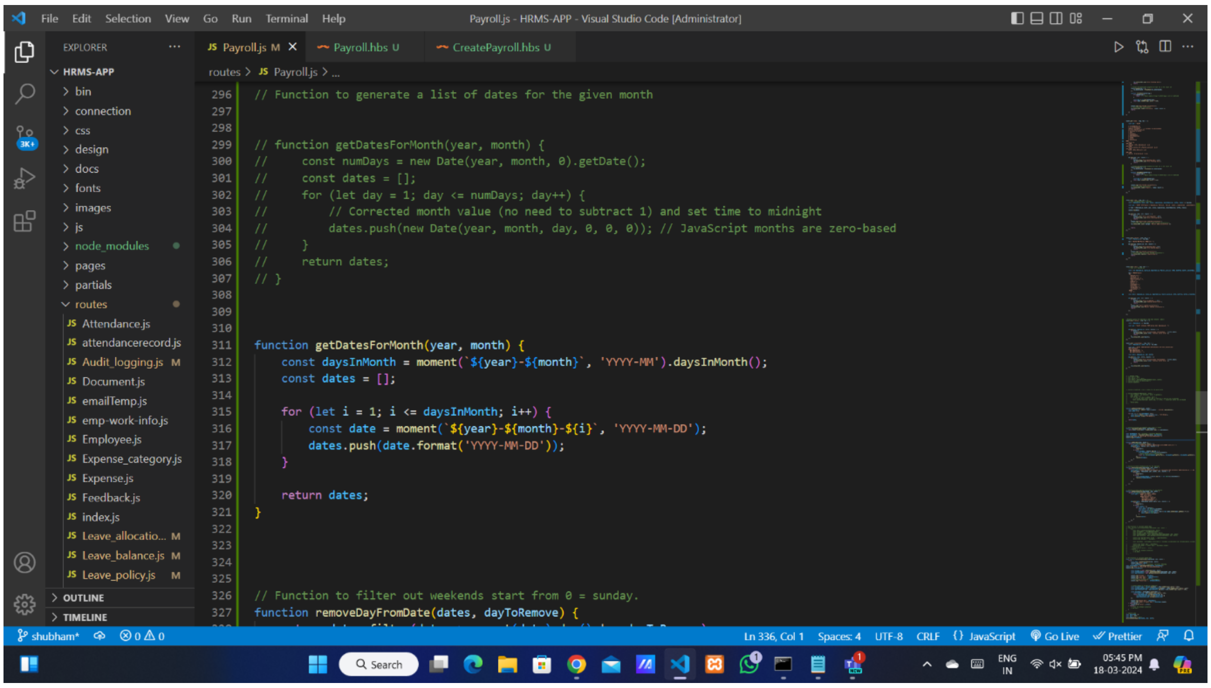
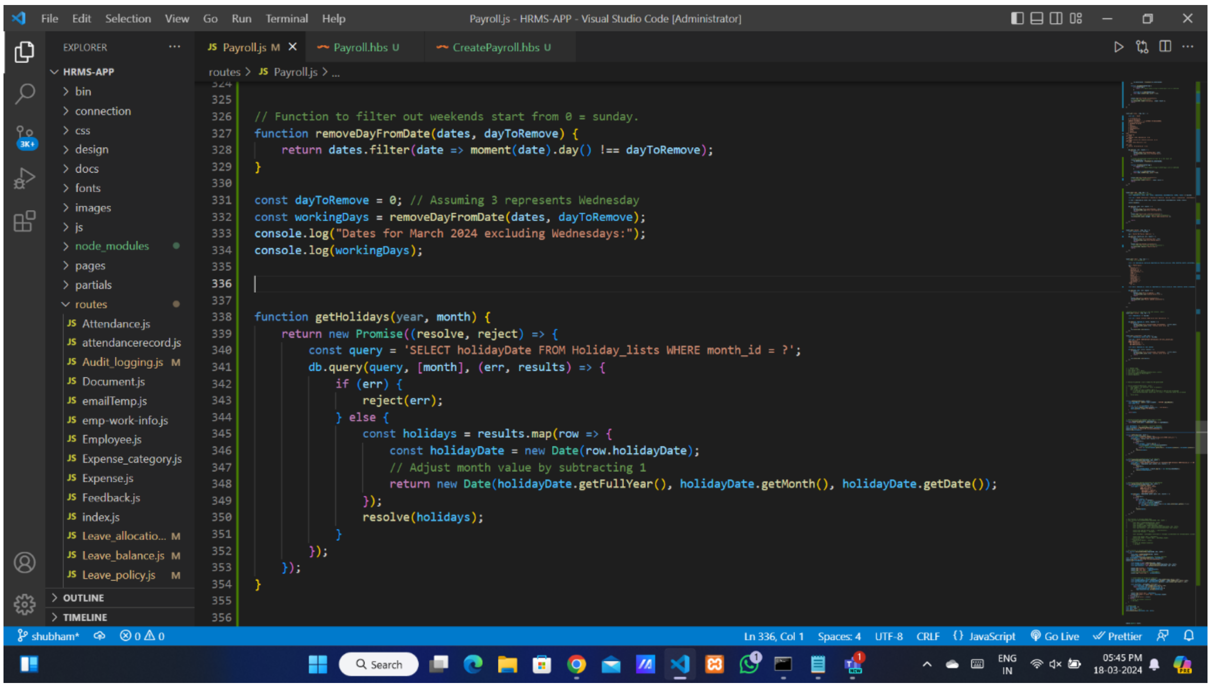
Payroll Calculation

