
    }
  }
}
```


2. Get Homework by ID

GET `/homework/:id`

Response:


```
{
  "success": true,
  "data": {
    // ... full homework object
  }
}
```


3. Create Homework

POST `/homework`

Authorization: Required (admin, teacher)

Request Body:

```
{
  "title": "Mathematics Assignment",
  "description": "Complete exercises 1-10",
  "subject": "Mathematics",
  "class": "10",
  "section": "A",
  "dueDate": "2024-01-07",
  "totalMarks": 100,
  "attachments": ["assignment.pdf"],
  "instructions": "Show all working"
}
```

Response:

```
{
  "success": true,
  "data": {
    "id": "homework_id",
    // ... complete homework object
  }
}
```

4. Update Homework

PUT `/homework/:id`

Authorization: Required (admin, teacher)

Request Body: Same as create, all fields optional

Response:

```
{
  "success": true,
  "data": {
    // ... updated homework object
  }
}
```

5. Delete Homework

DELETE `/homework/:id`

Authorization: Required (admin, teacher)

Response:

```
{
  "success": true,
```

```
  "message": "Homework deleted successfully"
}
```

6. Get Homework by Class

GET `/homework/class/:classId`

Response:

```
{
  "success": true,
  "data": [
    // ... array of homework for class
  ]
}
```

7. Get Homework by Subject

GET `/homework/subject/:subject`

Response:

```
{
  "success": true,
  "data": [
    // ... array of homework for subject
  ]
}
```

8. Get Homework by Teacher

GET `/homework/teacher/:teacherId`

Response:

```
{
  "success": true,
  "data": [
    // ... array of homework by teacher
  ]
}
```

9. Submit Homework

POST `/homework/:id/submit`

Request Body:

```
{
  "studentId": "student_id",
  "attachments": ["submission.pdf"],
```

```
  "notes": "Completed all questions"
}
```

Response:

```
{
  "success": true,
  "data": {
    "submissionId": "submission_id",
    "submissionDate": "2024-01-05T00:00:00.000Z",
    "status": "submitted"
  }
}
```

10. Grade Homework

POST `/homework/:id/grade`

Authorization: Required (admin, teacher)

Request Body:

```
{
  "studentId": "student_id",
  "marksObtained": 85,
  "feedback": "Good work, but check question 7"
}
```

Response:

```
{
  "success": true,
  "data": {
    "marksObtained": 85,
    "totalMarks": 100,
    "percentage": 85,
    "grade": "A"
  }
}
```

11. Get Homework Due Today

GET `/homework/due-today`

Response:

```
{
  "success": true,
  "data": [
    // ... array of homework due today
```

```
  ]
}
```

12. Get Overdue Homework

GET `/homework/overdue`

Response:

```
{
  "success": true,
  "data": [
    // ... array of overdue homework
  ]
}
```

13. Upload Homework File

POST `/homework/:id/upload`

Authorization: Required (admin, teacher)

Request Body: Form data with file

Response:

```
{
  "success": true,
  "data": {
    "fileUrl": "path/to/file.pdf",
    "fileName": "assignment.pdf"
  }
}
```

Report APIs

1. Get Dashboard Data

GET `/reports/dashboard`

Response:

```
{
  "success": true,
  "data": {
    "totalStudents": 500,
    "totalTeachers": 50,
    "totalClasses": 20,
    "attendanceToday": 450,
    "pendingFees": 25000,
    "overdueHomework": 15,
```

```
    "recentActivities": [
      {
        "type": "student_admission",
        "message": "New student John Doe admitted",
        "timestamp": "2024-01-01T00:00:00.000Z"
      }
    ]
  }
}
```

2. Get Student Report

GET `/reports/students`

Query Parameters: Standard pagination

Response:

```
{
  "success": true,
  "data": {
    "students": [
      {
        "studentId": "ST001",
        "name": "John Doe",
        "class": "10",
        "section": "A",
        "attendancePercentage": 95,
        "feeStatus": "paid",
        "academicPerformance": "excellent"
      }
    ],
    "pagination": {
      "page": 1,
      "limit": 10,
      "total": 500,
      "pages": 50
    }
  }
}
```

3. Get Teacher Report

GET `/reports/teachers`

Query Parameters: Standard pagination

Response:

```
{
  "success": true,
  "data": {
    "teachers": [
```

```
      {
        "teacherId": "T001",
        "name": "Jane Smith",
        "subjects": ["Mathematics", "Physics"],
        "classes": ["10A", "10B"],
        "experience": 5,
        "performance": "excellent"
      }
    ],
    "pagination": {
      "page": 1,
      "limit": 10,
      "total": 50,
      "pages": 5
    }
  }
}
```

4. Get Attendance Report

GET `/reports/attendance`

Query Parameters:

- `startDate` : Start date for report
- `endDate` : End date for report
- `class` : Filter by class
- `section` : Filter by section

Response:

```
{
  "success": true,
  "data": {
    "overall": {
      "totalStudents": 500,
      "averageAttendance": 85,
      "presentToday": 450,
      "absentToday": 50
    },
    "classwiseData": [
      {
        "class": "10A",
        "totalStudents": 30,
        "attendancePercentage": 90,
        "presentToday": 27,
        "absentToday": 3
      }
    ]
  }
}
```

5. Get Student Attendance Report

GET `/reports/attendance/student/:studentId`

Response:

