Backend

1. backend/controllers/monthlyAttendanceController.js

This controller converts the month and year into start and end dates (in ISO format) and then queries the Attendance collection for records falling in that date range. It also logs the computed boundaries and number of records returned for easier debugging.

javascript

```
// backend/controllers/monthlyAttendanceController.js

const Attendance = require("../models/attendanceModel");


exports.getMonthlyAttendance = async (req, res) => {
  try {
    const { studentId, month, year } = req.query;
    if (!studentId || !month || !year) {
      return res
        .status(400)
        .json({ error: "Please provide studentId, month, and year." });
    }

    // Parse the month and year as integers.
    const m = parseInt(month, 10);
    const y = parseInt(year, 10);
    if (isNaN(m) || isNaN(y)) {
      return res.status(400).json({ error: "Invalid month or year." });
    }

    // Compute start and end dates (using JS Date object)
    // Note: Months in JS Date are 0-indexed.
```

```
const startDate = new Date(y, m - 1, 1);
const endDate = new Date(y, m, 0); // last day of the month

const startStr = startDate.toISOString().split("T")[0];
const endStr = endDate.toISOString().split("T")[0];

console.log("Querying attendance for:", studentId);
console.log("Date range:", startStr, "to", endStr);

// Fetch all attendance records for this student between startStr and endStr.
const records = await Attendance.find({
  studentId,
  date: { $gte: startStr, $lte: endStr },
}).sort({ date: 1 });

console.log(`Found ${records.length} attendance records for ${studentId}.`);

// Build a list with one entry for each day of the month.
let daysInMonth = [];
for (let d = 1; d <= endDate.getDate(); d++) {
  let day = new Date(y, m - 1, d);
  let dayStr = day.toISOString().split("T")[0];
  // Find if an attendance record exists for this date.
  const rec = records.find(r => r.date === dayStr);
  daysInMonth.push({
    date: dayStr,
```

```
      status: rec ? rec.status : "Not Marked"
    });
  }

  res.json(daysInMonth);
 } catch (error) {
  console.error("Error in getMonthlyAttendance:", error);
  res.status(500).json({ error: "Server error retrieving monthly attendance.", details: error.message });
 }
};
```

2. backend/routes/monthlyAttendanceRoutes.js

This file registers the GET route for our monthly attendance API.

```
javascript

// backend/routes/monthlyAttendanceRoutes.js
const express = require("express");
const { getMonthlyAttendance } = require("../controllers/monthlyAttendanceController");

const router = express.Router();

router.get("/monthlyAttendance", getMonthlyAttendance);

module.exports = router;
```

> **Important:** Ensure that in your server.js file you mount this route. For example: > > javascript > // server.js (excerpt) > const monthlyAttendanceRoutes = require("./routes/monthlyAttendanceRoutes"); > // ... > app.use("/api", monthlyAttendanceRoutes); >

Frontend

3. frontend/monthlyAttendance.html

This HTML page provides a simple UI with input fields for student, month, and year.

html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Monthly Attendance View</title>
  <link rel="stylesheet" href="style.css" />
</head>
<body>
  <h1>View Monthly Attendance Datewise</h1>

  <div class="attendance-view-form">
    <div>
      <label for="attStudentList">Select Student: </label>
      <select id="attStudentList"></select>
    </div>
    <div>
      <label for="month">Month: </label>
      <input type="number" id="month" placeholder="Month (1-12)" min="1" max="12" />
    </div>
    <div>
      <label for="year">Year: </label>
```

```
    <input type="number" id="year" placeholder="Year" />
  </div>
  <div>
    <button onclick="viewMonthlyAttendance()">View Attendance</button>
  </div>
 </div>

 <div id="monthlyAttendanceDisplay"></div>

 <script src="script.js"></script>
</body>
</html>
```

4. Updated Sections in frontend/script.js

Below are the functions that load students into the dropdown for attendance selection and view monthly attendance data. This version includes detailed error handling and logging.

javascript

```
// Function to load students for attendance dropdown.
async function loadAttendanceStudents() {
 try {
  const response = await fetch("http://localhost:5000/api/students");
  if (!response.ok) {
   throw new Error("Failed to load students");
  }
  const students = await response.json();
  const studentList = document.getElementById("attStudentList");
  studentList.innerHTML = "";
```

```
    students.forEach(student => {
      const option = document.createElement("option");
      // Ensure using student.studentId if that is used in your schema.
      option.value = student.studentId;
      option.textContent = `${student.name} (${student.class})`;
      studentList.appendChild(option);
    });
  } catch (error) {
    console.error("Error loading attendance students:", error);
  }
}

// Function to view monthly attendance for a selected student.
async function viewMonthlyAttendance() {
  const studentId = document.getElementById("attStudentList").value;
  const month = document.getElementById("month").value;
  const year = document.getElementById("year").value;

  // Check if all fields are provided.
  if (!studentId || !month || !year) {
    alert("Please select a student, and enter month and year correctly.");
    return;
  }

  try {
    // Build the query URL.
```

```
  const url =
`http://localhost:5000/api/monthlyAttendance?studentId=${encodeURIComponent(studentId)}
&month=${encodeURIComponent(month)}&year=${encodeURIComponent(year)}`;
  console.log("Fetching URL:", url);

  const response = await fetch(url);
  if (!response.ok) {
   const errorData = await response.json();
   alert("Error: " + errorData.error);
   return;
  }

  const attendanceData = await response.json();

  // Build a table to display the attendance.
  let output = `<h2>Attendance for ${month}/${year}</h2>`;
  output += "<table><thead><tr><th>Date</th><th>Status</th></tr></thead><tbody>";
  attendanceData.forEach(day => {
   output += `<tr><td>${day.date}</td><td>${day.status}</td></tr>`;
  });
  output += "</tbody></table>";

  document.getElementById("monthlyAttendanceDisplay").innerHTML = output;

 } catch (error) {
  console.error("Error fetching monthly attendance:", error);
  alert("An error occurred while fetching monthly attendance.");
```

```
  }
}

// Load students when the DOM content is loaded.
document.addEventListener("DOMContentLoaded", loadAttendanceStudents);
```

Troubleshooting Steps

1. **Check Console Logs & Network Tab:**
    - Open your browser's Developer Tools (F12) and check for any JavaScript errors.
    - Verify in the Network tab that the GET request to /api/monthlyAttendance is sent with the correct query parameters.
    - Use the logs (console.log) added to the viewMonthlyAttendance() function and the controller to confirm the date range computed and the number of records found.
2. **Verify Date Formats:**
    - Ensure that when you "mark attendance" elsewhere in your application, the date is saved in ISO format (e.g., "YYYY-MM-DD"). The monthly attendance query uses ISO string comparisons.
    - If your attendance records have a different format (for example, "10-Oct-2010"), consider converting them to ISO format or adjust your query accordingly.
3. **Test with Known Data:**
    - Manually check your MongoDB attendance collection (using a GUI like MongoDB Compass) to ensure that attendance records exist for the given student and dates.

Once these updated files are in place and the server is running, select a student, enter a month and year, and click "View Attendance" to see a datewise listing (with records or "Not Marked") for each day of that month.

Feel free to ask if you need further assistance or additional customizations