# function to create total dates of particular month for payroll



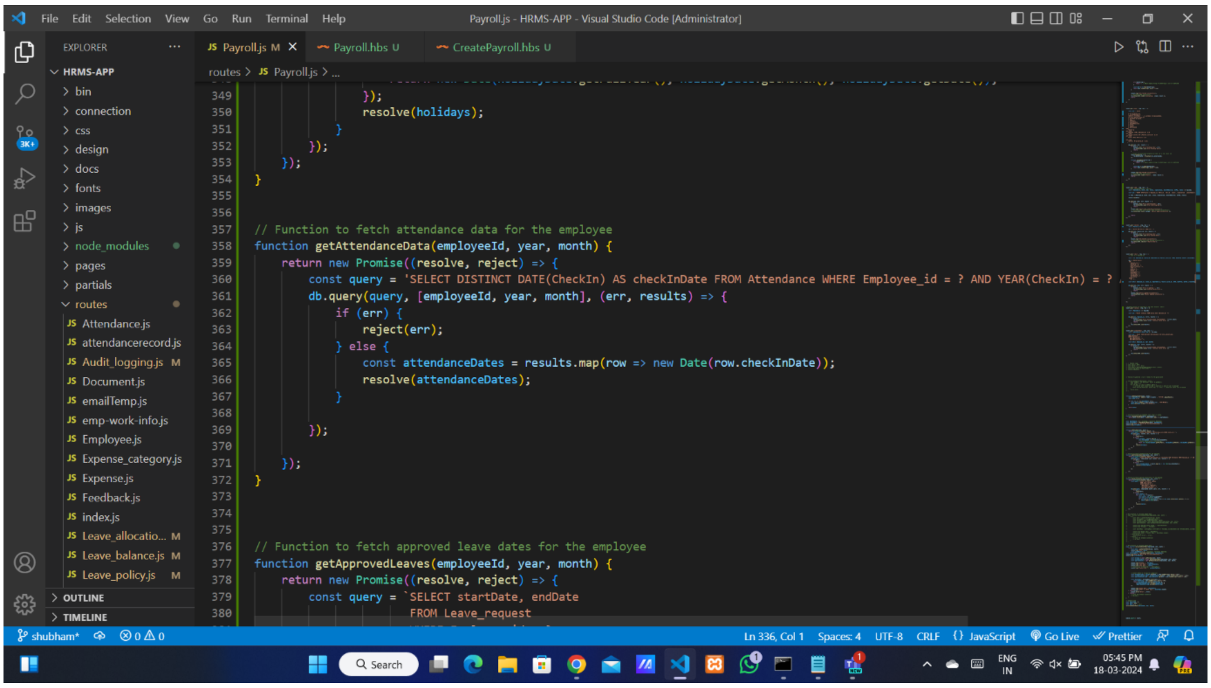
```
// Function to generate a list of dates for the given month


// function getDatesForMonth(year, month) {
//     const numDays = new Date(year, month, 0).getDate();
//     const dates = [];
//     for (let day = 1; day <= numDays; day++) {
//         // Corrected month value (no need to subtract 1) and set time to midnight
//         dates.push(new Date(year, month, day, 0, 0, 0)); // JavaScript months are zero-based
//     }
//     return dates;
// }



function getDatesForMonth(year, month) {
    const daysInMonth = moment(`${year}-${month}`, 'YYYY-MM').daysInMonth();
    const dates = [];

    for (let i = 1; i <= daysInMonth; i++) {
        const date = moment(`${year}-${month}-${i}`, 'YYYY-MM-DD');
        dates.push(date.format('YYYY-MM-DD'));
    }

    return dates;
}




// Function to filter out weekends start from 0 = sunday.
function removeDayFromDate(dates, dayToRemove) {
```