```
{
  "success": true,
  "data": {
    "studentId": "student_id",
    "studentName": "John Doe",
    "class": "10A",
    "totalDays": 100,
    "presentDays": 85,
    "absentDays": 10,
    "lateDays": 5,
    "attendancePercentage": 85,
    "monthlyData": [
      {
        "month": "January",
        "totalDays": 25,
        "presentDays": 22,
        "attendancePercentage": 88
      }
    ]
  }
}
```

6. Get Class Attendance Report

GET `/reports/attendance/class/:className`

Response:

```
{
  "success": true,
  "data": {
    "className": "10A",
    "totalStudents": 30,
    "attendanceData": [
      {
        "studentName": "John Doe",
        "attendancePercentage": 85,
        "presentDays": 85,
        "absentDays": 10
      }
    ]
  }
}
```

7. Get Fee Report

GET `/reports/fees`

Response:

```
{
  "success": true,
  "data": {
    "totalFees": 100000,
    "collectedFees": 75000,
    "pendingFees": 25000,
    "overdueFees": 10000,
    "collectionRate": 75,
    "monthlyCollection": [
      {
        "month": "January",
        "collected": 15000,
        "pending": 5000
      }
    ]
  }
}
```

8. Get Student Fee Report

GET `/reports/fees/student/:studentId`

Response:

```
{
  "success": true,
  "data": {
    "studentId": "student_id",
    "studentName": "John Doe",
    "totalFees": 5000,
    "paidFees": 3000,
    "pendingFees": 2000,
    "feeHistory": [
      {
        "date": "2024-01-01",
        "amount": 1000,
        "type": "tuition",
        "status": "paid"
      }
    ]
  }
}
```

9. Get Overdue Fee Report

GET `/reports/fees/overdue`

Response:

```
{
  "success": true,
  "data": {
    "totalOverdue": 10000,
    "overdueCount": 25,
    "students": [
      {
        "studentName": "John Doe",
        "class": "10A",
        "overdueAmount": 2000,
        "daysPastDue": 15
      }
    ]
  }
}
```


10. Get Homework Report

GET `/reports/homework`

Response:


```
{
  "success": true,
  "data": {
    "totalHomework": 50,
    "submittedHomework": 35,
    "pendingHomework": 15,
    "overdueHomework": 5,
    "submissionRate": 70
  }
}
```


11. Get Student Homework Report

GET `/reports/homework/student/:studentId`

Response:


```
{
  "success": true,
  "data": {
    "studentId": "student_id",
    "studentName": "John Doe",
    "totalHomework": 10,
    "submittedHomework": 8,
    "pendingHomework": 2,
    "averageGrade": 85,
    "submissions": [
      {
        "subject": "Mathematics",
        "title": "Assignment 1",
```

```
        "submissionDate": "2024-01-05",
        "grade": 85
      }
    ]
  }
}
```

12. Get Class Homework Report

GET `/reports/homework/class/:className`

Response:

```
{
  "success": true,
  "data": {
    "className": "10A",
    "totalHomework": 20,
    "submissionStats": [
      {
        "homeworkTitle": "Math Assignment 1",
        "totalStudents": 30,
        "submitted": 25,
        "pending": 5,
        "submissionRate": 83
      }
    ]
  }
}
```

13. Get Academic Performance Report

GET `/reports/academic-performance`

Response:

```
{
  "success": true,
  "data": {
    "overall": {
      "averageGrade": 78,
      "passRate": 85,
      "excellentPerformers": 45
    },
    "subjectwise": [
      {
        "subject": "Mathematics",
        "averageGrade": 80,
        "passRate": 88,
        "topPerformers": 12
      }
    ]
```

```
  }
}
```

14. Export Student Data

GET `/reports/export/students`

Authorization: Required (admin)

Response: CSV/Excel file download

15. Export Attendance Data

GET `/reports/export/attendance`

Authorization: Required (admin)

Response: CSV/Excel file download

16. Export Fee Data

GET `/reports/export/fees`

Authorization: Required (admin)

Response: CSV/Excel file download

Error Responses

All APIs can return the following error responses:

400 Bad Request

```
{
  "success": false,
  "error": "Invalid request data"
}
```

401 Unauthorized

```
{
  "success": false,
  "error": "Access denied. Token required."
}
```

403 Forbidden

```
{
  "success": false,
```

```
  "error": "Access denied. Insufficient permissions."
}
```

404 Not Found

```
{
  "success": false,
  "error": "Resource not found"
}
```

500 Internal Server Error

```
{
  "success": false,
  "error": "Internal server error"
}
```

Rate Limiting

- All APIs are rate-limited to 100 requests per minute per IP address
- Authentication APIs have stricter limits: 10 requests per minute

File Upload Guidelines

- Maximum file size: 10MB
- Supported formats for photos: jpg, jpeg, png, gif
- Supported formats for documents: pdf, doc, docx, txt
- Files are stored in `/uploads` directory with UUID names

Notes

- All dates are in ISO 8601 format
- All responses include proper HTTP status codes
- MongoDB ObjectIds are used for all entity references
- Pagination is zero-based (page 1 = first page)
- All monetary values are in the smallest currency unit (cents/paise)