# Function to remove weekly offs and public holidays from total dates



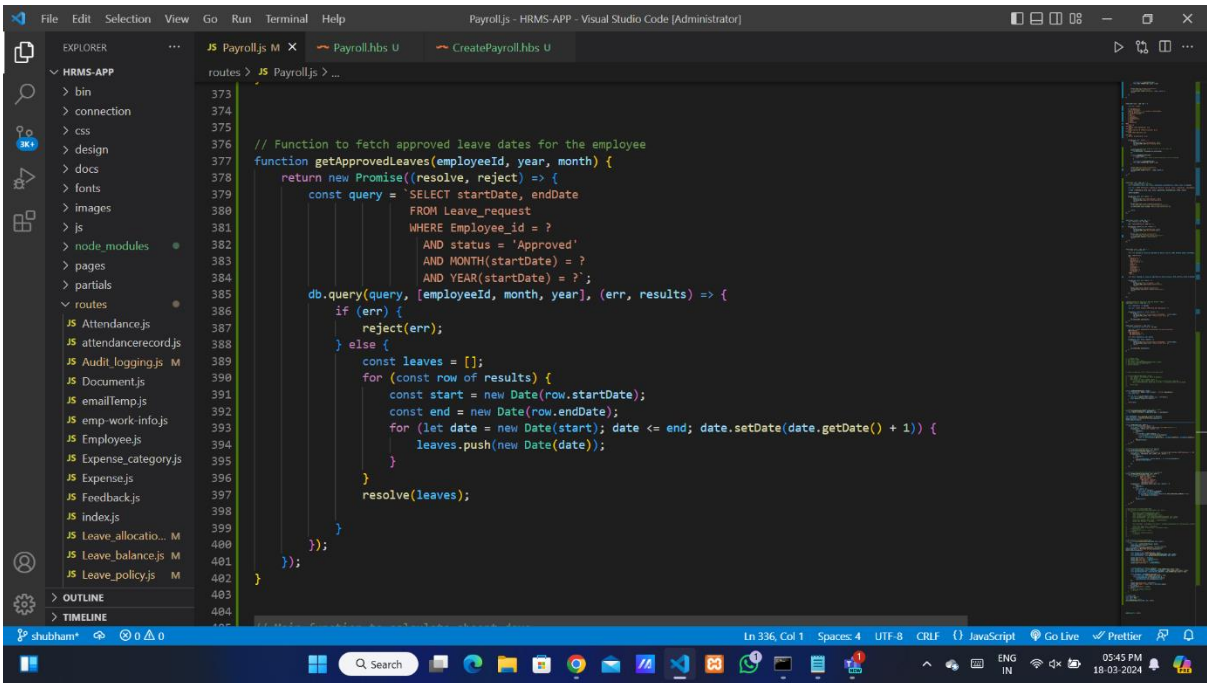
```

// Function to filter out weekends start from 0 = sunday.
function removeDayFromDate(dates, dayToRemove) {
    return dates.filter(date => moment(date).day() !== dayToRemove);
}

const dayToRemove = 0; // Assuming 3 represents Wednesday
const workingDays = removeDayFromDate(dates, dayToRemove);
console.log("Dates for March 2024 excluding Wednesdays:");
console.log(workingDays);



function getHolidays(year, month) {
    return new Promise((resolve, reject) => {
        const query = 'SELECT holidayDate FROM Holiday_lists WHERE month_id = ?';
        db.query(query, [month], (err, results) => {
            if (err) {
                reject(err);
            } else {
                const holidays = results.map(row => {
                    const holidayDate = new Date(row.holidayDate);
                    // Adjust month value by subtracting 1
                    return new Date(holidayDate.getFullYear(), holidayDate.getMonth(), holidayDate.getDate());
                });
                resolve(holidays);
            }
        });
    });
}


```

# function to get the total present days in a month from attendance table


Payroll.js - HRMS-APP - Visual Studio Code [Administrator]
Payroll.js M
Payroll.hbs U
CreatePayroll.hbs U
routes > Payroll.js > ...

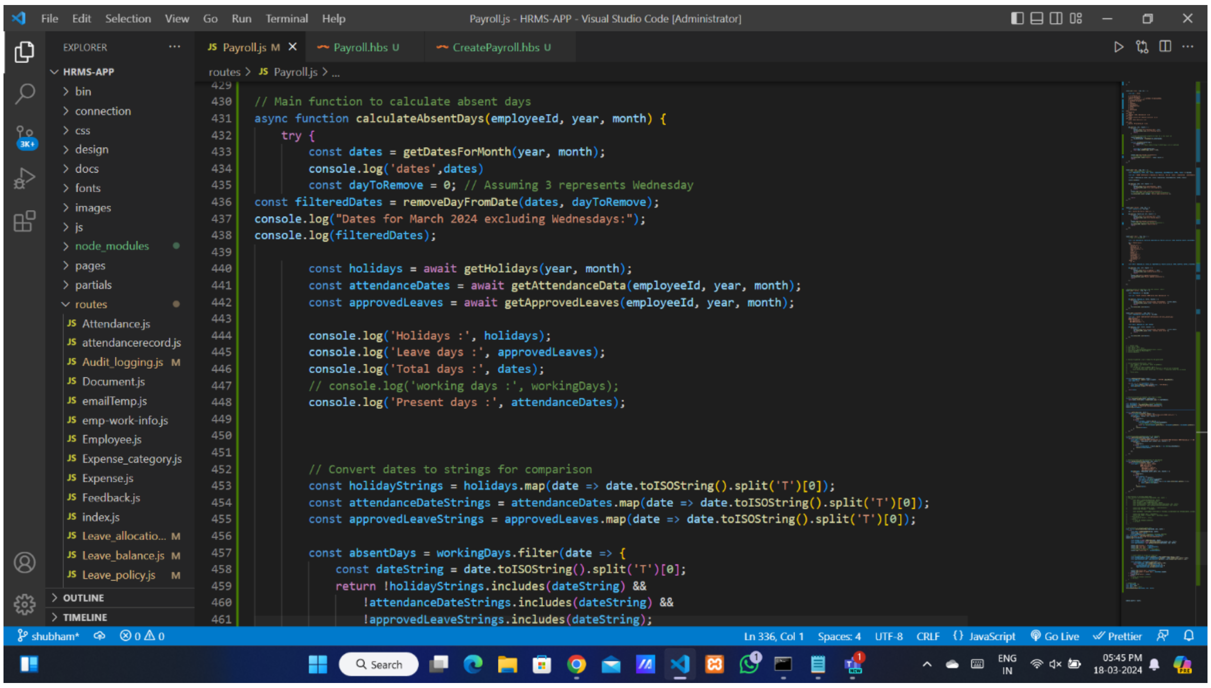
```javascript
                });
                resolve(holidays);
            }
        });
    });
}


// Function to fetch attendance data for the employee
function getAttendanceData(employeeId, year, month) {
    return new Promise((resolve, reject) => {
        const query = 'SELECT DISTINCT DATE(CheckIn) AS checkInDate FROM Attendance WHERE Employee_id = ? AND YEAR(CheckIn) = ?
        db.query(query, [employeeId, year, month], (err, results) => {
            if (err) {
                reject(err);
            } else {
                const attendanceDates = results.map(row => new Date(row.checkInDate));
                resolve(attendanceDates);
            }

        });

    });
}



// Function to fetch approved leave dates for the employee
function getApprovedLeaves(employeeId, year, month) {
    return new Promise((resolve, reject) => {
        const query = `SELECT startDate, endDate
                       FROM Leave_request
```

# function to check the approved leaves in that month

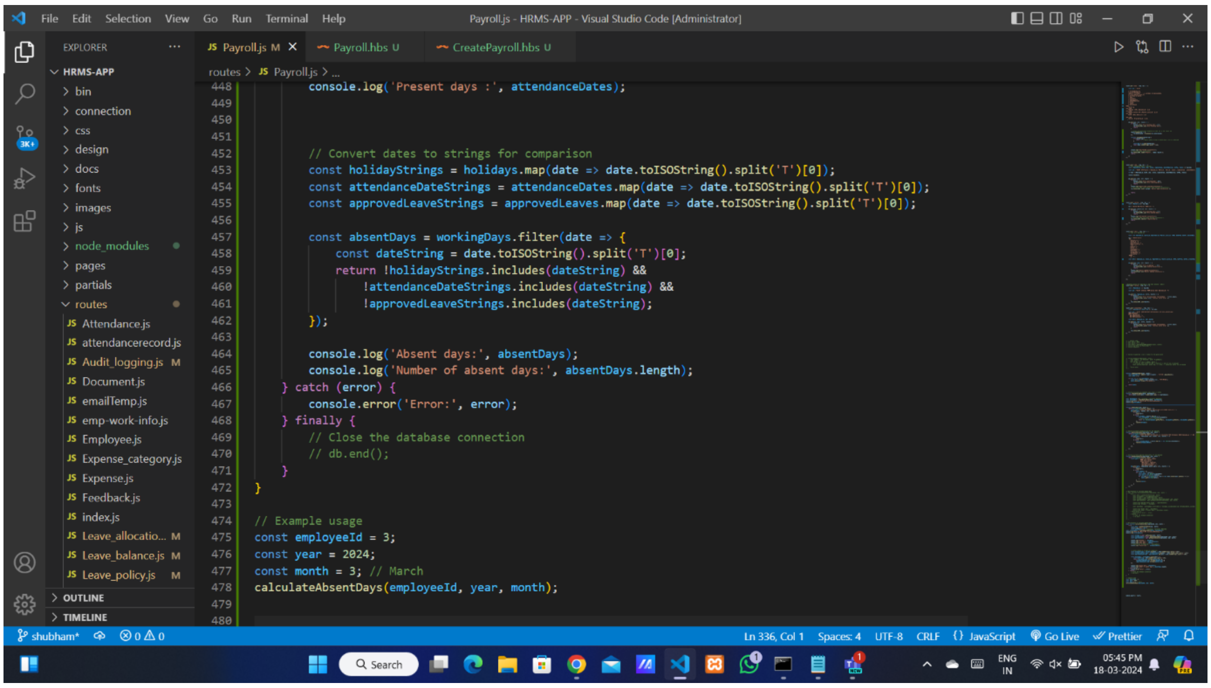

Payroll.js - HRMS-APP - Visual Studio Code [Administrator]
Payroll.js M
Payroll.hbs U
CreatePayroll.hbs U
routes > Payroll.js > ...

```javascript



// Function to fetch approved leave dates for the employee
function getApprovedLeaves(employeeId, year, month) {
    return new Promise((resolve, reject) => {
        const query = `SELECT startDate, endDate
                       FROM Leave_request
                       WHERE Employee_id = ?
                         AND status = 'Approved'
                         AND MONTH(startDate) = ?
                         AND YEAR(startDate) = ?`;
        db.query(query, [employeeId, month, year], (err, results) => {
            if (err) {
                reject(err);
            } else {
                const leaves = [];
                for (const row of results) {
                    const start = new Date(row.startDate);
                    const end = new Date(row.endDate);
                    for (let date = new Date(start); date <= end; date.setDate(date.getDate() + 1)) {
                        leaves.push(new Date(date));
                    }
                }
                resolve(leaves);

            }
        });
    });
}

```

# this function is calculating total absent days after refering attendance , holidays and leaves ; to calculate the salary deduction



```javascript
// Main function to calculate absent days
async function calculateAbsentDays(employeeId, year, month) {
    try {
        const dates = getDatesForMonth(year, month);
        console.log('dates',dates)
        const dayToRemove = 0; // Assuming 3 represents Wednesday
const filteredDates = removeDayFromDate(dates, dayToRemove);
console.log("Dates for March 2024 excluding Wednesdays:");
console.log(filteredDates);

        const holidays = await getHolidays(year, month);
        const attendanceDates = await getAttendanceData(employeeId, year, month);
        const approvedLeaves = await getApprovedLeaves(employeeId, year, month);

        console.log('Holidays :', holidays);
        console.log('Leave days :', approvedLeaves);
        console.log('Total days :', dates);
        // console.log('working days :', workingDays);
        console.log('Present days :', attendanceDates);



        // Convert dates to strings for comparison
        const holidayStrings = holidays.map(date => date.toISOString().split('T')[0]);
        const attendanceDateStrings = attendanceDates.map(date => date.toISOString().split('T')[0]);
        const approvedLeaveStrings = approvedLeaves.map(date => date.toISOString().split('T')[0]);

        const absentDays = workingDays.filter(date => {
            const dateString = date.toISOString().split('T')[0];
            return !holidayStrings.includes(dateString) &&
                !attendanceDateStrings.includes(dateString) &&
                !approvedLeaveStrings.includes(dateString);
```



```javascript
        console.log('Present days :', attendanceDates);



        // Convert dates to strings for comparison
        const holidayStrings = holidays.map(date => date.toISOString().split('T')[0]);
        const attendanceDateStrings = attendanceDates.map(date => date.toISOString().split('T')[0]);
        const approvedLeaveStrings = approvedLeaves.map(date => date.toISOString().split('T')[0]);

        const absentDays = workingDays.filter(date => {
            const dateString = date.toISOString().split('T')[0];
            return !holidayStrings.includes(dateString) &&
                !attendanceDateStrings.includes(dateString) &&
                !approvedLeaveStrings.includes(dateString);
        });

        console.log('Absent days:', absentDays);
        console.log('Number of absent days:', absentDays.length);
    } catch (error) {
        console.error('Error:', error);
    } finally {
        // Close the database connection
        // db.end();
    }
}

// Example usage
const employeeId = 3;
const year = 2024;
const month = 3; // March
calculateAbsentDays(employeeId, year, month);
```